Learning Team-Level Information Integration in Multi-Agent Communication

Anonymous authors
Paper under double-blind review

Abstract

In human cooperation, both individual knowledge and team consensus play important roles in accomplishing tasks. However, existing multi-agent reinforcement learning (MARL) communication methods commonly focus on individual-level communication, which overlooks the global team information for decision-making. Meanwhile, individual-level communication is often infeasible when the communication bandwidth is limited. To tackle these problems, we propose a team-level information integration model called *Double Channel Communication Network* (DC2Net). DC2Net highlights the significance of independent team feature learning by separating individual and team feature learning into two independent channels. In this model, agents no longer communicate with each other in a peer-to-peer paradigm; instead, all interactions are carried out in the team channel. By combining individual and global features, decisions are made collaboratively. We conduct experiments on several multi-agent cooperative environments and the results show that the DC2Net is not only competitive with other MARL communication models but also reduces the communication costs. Furthermore, the two channels can enable adaptive balancing of individual and team feature learning based on task requirements.

1 Introduction

Deep reinforcement learning (DRL) Sutton & Barto (2018) is widely applied in various domains such as autonomous vehicles Faust et al. (2018); Hoel et al. (2019), traffic light control Wiering et al. (2000), robot control Sartoretti et al. (2019); Akkaya et al. (2019), and game AI Silver et al. (2016); Berner et al. (2019). With the rise of DRL, multi-agent reinforcement learning (MARL) Busoniu et al. (2008) has been widely used in practice in recent years Shalev-Shwartz et al. (2016); Leibo et al. (2017); Roesch et al. (2020). As communication is a fundamental aspect of human interaction, MARL communication methods have been gaining increasing attention recently. Different from MARL cooperative algorithms Rashid et al. (2018); Foerster et al. (2018); Mahajan et al. (2020), communication learning aims to enrich agents' observation and make collaborative decisions based on information gain (IG). However, the majority of current MARL communication methods primarily focus on individual-level communication Foerster et al. (2016); Das et al. (2019); Jiang et al. (2020), i.e., peer-to-peer communication. These methods overlook the importance of team-level information and incur significant communication costs. For example, in a system of N agents, the maximum cost of individual-level communication is $\binom{N}{2}$. This cost becomes unmanageable when dealing with a large number of agents or limited communication bandwidth Kim et al. (2019). Furthermore, how to select appropriate partners to communicate with is another challenge that individual-level communication is faced with Jiang & Lu (2018); Ding et al. (2020).

Considering the behavior of human cooperation to complete complex tasks, it is a common strategy to reach a team consensus to help individuals to make better decisions. Several work have studied ways to raise team-level information in MARL communication Sukhbaatar et al. (2016); Singh et al. (2018). However, in these work, team information is typically considered as a supplementary factor to individual learning. The insufficient focus on team information reflects a mixture of team learning and individual learning, which can have negative impacts on decision-making. For example, in scenarios where each individual has extremely limited observations, peer-to-peer communication may not provide sufficient IG for effective agent decision-making. And the mixing of individual and team learning can lead to the gradients being backpropagated through the two parts, making it more difficult to learn accurate representations of both individual and team information. Hence, it is essential to recognize team information as a vital component of communication, treating it as an independent and prominent signal that requires feature learning at the team level.

In this paper, we propose a communication model named Double Channel Communication Network (DC2Net) which employs team-level communication and consults individual and team learning in two independent channels. In DC2Net, individual information is fed into a team channel for team information integration, and all communication is conducted exclusively within the team channel. In the team channel, aggregated individual features are learned independently by a neural network and the gradients would not affect the learning process of the individual channel, and vice versa. The learned individual and global features are used together to make decisions. Our model solves the problem of mixed learning while simultaneously reducing the costs of communication.

DC2Net is implemented based on the *Q*-learning algorithm and trained end-to-end. We experimentally show the performance of DC2Net in the multi-agent particle environment and the traffic junction environment. The proposed DC2Net outperforms several state-of-the-art MARL communication models. The experimental results also show that the two channels in DC2Net can adaptively balance individual and team feature learning for different tasks. Ablation studies are made to illustrate the contributions of different parts of the DC2Net. In addition, we tested the performance of several communication models with limited bandwidth, and the experimental results show that DC2Net can achieve good performance with fewer communication costs. Finally, we conclude with a further discussion of our proposed model.

There are three remarkable contributions in this paper:

- To tackle the challenge of high communication costs, we eliminate peer-to-peer communication, and a team channel is utilized for all communications.
- To address the problem of mixed learning, we employ two independent channels to learn individual and team features for decision-making respectively.
- We empirically validate the DC2Net in several MARL cooperative tasks and provide insights into our model. The experimental results demonstrate that our model can adaptively balance individual and team feature learning based on task requirements, while also achieving favorable results with limited communication bandwidth.

2 Background

2.1 Multi-Agent Reinforcement Learning

A standard MARL task can be modeled as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) Oliehoek & Amato (2016), which allows the distributed control by partially observable multiple agents. The Dec-POMDP consists of a tuple $\langle \mathcal{S}, \mathcal{A}, R, P, O, \Omega, N, \gamma \rangle$. $s \in \mathcal{S}$ is the true state of the environment. Each agent $i \in N$ has a partial observation $o_i \in \Omega$ according to the observation function $O(\mathbf{s}, i)$: $\mathcal{S} \times N \rightarrow \Omega$ and chooses an action $a_i \in \mathcal{A}$ based on the o_i. A joint action $\mathbf{a}$ is formed by $\mathbf{a} = [a_i] \in \mathcal{A}^N$. After taking the joint action $\mathbf{a}$, the transition probability function $P : \mathcal{S} \times \mathcal{A}^N \times \mathcal{S} \rightarrow [0, 1]$ causes the transition on the environment. The reward $R(s, \mathbf{a})$: $S \times \mathcal{A}^N \rightarrow \mathbb{R}$ is shared by all the agents. $\gamma \in [0, 1]$ is the discount factor.

