Individualized Controlled Continuous Communication Model for Multiagent Cooperative and Competitive Tasks

Amanpreet Singh[1,*] amanpreet@nyu.edu
Tushar Jain[1,*] tushar@nyu.edu
Sainbayar Sukhbaatar [1] sainbar@cs.nyu.edu
[1]Department of Computer Science
Courant Institute, New York University

Abstract

Learning when to communicate and doing that effectively according to level of inter-agent communication allowed is essential in multi-agent tasks. Continuous communication allows efficient training with back-propagation, but is limited to fully-cooperative tasks as agents have to expose all of their information while sharing their hidden states. Credit assignment also becomes important with increased scaling in these scenarios as agents must know their individual contributions to perform better. In this paper, we present Individualized Controlled Continuous Communication Model (IC3Net) which controls continuous communication with a gating mechanism and uses individualized rewards for each agent to gain better performance and scalability. This model has better training efficiency than simple continuous communication model, and can be applied to semi-cooperative and competitive settings along with the cooperative settings. We test our architecture on variety of tasks including StarCraft BroodWars™micro-management scenarios, to show that our architecture yields improved performance and convergence rates with increasing scale in all of the scenarios than the baselines.

1 Introduction

Communication is an essential element of intelligence as it helps in learning from others experience, work better in teams and pass down knowledge. In multi-agent settings, communication allows agents to cooperate towards common goals. Particularly in partially observable environments, when the agents are observing different parts of the environment, they can share information and learnings from their observation through communication.

Recently there have been a lot of successes in the field of reinforcement learning (RL) in playing Atari Games [16], playing Go [21] etc, most of which have been in the single agent domain. However, the number of systems and applications having multi-agents have been growing from team of robots working in manufacturing plants to a network of self-driving cars. Thus, it is crucial to successfully *scale* RL to multi-agent environments in order to build intelligent systems capable of higher productivity.

There are different kinds of interactions going on between agents, from coordination to competition to combination (mixed) of both coordination and competition for which we need to develop scalable systems. Moreover, scenarios other than cooperative (CS), namely semi-cooperative (or mixed) scenarios (MS) and competitive (or adversarial) scenarios (AS) have not even been studied as extensively for multi-agent systems.

[*]Equal Contribution

The mixed scenarios can be compared to most of the real life scenarios as humans are cooperative but not fully-cooperative in nature. Humans work towards their individual goals while cooperating with each other. In competitive scenarios, agents are essentially competing with each other for better rewards. In real life, humans always have an option to communicate but can choose when to actually communicate. For example, in a sports match two teams which can communicate, can choose to not communicate at all (to prevent sharing strategies) or use dishonest signaling (to misdirect opponents) [12] in order to optimize their own reward and handicap opponents. Similarly, our agents should also learn to choose *when* to communicate in order to perform better.

Recently, there has been a lot of interest in emergent communication in multi-agent settings using deep learning [11, 17] Teaching agents how to communicate makes it is unnecessary to hand code the communication protocol with expert knowledge. While the content of communication is important, it is also important to know when to communicate either to increase scalability and performance (CS, MS) or to increase competitive edge (AS). For example, an agent needs to learn when to communicate to avoid communicating its location with its enemies.

Sukhbaatar et al. [23] showed that agents communicating through a continuous vector are easier to train and have a higher information throughput than communication based on discrete symbols. Their continuous communication is differentiable, so it can be trained efficiently with back-propagation algorithm. However, their model assumes full-cooperation between agents and average their rewards. This restricts the model from being used in MS or AS as full-cooperation involves sharing hidden states to everyone leading to poor performance by all agents as shown by our results. Furthermore, the average global reward for all agents makes the credit assignment problem even harder and difficult to scale as agents don't know their individual contributions in MS or AS where they want themselves to succeed before others.

In this paper, we propose Individualized Controlled Continuous Communication Model (IC3Net), having both discrete and continuous communication elements with individualized credit assignment for each agent, which can be applied to any scenario regardless of cooperation or not with higher performance and better scalability. The content of communication is based on a continuous vector similar to [23], but the decision about whether to communicate or not is based on a discrete action. Since the content of communication is continuous, we can employ back-propagation for efficient training. On the other hand, the discrete action allows the model to be used even in competitive setting since it is trained to maximize the agent's own reward. Moreover, the convergence rate is better even in fully-cooperative tasks because of more scalable reward management system.

2 Model

In this section, we introduce our model **Individualized Controlled Continuous Communication Model (IC3Net)** as shown in figure 1 to work in multi-agent cooperative, competitive and mixed settings where agents learn what to communicate as well as when to communicate.