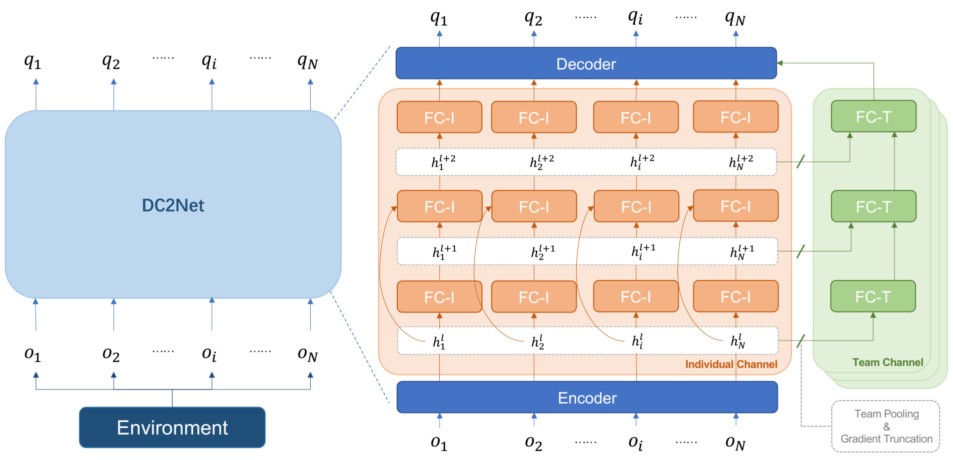
2.2 Individual-Level and Team-Level Communication

Different communication structures hold a varied impact on communication effectiveness. In this paper, we briefly define two communication structures which are individual-level communication and team-level communication.

Definition 1 *(Individual-Level Communication.) Each agent can choose one or more individuals to communicate with, i.e. peer-to-peer communication.*

Definition 2 *(Team-Level Communication.) The information interaction only takes place in a public communication channel without any peer-to-peer communication.*

Existing work Foerster et al. (2016); Das et al. (2019); Jiang et al. (2020); Niu et al. (2021) commonly focus on individual-level communication, which captures detailed communication relation-



q_1 q_2 ...... q_i ...... q_N
DC2Net
o_1 o_2 ...... o_i ...... o_N
Environment
Decoder
FC-I
h_1^{l+2} h_2^{l+2} h_i^{l+2} h_N^{l+2}
h_1^{l+1} h_2^{l+1} h_i^{l+1} h_N^{l+1}
h_1^{l} h_2^{l} h_i^{l} h_N^{l}
Individual Channel
FC-T
Team Channel
Encoder
Team Pooling & Gradient Truncation

Figure 1: *Left*: A high-level view of DC2Net's interaction with the environment. *Right*: o_i is encoded into the hidden state h_i by an encoder. The FC-I learns features in the individual channel while the skip connection ensures the reuse of the feature extracted from different layers. In the team channel, the multi-head team channel mechanism is employed to capture information from different representation subspaces. The FC-T learns the features after team pooling and gradient truncation. The decoder aggregates the features of the individual channel and team channel and outputs the Q value of each agent.

ships. Some work Sukhbaatar et al. (2016); Singh et al. (2018); Kim et al. (2019) learn team features to form team-level communication, but individual and team feature learning are often mixed which could harm the learning process. To the best of our knowledge, our proposed model, DC2Net, is the first model separating team learning from individual learning by using independent gradient backpropagation in the two channels.

3 MODEL

In this section, we will describe the proposed DC2Net as shown in Fig. 1 in detail.

3.1 HIGH-LEVEL VIEW

We use the Q-learning algorithm as the basis of our model. Agents interact with the environment and take actions based on their optimal Q function. Considering a partially observable environment, let o_i^t be the local observation of each agent i at timestep t. The input of our model $O = \{o_1, o_2, ..., o_n\}$ (omitting time index for brevity) is formed by concatenating the agents' observation. In a high-level view, the DC2Net can be regarded as a black box of an action-value Q function, which generates the Q value of the agents, where $Q = \{q_1, q_2, ..., q_n\}$.

In the DC2Net, agents do not communicate with each other under our premise of no individual-level interaction. For the agents' communication, we establish a team channel and allow information to be fused in it. In a high-level view, the individual channel and team channel are two components of the model, which are responsible for individual learning and communication learning, respectively. These two channels will be introduced in the following two subsections.

3.2 INDIVIDUAL CHANNEL

In our model, the individual channel is responsible for the independent learning of individual information. For each agent, the local observation o_i is encoded into a hidden state h_i by an encoder:

$$h_i^0 = Encoder(o_i) \tag{1}$$

where the superscript of h indicates the layer of the network. The encoder can be a CNN when there is a visual input or an MLP when there is a low-dimensional input.

After the encoding, several controllers are used to extract features of $h = \{h_1, h_2, ..., h_n\}$. Here we use fully connected neural networks as the controllers in the individual channel (FC-I in Fig. 1).

$$h_i^{l+1} = f_{FC-I}^l(h_i^l) \tag{2}$$

where l indicates the lth layer of the network.

Inspired by the skip connection in ResNetHe et al. (2016), we use skip connections to maintain and reuse the individual information in different layers.

$$h_i^{l+2} = f_{FC-I}^{l+1}(h_i^l, f_{FC-I}^l(h_i^l)). \tag{3}$$

In the individual channel, there are no communications between agents, which ensures the independence of individual learning. Meanwhile, the features in different layers can be reused by the skip connection, which contributes to individual feature extraction.

The individual learning channel is like the structure of independent *Q*-learning (IQL) Tampuu et al. (2017). IQL is a simple MARL method. Since it does not take into account the behavior of other agents, it is difficult to form a collaborative behavior. Moreover, the IQL is hard to converge due to the non-stationary of the environment. To deal with these problems, we employ a team communication channel to conduct information interaction among agents and facilitate their collaboration.

3.3 TEAM CHANNEL

The team channel is where individual information is integrated. In our opinion, the order of the individuals should not affect the output of the team channel. So first, we make a definition of a permutation invariant channel.

Definition 3 *(Permutation Invariant Channel.) We define a channel as $z = \mathcal{C}(\mathcal{I})$, where $\mathcal{C}(\cdot)$ indicates the function of the channel, z is the output of the channel, and $\mathcal{I} = \{I_i | i = 1, 2, ..., N\}$ is the information of individuals. The permutation invariant channel is formulated as:*

$$\mathcal{C}\left(\{I_i \mid i = 1, 2, \ldots, n\}\right) = \mathcal{C}\left(\left\{I_{p(i)} \mid i = 1, 2, \ldots, n\right\}\right) \tag{4}$$

where $p(i)$ is any permutation.

Team Pooling Here we use a team pooling operation to integrate the individual hidden states h_i. To show the mechanism of the team pooling, we define a hidden state map H^l which stacks the hidden state h_i^l of each agent i in the lth layer.

$$H^l = concatenate(h_i^l), i = 1, 2, ..., N. \tag{5}$$

More specifically,

$$H^l = \begin{pmatrix} h_{11}^l & h_{12}^l & \ldots & h_{1D}^l \\ h_{21}^l & h_{22}^l & \ldots & h_{2D}^l \\ \vdots & \vdots & h_{id}^l & \vdots \\ h_{N1}^l & h_{N2}^l & \ldots & h_{ND}^l \end{pmatrix} \tag{6}$$

where h_{id}^l indicates the dth dimension of hidden state h_i^l. Here we use the max pooling which is widely used in image processing Krizhevsky et al. (2012); Simonyan & Zisserman (2014) to obtain the maximum feature in each column of H^l:

$$\tilde{h}^l = \{\max_i h_{id}^l | d = 1, 2, ..., D\} \tag{7}$$

where $\tilde{h}^l$ is the team information gathered by the max pooling. As the team max pooling operation is permutation invariant, our team channel is a permutation invariant channel.

Lemma 1 *(Permutation Invariance of the Team Channel.)* The team channel of DC2Net can be abstracted as follows:

$$z = \mathcal{T}\left(\{I_i \mid i = 1, 2, \ldots, n\}\right). \tag{8}$$

For any permutation $p(i)$, $\mathcal{T}$ is permutation invariant, i.e.,

$$\mathcal{T}\left(\{I_i \mid i = 1, 2, \ldots, n\}\right) = \mathcal{T}\left(\left\{I_{p(i)} \mid i = 1, 2, \ldots, n\right\}\right) \tag{9}$$

The proof of the lemma is in Appendix A.

Since the team channel is learned independently under our premise, we make a **gradient truncation** of h_i^l. There are no gradients from individual learning in $\tilde{h}^l$ so the update of parameters in the team channel can be independent of the individual channel during the backpropagation.

Like the learning process of individuals, we use several fully connected layers to extract features in the team channel (FC-T in Fig. 1). Except for the first layer, the input of each FC-T consists of the pooled feature $\tilde{h}^l$ and the output $\tilde{h}^{l-1}$ from the previous layer:

$$\tilde{h}_i^l = f_{FC-T}^l(\tilde{h}_i^{l-1}, pooling(H^l)). \tag{10}$$

Multi-Head Team Channel Inspired by the multi-head attention mechanism Vaswani et al. (2017), we use several team channels to capture information from different representation subspaces at different positions (see Fig. 2):

$$z_m = \mathcal{T}_m(\mathcal{I}), m = 1, 2, ..., M \tag{11}$$

where m indicates the mth team channel. In this work, we employ $M = 3$ parallel team channels. For making the computational cost similar to that of a single team channel, we reduced the dimension of each FC-T in each team channel.