First, let us describe an independent controller model where each agent is controlled by an individual LSTM. For the j-th agent, its policy takes the form of:

$$\begin{aligned} h_j^{t+1}, s_j^{t+1} &= LSTM(e(o_j^t), h_j^t, s_j^t) \\ a_j^t &= \pi(h_j^t), \end{aligned}$$

where o_j^t is the observation of the j-th agent at time t, $e(\cdot)$ is an encoder function parameterized by a fully-connected neural network and π is an agent's action policy. Also, h_j^t and s_j^t are the hidden and cell states of the LSTM. We use the same LSTM model for all agents, sharing their parameters. This way, the model is invariant to permutations of the agents.

IC3Net extends this independent controller model by allowing agents to communicate their internal state, gated by a discrete action. The policy of the j-th agent in a IC3Net is given by

$$\begin{aligned} g_j^{t+1} &= f^g(h_j^t) \\ h_j^{t+1}, s_j^{t+1} &= LSTM(e(o_j^t) + c_j^t, h_j^t, s_j^t) \\ c_j^{t+1} &= \frac{1}{J-1} C \sum_{j' \neq j} h_{j'}^{t+1} \odot g_{j'}^{t+1} \\ a_j^t &= \pi(h_j^t), \end{aligned}$$

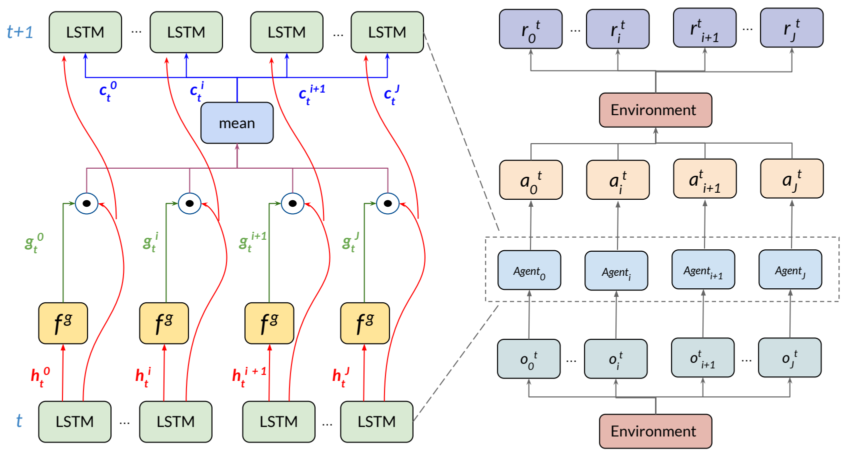


Figure 1: An overview of IC3Net. (Left) In-depth view of a single communication step. LSTM gets hidden state, h_t and cell state, s_t (not shown) from previous time-step. Hidden state h_t is passed to Communication-Action module f_g for a communication binary action g_t. Finally, communication vector c_t is calculated by averaging hidden states of other active agents gated by their communication action ac_t and is passed through a linear transformation C before fed to LSTM along with the observation. (Right) High-level view of IC3Net which optimizes individual rewards r^t for each agent based on observation o_t

where c_j^t is the communication vector for the j-th agent, C is a linear transformation matrix for transforming gated average hidden state to a communication tensor, J is the number of alive agents currently present in the system and $f^g(.)$ is a simple network containing a soft-max layer for 2 actions (communicate or not) on top of a linear layer with non-linearity. The binary action g_j^t specifies whether agent j wants to communicate with others, and act as a gating function when calculating the communication vector. Note that the gating action for next time-step is calculated at current time-step.

We train both the action policy π and the gating function f^g with REINFORCE [26]. Unlike Sukhbaatar et al. [22]'s model, each agent is trained to maximize their *individual* expected cumulative reward instead of expected cumulative global reward. This has two benefits: (i) it allows the model to be applied to both cooperative and competitive scenarios, (ii) it also helps resolve the credit assignment issue faced by many multi-agent[23, 4] algorithms while improving performance with scalability and is coherent with the findings in Chang et al. [3].

3 Related work

The simplest approach in multi-agent reinforcement learning (MARL) settings is to use an independent controller for each agent. This was attempted with Q-learning in Tan [24]. However, in practice it performs poorly [15], which we also show in comparison with our model. The major issue with this approach is that due to multiple agents, the stationarity of the environment is lost and naïve application of experience replay doesn't work well.

The nature of interaction between agents can either be cooperative, competitive, or both. Most algorithms are designed only for a particular nature of interaction and the most studied are cooperative settings [18, 10, 14], with strategies which indirectly arrive at cooperation via sharing policy parameters [7]. These algorithms are generally not applicable in competitive or mixed settings. See [2] for survey of MARL in general and [19] for survey of cooperative multi-agent learning.

Our work can be considered as a derivative of Sukhbaatar et al. [23]'s CommNet model for collaboration among multiple agents using continuous communication usable only in cooperative settings. Due to continuous communication, the controller can be learned via backpropagation. However, this model is restricted to fully cooperative tasks as hidden states are fully communicated to others which exposes everything about agent. On the other hand, due to global reward for all agents, CommNet also suffers from credit assignment issue. We resolve these issues as described in section 2.

The Multi-Agent Deep Deterministic Policy Gradient (MADDPG) model presented by Lowe et al. [13] also tries to achieve similar goals. However, they differ in the way of providing the coordination signal. In their case, no direct communication among agents (actors with 1a different policy per agent) takes place but instead, a different centralized critic per agent which can access the actions of all the agents provide the signal. Concurrently at that same time the COMA model by Foerster et al. [6] was then proposed to tackle the challenge of multiagent credit assignment through the introduction of the counterfactual reward, the idea of training multiagent systems in the centralised critic and decentralised actors similar to how it is adopted in MADDPG.

The work by Foerster et al. [5] also learns a communication protocol. However, the agents communicate only in a discrete manner through their actions. This contrasts with our model where multiple continuous communication cycles can be used at each time step to decide the actions of all agents. Furthermore, our approach is amenable to dynamic variation in the number of agents.

Peng et al. [20] also attempts to solve micromanagement tasks using communication. However, they have non-symmetric addition of agents in communication channel and are restricted to only cooperative scenarios.

4 Experiments[2]

To demonstrate the flexibility of our architecture in solving cooperative, mixed and adversarial tasks where communication is open as in real world, we consider three environments with variation in difficulty and coordination.

First, a predator-prey environment where predators with limited vision look for a prey on a square grid. Second, a traffic junction environment similar to Sukhbaatar et al. [23] where agents with limited vision must learn to communicate in order to avoid collisions. Third, we test our architecture on StarCraft BroodWars[3] explore and combat tasks which tests control on multiple agents in various scenarios while testing observation understanding for multiple opposition units. Finally, we show how communication action is learned in IC3Net on competitive and cooperative scenarios of Predator-Prey and StarCraft explore tasks.

4.1 Setup

We compare IC3Net with **baselines** specified below in all scenarios.

Individual Reward Independent Controller: In this controller, model is applied individually to all of the agents' observations to produce the action to be taken. We implement this controller as an LSTM network and there is no communication between the agents. We also use individualized rewards for each agent unlike global rewards in CommNet.

CommNet: Introduced in Sukhbaatar et al. [23], CommNet allows communication in between agents over a channel where the agent is provided hidden state representations of other agents as a communication signal. Like IC3Net, CommNet also uses continuous signals to communicate in between the agents. We average the hidden states from other agents before passing to the agent.

Training: We set the hidden layer size to 128 units and we use LSTM [9] with recurrence for all of the baselines and IC3Net. We use RMSProp [25] with initial learning rate as a tuned hyper-parameter. All of the models use skip-connections [8]. The training is distributed over 16 cores and each core runs a mini-batch till total episodes steps are 500 or more. We do 10 weight updates per epoch. We run all experiments for 1000 epochs and report the final results. In mixed case, we report the mean score of all agents, while in cooperative case we report any agent's score as they are same. We implement our model using PyTorch and environments using Gym [1].We use REINFORCE [26] to train our setup. We conduct 5 runs on each of the tasks to compile our results. We set communication action to 1 for results of next subsections and we separately analyze communication action and its working in subsection 4.5. The training time for different tasks varies; StarCraft tasks usually takes more than a day (depends on number of agents and enemies), while predator-prey and traffic junction tasks complete under 12 hours.

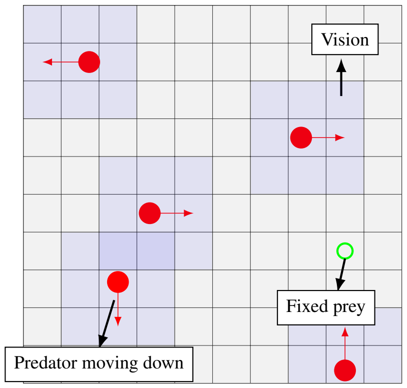
4.2 Predator prey

In this task, we have n agents with limited vision trying to find a stationary prey. Once an agent is on top of a prey, it stays there and always gets a positive reward, until rest of the agents reach the prey or

[2]The code is available at https://github.com/IC3Net/IC3Net.