We use the cosine distance to measure the difference between the outputs of channels and make it a regularization term. Maximizing the cosine distance as a regularization will encourage team channels to capture different representations of the team feature, and make the learning of the total team channel more stable in different tasks.

$$D_{mm'} = \cos(z_m, z_{m'}) \quad m, m' = 1, 2, ..., M \; and \; m \neq m' \tag{12}$$

A linear function is employed to aggregate the outputs of team channels, which can be formulated as:

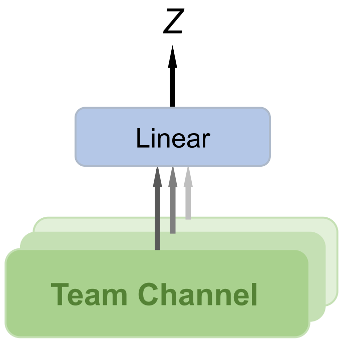


Figure 2: Multi-Head Team Channel. M team channels learn independently and the outputs are aggregated by a linear function to the total team feature z

$$z = Linear(z_m), m = 1, 2, ..., M. \tag{13}$$

The output z of the total team channel will be fed into a decoder with the output h_i^L of the individual channel to collaboratively obtain the Q-value. The decoder can be an MLP that outputs the Q value.

$$q_i = Decoder(h_i^L + z) \tag{14}$$

3.4 Training

We use the Q-learning algorithm to train our model. During the training process, we store $[\mathcal{O}, \mathcal{A}, R, \mathcal{O}']$ in the buffer, where observation $\mathcal{O} = [o_1, o_2, ...o_n]$, action $\mathcal{A} = [a_1, a_2, ...a_n]$, reward $r_1 = r_2 = ... = r_n = R$, next observation $\mathcal{O}' = [o_1', o_2', ...o_n']$. For each agent, we use the following TD loss:

$$\mathcal{L}_{TD}(\theta) = \frac{1}{\mathrm{B}} \sum_{\mathrm{B}} \frac{1}{\mathrm{N}} \sum_{i=1}^{\mathrm{N}} \left(y_i - Q\left(o_i, a_i; \theta\right)\right)^2 \tag{15}$$

where $y_i = r + Q\left(o_i', a_i'; \theta'\right)$, B is the size of a sampled minibatch. The TD loss is used to update the network. Similar to DQN Mnih et al. (2013), we also employ a target Q network and use soft update in DDPG Lillicrap et al. (2015) to update the parameters:

$$\theta' \leftarrow \tau\theta + (1 - \tau)\theta'. \tag{16}$$

As mentioned in the team channel section, a regularization term is applied to improve the output difference of different team channels. The loss can be formulated as:

$$\mathcal{L}_D(\theta) = \frac{1}{S} \sum_{m=1}^{\mathrm{M}} \sum_{m'=m+1}^{\mathrm{M}} \cos\left(z_m, z_{m'}\right) \tag{17}$$

Table 1: The settings of the environments.

Environment	**Task**	**Settings**
MPE	PP	N_1=2, N_2=4, N_3=2, T=40
TJ	Easy TJ	N=5, T=20
	Medium TJ	N=10, T=40
	Hard TJ	N=20, T=80

where $S = \binom{M}{2} = \frac{M(M-1)}{2}$ indicates the number of combinations of team channels. By combining the TD loss and multi-head team channel regularization, the total loss of our model is:

$$\mathcal{L}(\theta) = \mathcal{L}_{TD}(\theta) - \lambda \mathcal{L}_D(\theta) \tag{18}$$

where λ is the coefficient for the regularization loss.

4 Experiment

4.1 Environments

We evaluate DC2Net on a variety of tasks which are different levels of Traffic Junction (TJ) Sukhbaatar et al. (2016) environment and Predator-Prey (PP) in Multi-agent Particle Environment (MPE) Mordatch & Abbeel (2017); Lowe et al. (2017). We use the MPE with discrete actions in PettingZoo Terry et al. (2020) which is a library that inherits features of GYM Brockman et al. (2016) and is applied in MARL. The number of agents N and the number of timesteps T per episode are shown in Table 1. In the PP task, we make this task harder by setting the number of prey N_1, predators N_2, and obstacles N_3 to 2, 4, and 2, respectively. As the prey is faster than the predator, the predators must learn to capture prey in a team of two in order to accomplish this task. More environmental details are shown in Appendix B.

4.2 Baselines

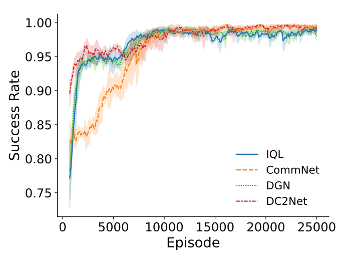
DC2Net is trained end-to-end by optimizing a neural network representation of the action-value function Q. To make the comparison fair, we use several Q-learning-based MARL communication models to test the efficiency of our model. **IQL** is a common baseline where agents are controlled independently without any communication. The structure of our model is similar to which of the **CommNet**. But there are no independent learning channels in the CommNet. Note that the original CommNet is learned through REINFORCE Sutton & Barto (2018). Here, we use the structure of CommNet to form a Q-learning-based algorithm. **DGN** learns the mutual interplay between agents by considering an underlying graph structure. In the DGN model, the attention mechanism Vaswani et al. (2017) is the main part to learn the relative importance between agents.

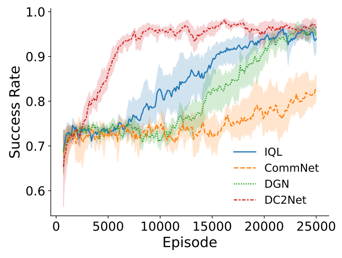
4.3 Results

Traffic Junction Fig. 3 shows the learning curves of the four models in terms of success rate on TJ. The quantitative results of success rates can be found in Table 2. Our proposed DC2Net is competitive compared to other models.

In the easy level TJ, all of the four models have similar performance with success rates of more than 99%. This is because there is only one grid of junction at the easy level. Agents just need to learn to wait when there is a car at the intersection to complete the task. In spite of this, from the perspective of convergence speed, DC2Net makes faster learning than IQL and DGN, and far beyond CommNet.

In the medium level TJ, DC2Net reaches 97.39% in success rate. The results of IQL and DGN are comparable, but their convergence speed is both far slower than that of DC2Net. In both easy and medium levels, the performance of CommNet is poor compared to the other three models. We think this is because instead of being effective, the team information generated by averaging operations could affect individual learning. Since these two tasks are not particularly difficult (only one junction), providing too much information in the individual learning process will make it more difficult to capture features. The comparable performance of IQL also proves that excessive communication is not required in these two environments, individual learning is crucial to complete both tasks. We conclude that these two methods make too many communications that harm individual learning.


(a) Easy


(b) Medium

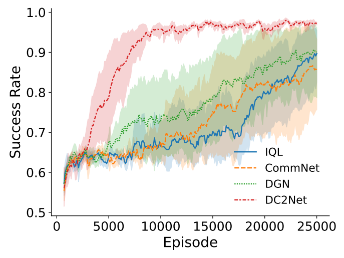

(c) Hard

Figure 3: Experimental results on TJ. Shaded regions are standard deviations over 5 runs.

Table 2: Success rates on TJ.

Model	Easy	Medium	Hard
IQL	99.02%	97.02%	89.88%
CommNet	99.22%	92.86%	85.43%
DGN	99.14%	97.22%	90.45%
DC2Net	**99.42%**	**97.39%**	**97.14%**

In the hard level TJ, DC2Net has absolute superiority. DC2NET achieves a winning rate of 97.14% compared with the other three methods reaching about 90%. The poor performance of IQL in the early stage shows that this task requires communication to speed up learning, and the other two communication methods can obtain a faster convergence speed through communication. Nevertheless, DC2Net's learning speed is significantly the fastest. Comparing the results at the medium level and hard level, we can find that DC2Net performs well both in the task that emphasizes individual learning and the task that emphasizes communication learning, which shows that the DC2Net can balance individual learning and communication learning through the two channels, and maintain stable performance in different tasks.

Predator-Prey To make PP a cooperative task, we control the actions of predators to capture randomly acted prey. The experimental results are shown in Fig. 4 in terms of mean reward, averaged over all agents and timesteps.

In our settings of agent number, the predators should learn to capture prey in a team of two to complete the task. It can be seen that in this cooperative predation task, due to the lack of communication, the agents in IQL cannot perform well in teaming. The performance of DC2Net and DGN is better because these two methods use specific network structures for communication learning (compared with the simple average operation in CommNet), but from the perspective of convergence speed, DC2Net is still better than DGN.

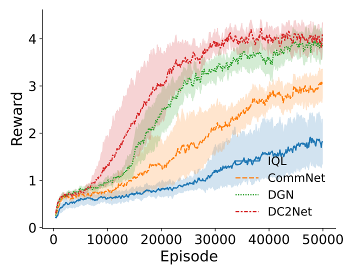


Figure 4: Experimental results on PP. Shaded regions are standard deviations over 5 runs.

4.4 Ablation Study

We perform experiments to investigate the influence of the components in the DC2Net. Fig. 5 gives the ablation results of the DC2Net without the multi-head mechanism (DC2Net-M), the gradient truncation (DC2Net-G), and the team channel (execution phase only) (DC2Net-T), respectively. The experimental results are as shown in Fig. 5. DC2Net-T has the poorest performance, which shows the importance of the team channel in execution. Under the condition of no gradient truncation, mixed learning of the team channel and individual channel leads to an unfavorable result shown by DC2Net-G. This confirms the importance of independent learning in the individual and team channels. The result of only using a single team channel is comparable, but the DC2Net has a better performance by employing the multi-head team channel mechanism. For these phenomena, we believe that these three components make contributions to our model.

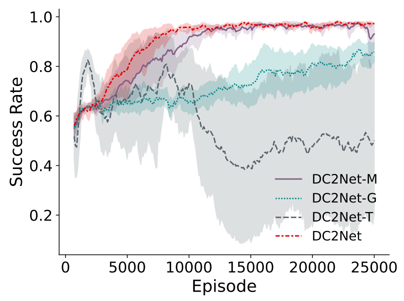


Figure 5: The results of the ablation study on Hard TJ.

Table 3: Amount of information transmitted in communication per timestep (the fewer transmissions, the smaller the channel occupation). L: number of network layers, N: number of agents, D: number of information dimensions

	CommNet	**DGN**	**DC2Net**
Amount of information (bits/timestep)	$2LND$	LN^2D	$(L+1)ND$

Further, we demonstrate the effectiveness of the multi-head team channel. Multi-head team channel is employed to capture information from different representation subspaces. We downscale and visualize the output of the three team channels (see Fig. 6). It can be seen that the outputs of the three channels form three clusters, which indicates that our proposed multi-head team channel mechanism can explore and capture different features in multiple subspaces. We have reason to believe that the multi-head team channel has a stronger ability to extract team features than a single channel.

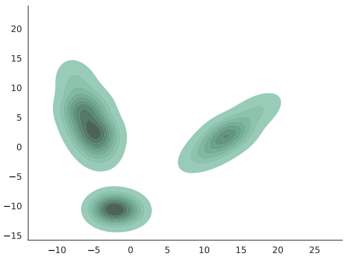


Figure 6: Visualization of the output of the three team channels after dimensionality reduction by PCA.