[3]StarCraft is a trademark or registered trademark of Blizzard Entertainment, Inc., in the U.S. and/or other countries. Nothing in this paper should not be construed as approval, endorsement, or sponsorship by Blizzard Entertainment, Inc

the episode reaches its maximum number of steps. In case of zero vision, agents don't have a direct way of knowing where prey is located unless they jump on it.



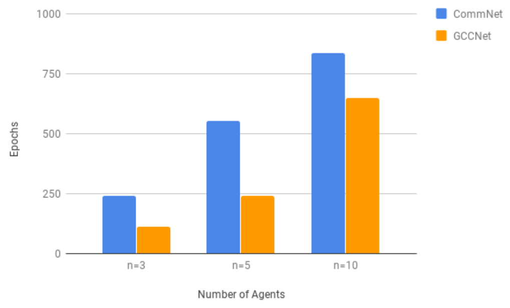


Figure 2: Predatory prey task where n predators are trying to catch a prey. Red circles are predators, green circle is fixed prey and blue region shows vision range of an agent. Agents are initialized at random positions.(Left) Version where 5 predators try to catch a prey in 10×10 grid with vision 1. Agents can see any predators and prey in a square of side 3. (Right) IC3Net converges faster than CommNet as the number of predators (agents) increase in Predator-Prey environment.

We create three cooperation settings for this task with different reward structure to test our architecture's working in competitive, mixed and cooperative scenarios. In all three settings, a predator agent gets a constant time-step penalty $r_{explore} = -0.05$, until it reaches the prey. This makes sure that agent doesn't slack in finding the prey. In mixed setting, once on top of prey, agent always gets a positive reward r_{prey} = 0.05 which doesn't depend on number of agents on prey. There is no loss or benefit from communicating in this scenario. Similarly, in cooperative setting, agent gets a positive reward of $r_{coop} = r_{prey} * n$, and in competitive setting, agent gets a positive reward of $r_{comp} = r_{prey}$ / n after it reaches on prey where n is number of agents on a prey. Thus, in competitive setting, you get lower rewards if other agents reach the prey and in cooperative setting, reward increases when more agents are on the prey. The total reward at time t for an agent i can be written as:

$$r_i^{pp}(t) = \delta_i * r_{explore} + (1 - \delta_i) * n_t^{\lambda} * r_{prey} * |\lambda|$$

where δ_i is whether agent i has found the prey or not, n_t is number of agents on prey at time-step t and λ is -1, 0 and 1 in competitive, mixed and cooperative scenarios respectively. We show results on mixed cooperation settings in Table 1 (left) and test communication action's working in subsection 4.5 to support our claim that our architecture works in scenarios with different cooperative settings.

We create this environment in three different sizes to compare our network's performance as the number of agents increase with grid size. 10×10 grid version with 5 agents is shown in Figure 2 (left). Maximum episode steps are set to 20, 40 and 80 for 5×5, 10×10 and 20×20 grid respectively. Predator agents and prey are initialized randomly anywhere in the grid at the start of an episode. Each agent can take one of the five basic movement actions i.e. $up, down, left, right$ or $stay$. Predator, prey and all locations on grid are considered unique classes in vocabulary and are represented as one-hot binary vectors. Observation obs, at each point will be the sum of all one-hot binary vectors of location, predators and prey present at that point. With vision of 3, observation of each agent have dimension $3^2 \times |obs|$.

Results: Table 1 (left) shows average steps taken by the models to complete an episode i.e. find the prey in mixed setting. IC3Net reaches prey faster than the baselines as we increase the number of agents as well as the size of the maze. In 20×20 version, average steps gap is almost close to 36 steps which is a huge improvement over baselines. Figure 2 (right) shows the scalability graph for IC3Net and CommNet which supports the claim that with increasing number of agents, IC3Net converges faster and at better optimum than CommNet. Through these results on Predator-Prey task, we can realize that CommNet doesn't work well in mixed scenario, suggesting superiority of IC3Net in mixed scenarios. Finally, Figure 4 shows training plot of 20×20 grid with 10 agents trying to find a prey. The plot clearly depicts the fast rise in performance for IC3Net in contrast to CommNet which takes long time to achieve a minor jump.