4.5 Discussion

Why Team Pooling? Learning representations is crucial for deep learning. Pooling, as a common technology in computer vision, is widely used in image dimensionality reduction. We choose pooling when there are many methods such as linear function, MLP, attention, etc. that can be used to aggregate individual information for two reasons. First, it is simple. The pooling operation does not introduce any new parameters, it's just a maximization operation in the specified dimension, which reduces the size of our network while speeding up the computation. Second, by analogy with the max pooling in image processing, we can extract the most important parts from the hidden states while reducing the dimensionality. More importantly, the pooling operation is proven to be a universal approximator for any set function Zaheer et al. (2017). So theoretically we can learn any mapping from individuals to a team through the team pooling.

Does the DC2NET Communicate Less? We list the amount of information transmitted in communication per timestep of the three communication models and the results are shown in Table 3. It can be seen that the DC2NET has less communication theoretically. Empirically, we set up a scenario on hard TJ with limited communication bandwidth (A limited amount of bits per episode during execution. The model can only make decisions based on individual information after the limited communication bandwidth has been exhausted). The experimental results of the three models in this scenario are shown in Fig. 7 (the similarly colored curves are the performance of the same model at full and limited bandwidths). It can be seen that DC2Net maintains favorable performance under the condition of limited bandwidth. However, the performance of the other two communication models has declined significantly under this condition.

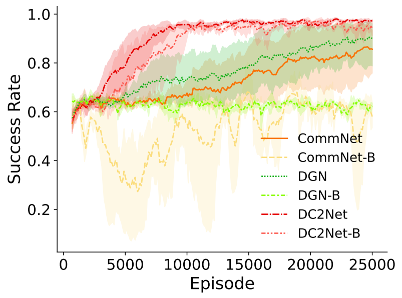


Figure 7: Experimental results on hard level TJ at fixed bandwidth.

Another Perspective to Understand Team Channel. In the previous introduction, the team channel is where the team information is transmitted. But from another perspective, the team channel can be regarded as a virtual agent. The observation of this agent is the aggregation (through the max pooling operation) of all other agents' observations. This virtual agent does not interact with the environment, but hands over the information learned by itself to other agents for their learning and decision-making. In this way, we can employ several virtual agents to help with learning, and this is in fact our multi-head team channel mechanism.

5 Related Work

5.1 Multi-Agent Reinforcement Learning

MARL uses RL-based algorithms to make a control on multi-agent systems. In the field of MARL, the main research directions can be divided into four parts Hernandez-Leal et al. (2018) which are analysis of emergent behaviors, learning communication, learning cooperation, and agents modeling agents. Our work is related to the field of differentiable learning of communication by training a well-designed neural network via backpropagation.

5.2 Learning Communication in MARL

Multi-agent communication was first proposed in RIAL and DIAL Foerster et al. (2016). The communication message is generated in the training process of an end-to-end neural network in these two methods. CommNet Sukhbaatar et al. (2016) trains a neural network model to learn a communication protocol. The communication message is implicitly transmitted in the hidden layer of the network. This model consists of multiple agents and the communication between them is learned alongside their policy. ATOC Jiang & Lu (2018) uses an attentional communication model to learn when to communicate and how to integrate shared information for making cooperative decisions. The model leads to efficient and effective communication for large-scale multi-agent cooperation. IC3Net Singh et al. (2018) controls continuous communication with a gating mechanism. It aims to solve multi-agent tasks by learning when to communicate while continuous communication enables efficient training by backpropagation. TarMAC Das et al. (2019) proposes a targeted communication architecture to learn a targeting behavior solely from downstream task-specific reward without any communication supervision. The agents learn both what messages to send and whom to address them to while performing cooperative tasks in partially-observable environments. SchedNet Kim et al. (2019) studies a scenario with limited communication bandwidth and learns to decide which agents should be entitled to broadcasting their messages by learning the importance of each agent's partially observed information. DGN Jiang et al. (2020) considers the underlying graph of agents and proposes graph convolutional reinforcement learning. The graph convolution can adapt to the dynamics of the underlying graph of the multi-agent environment, and the latent features produced by convolutional layers from gradually increased receptive fields are exploited to learn cooperation. MAGIC Niu et al. (2021) learns a scheduler to decide when to communicate and whom to address messages to, and a message processor to deal with communication signals. It's a model based on individual-level communication. FCMNet Wang & Sartoretti (2022) learns a multi-hop communications protocol based on recurrent neural networks, which enables team-level decision-making. This work uses a common channel for communication but does not realize the independent channel learning. We list the usage of team information of these work in Table 4 as a brief comparison.

Table 4: Usage of team information of several MARL communication methods.

Method	**Team Info**
RIAL & DIAL	✘
CommNet	✔
ATOC	✘
IC3Net	✔
TarMAC	✘
SchedNet	✔
DGN	✘
MAGIC	✘
FCMNet	✔
DC2Net (this paper)	✔

6 Conclusion

In this paper, we proposed the DC2Net for multi-agent communication by learning in two independent channels which are responsible for individual learning and communication learning. The team-level communication not only effectively captures the team feature but also reduces the costs of communication. Empirically, the DC2Net model outperforms several state-of-the-art MARL communication models on a variety of cooperative multi-agent tasks. The experiment also illustrates that DC2Net can balance individual learning and communication learning in different tasks.

For a broader impact, we demonstrate the effectiveness of independent learning in both individual and team channels for multi-agent communication. The team-level learning manner deserves to be studied more widely.

REFERENCES

Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym, 2016.

Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning*, pp. 1538–1546. PMLR, 2019.

Ziluo Ding, Tiejun Huang, and Zongqing Lu. Learning individually inferred communication for multi-agent cooperation. *Advances in Neural Information Processing Systems*, 33:22069–22079, 2020.

Aleksandra Faust, Kenneth Oslund, Oscar Ramirez, Anthony Francis, Lydia Tapia, Marek Fiser, and James Davidson. Prm-rl: Long-range robotic navigation tasks by combining reinforcement learning and sampling-based planning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 5113–5120. IEEE, 2018.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to Communicate with Deep Multi-Agent Reinforcement Learning. *arXiv:1605.06676 [cs]*, May 2016. URL `http://arxiv.org/abs/1605.06676`. arXiv: 1605.06676.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor. Is multiagent deep reinforcement learning the answer or the question? a brief survey. *learning*, 21:22, 2018.

Carl-Johan Hoel, Katherine Driggs-Campbell, Krister Wolff, Leo Laine, and Mykel J Kochenderfer. Combining planning and deep reinforcement learning in tactical decision making for autonomous driving. *IEEE transactions on intelligent vehicles*, 5(2):294–305, 2019.

Jiechuan Jiang and Zongqing Lu. Learning Attentional Communication for Multi-Agent Cooperation. *arXiv:1805.07733 [cs]*, November 2018. URL `http://arxiv.org/abs/1805.07733`. arXiv: 1805.07733.

Jiechuan Jiang, Chen Dun, Tiejun Huang, and Zongqing Lu. Graph Convolutional Reinforcement Learning. *arXiv:1810.09202 [cs, stat]*, February 2020. URL `http://arxiv.org/abs/1810.09202`. arXiv: 1810.09202.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to Schedule Communication in Multi-agent Reinforcement Learning. *arXiv:1902.01554 [cs]*, February 2019. URL `http://arxiv.org/abs/1902.01554`. arXiv: 1902.01554.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60:84 – 90, 2012.

Joel Z. Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel. Multi-agent reinforcement learning in sequential social dilemmas. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pp. 464–473, Richland, SC, 2017. International Foundation for Autonomous Agents and Multiagent Systems.

Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Neural Information Processing Systems (NIPS)*, 2017.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: Multi-Agent Variational Exploration. *arXiv:1910.07483 [cs, stat]*, January 2020. URL `http://arxiv.org/abs/1910.07483`. arXiv: 1910.07483.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing Atari with Deep Reinforcement Learning. *arXiv:1312.5602 [cs]*, December 2013. URL `http://arxiv.org/abs/1312.5602`. arXiv: 1312.5602.

Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *arXiv preprint arXiv:1703.04908*, 2017.

Yaru Niu, Rohan R Paleja, and Matthew C Gombolay. Multi-agent graph-attention communication and teaming. In *AAMAS*, pp. 964–973, 2021.

Frans A Oliehoek and Christopher Amato. *A concise introduction to decentralized POMDPs*. Springer, 2016.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. *arXiv:1803.11485 [cs, stat]*, June 2018. URL `http://arxiv.org/abs/1803.11485`. arXiv: 1803.11485.

Martin Roesch, Christian Linder, Roland Zimmermann, Andreas Rudolf, Andrea Hohmann, and Gunther Reinhart. Smart grid for industry using multi-agent reinforcement learning. *Applied Sciences*, 10(19):6900, 2020.

Guillaume Sartoretti, Yue Wu, William Paivine, TK Kumar, Sven Koenig, and Howie Choset. Distributed reinforcement learning for multi-robot decentralized collective construction. In *Distributed autonomous robotic systems*, pp. 35–49. Springer, 2019.

Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. Safe, multi-agent, reinforcement learning for autonomous driving. *arXiv preprint arXiv:1610.03295*, 2016.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. *arXiv preprint arXiv:1812.09755*, 2018.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning Multiagent Communication with Backpropagation. *arXiv:1605.07736 [cs]*, October 2016. URL `http://arxiv.org/abs/1605.07736`. arXiv: 1605.07736.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente. Multiagent cooperation and competition with deep reinforcement learning. *PLOS ONE*, 12(4):e0172395, April 2017. ISSN 1932-6203. doi: 10.1371/journal.pone.0172395. URL `https://dx.plos.org/10.1371/journal.pone.0172395`.

J. K Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sulivan, Luis Santos, Rodrigo Perez, Caroline Horsch, Clemens Dieffendahl, Niall L Williams, Yashas Lokesh, Ryan Sullivan, and Praveen Ravi. Pettingzoo: Gym for multi-agent reinforcement learning. *arXiv preprint arXiv:2009.14471*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Yutong Wang and Guillaume Sartoretti. Fcmnet: Full communication memory net for team-level cooperation in multi-agent systems. In *Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '22, pp. 1355–1363, Richland, SC, 2022. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450392136.

Marco A Wiering et al. Multi-agent reinforcement learning for traffic light control. In *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, pp. 1151–1158, 2000.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. *Advances in neural information processing systems*, 30, 2017.

A Proof of Lemma 1

Proof: As the hidden states are parallelly pooled and computed, the output of the team channel is permutation invariant with regard to the input order. We present the simple proof here.

The aggregation process in the team channel can be abstracted as:

$$z = \mathcal{T}\left(\{I_i \mid i = 1, 2, \ldots, n\}\right). \tag{19}$$

In Eq.equation 19, the output z is computed as follows:

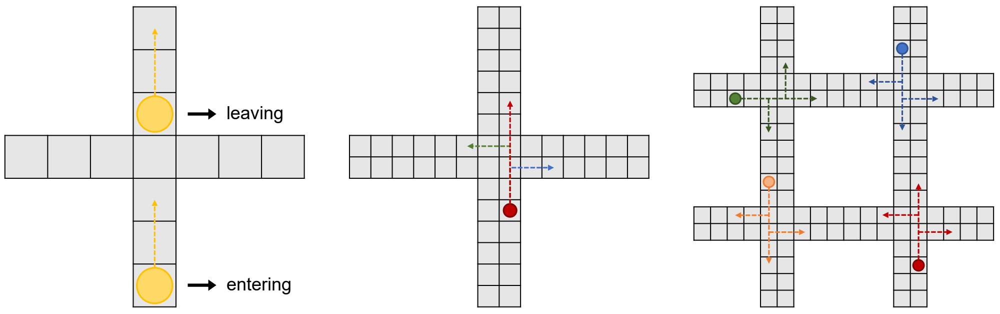
$$\begin{aligned} z &= Linear(\{z_i \mid i = 1, 2, \ldots, m\}) \\ &= Linear(\{f_{FC-T}(\tilde{h}_i, pooling(H))) \mid i = 1, 2, \ldots, m\}) \\ &= Linear(\{f_{FC-T}(\tilde{h}_i, \max_i h_{id})) \mid i = 1, 2, \ldots, m\}) \end{aligned}$$

where the $Linear(*)$, $f_{FC-T}(*)$, and $max(*)$ operation are permutations invariant to the index of the agents. Therefore, the team channel output z is permutation invariant to the input set $\{I_i \mid i = 1, 2, \ldots, n\}$.

B Environmental Settings

Traffic Junction This environment consists of tasks in three difficulty levels (see Fig. 8) on a $L \times L$ grid. At each timestep, "new" cars enter the grid with a probability from each of the four directions. The vision of the cars is 1 (3×3 gird). The total number of cars at any given time is limited to N. Each car occupies a single cell at any given time and is randomly assigned to one of the possible routes while keeping to the right-hand side of the road. At each timestep, a car has two possible actions: *gas* which advances it by one cell on its route or *brake* to stay at its current location. A car will be removed once it reaches its destination at the edge of the grid.

- Easy level TJ: A junction of two one-way roads on a 7×7 grid.
- Medium level TJ: A junction of two two-way roads on a 14×14 grid.
- Hard level TJ: Four connected junctions of two-way roads on an 18×18 grid.



(a) TJ-Easy (b) TJ-Medium (c) TJ-Hard

Figure 8: Three different level tasks in TJ. New cars enter the grid with a probability from each of the four directions. A car will be removed once it reaches its destination at the edge of the grid. All of the cars keep to the right-hand side of the road.

The reward of car i is defined as:

$$r_i^t = C_i^t r_{collision} + \tau_i r_{time} \tag{20}$$

where $r_{collision} = -10$, $r_{time} = -0.01$. The car i has C_i^t collisions at timestep t and has spent time τ_i in the junction. And the team reward is defined as:

$$r_{team}^t = \sum_{i=1}^{n} r_i^t \tag{21}$$

Predator-Prey: The PP task (see fig. 9) has N_1 preys (green) which are faster and receive a negative reward for being hit by predators (reward -10 for each collision). N_2 predators (red) are slower and are rewarded for hitting good agents (reward +10 for each collision). N_3 obstacles (black) block the way. By default, $N_1 = 1$, $N_2 = 3$ and $N_3 = 2$.

Here, we make this task harder by changing the number of predators and prey to 4 and 2, respectively. As the prey is faster than the predator, the predators must learn to capture prey in a team of two in order to accomplish this task. The length T of each episode is 40 timesteps.

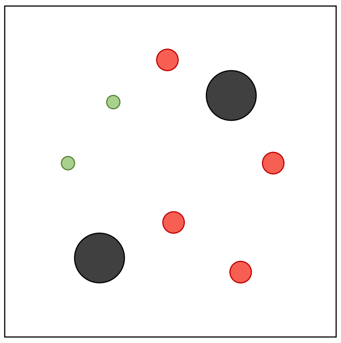


Figure 9: PP in MPE. In our settings, the number of predators (red) and prey (green) are 4 and 2, respectively.

The reward of car i is defined as:

$$r_i^t = -d_i^t \tag{22}$$

where d_i^t is the distance of prey i to its nearest predator. The team reward is defined as:

$$r_{team}^t = -\sum_{i=1}^{n} r_i^t + C^t r_{collision} \tag{23}$$

where $r_{collision} = -1$, C^t is the number of collisions (when the distance between two agents is less than the sum of their sizes) that occurred at timestep t.

C Implementation Details

In both traffic junction and predator-prey environments, our model is trained based on the Q-learning algorithm. The individual and team channels are realized by three fully connected layers. The size of the hidden layers is 72. All of the models in the paper use the same set of hyperparameters settings. In addition to the mechanisms presented in the paper, we didn't use any tricks such as learning rate annealing, gradient clipping, reward shaping, etc. to improve the performance particularly for our model.

Table 5 summarizes the hyperparameters used by DC2Net and the baselines in the experiments.

Table 5: The hyperparameters in the experiments.

Hyperparameter	IQL	CommNet	DGN	DC2Net
MLP layers	3			
hidden layer size	72			
batch size	64			
buffer size	100,000			
γ (discount factor)	0.99			
ϵ (epsilon greedy)	0.9 to 0.02			
τ (soft update)	0.96			
activation	ReLU			
optimizer	RMSprop			
learning rate	0.0005			