4.3 Traffic junction

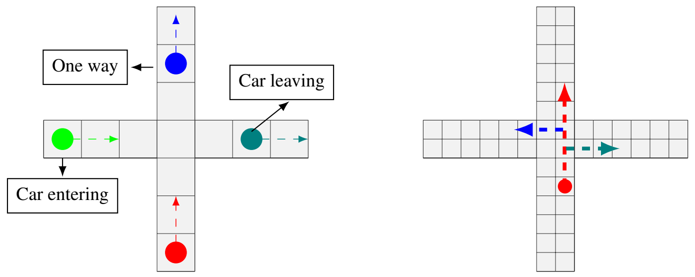


Figure 3: Traffic Junction task: Cars have to cross the the whole path without any collisions with two actions, gas and $brake$. Agents have zero vision and can only observe their own location. (Left) Easy version of traffic junction where there are only entry points and two one-way roads. (Right) Medium version of traffic junction task where there are four entry points and at each entry point you have 3 possible routes increasing chances of collision.

Table 1: (Left) Predator Prey: Average number of steps taken to complete the episode in three different environment sizes in mixed settings. IC3Net completes the episode faster than the baselines by finding the prey. (Right) Traffic Junction: Success rate on various difficulty levels with zero vision in each of them. IC3Net provides better performance than baselines consistently especially as the scale increases.

	Predatory-Prey (Avg. Steps)		
Model	**5x5, n=3**	**10x10, n=5**	**20x20, n=10**
Independent	16.5± 0.1	28.1± 0.2	75.0± 1.4
CommNet	9.1± 0.1	13.1± 0.01	76.5± 1.3
IC3Net	**8.9**± **0.02**	**13.0**± **0.02**	**40.5**± **3.4**

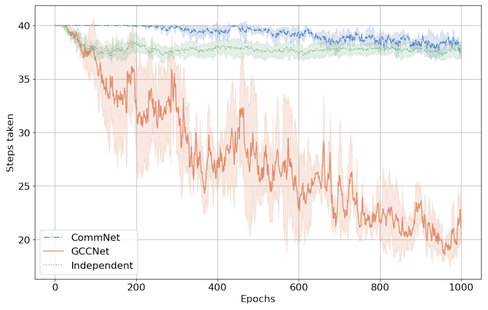
	Traffic Junction (Success %)		
Model	**Easy**	**Medium**	**Hard**
Independent	29.8± 0.7	3.4± 0.5	35.0± 0.6
CommNet	93.0± 4.2	54.3± 14.2	41.2± 13.5
IC3Net	**93.0**± **3.7**	**89.3**± **2.5**	**72.4**± **9.6**

Following Sukhbaatar et al. [23], we test our architecture on traffic junction task as it is a good proxy for testing whether communication is working. This task will also help in supporting our claim that IC3Net provides good performance and faster convergence in fully-cooperative scenarios similar to mixed ones. In traffic junction, cars enter a junction from any entry point with a probability p_{arr}. The numbers of maximum cars at a time in the junction is limited. Cars can take two actions at each time-step, gas and $brake$ respectively. Task has three difficulty levels (see Figure 3) which vary in number of possible routes, entry points and junctions. We make this task harder by always setting vision to zero in all three difficulty levels. This makes sure that task is not solvable without communication. We set maximum number of steps to 20, 40 and 60 in easy, medium and hard difficulty respectively.

Traffic junction's observation vocabulary has one-hot vectors for all of the locations in the grid and car class. Each agent observes its previous action, route identifier and a vector specifying sum of one-hot vectors of all the classes present at that agent's location. Collision occurs when two cars are on same location. Similar to Sukhbaatar et al. [23], we provide a negative reward r_{coll} = -10 on collision. To cut off traffic jams, we provide a negative reward $\tau_i r_{time}$ = -0.01 τ_i where τ_i is time spent by the agent in the junction at time-step t. Reward for ith agent which is having C_i^t collisions at time-step t can be written as:

$$r_i^{tj}(t) = r_{coll} C_i^t + r_{time} \tau_i$$

Results: Table 1 (right) shows success ratio for Traffic Junction task. Due to zero vision, it is not possible to perform well without communication as evident by results of Independent Controller. In Sukhbaatar et al. [23], well-performing agents in medium and hard versions of CommNet have vision > 0. With zero vision, IC3Net is superior to CommNet and Independent controller with a performance gap greater than 30%. This provides evidence to the fact that individualized rewards in IC3Net help achieve better or similar performance to CommNet in fully-cooperative tasks with communication due to better credit assignment. Interestingly, independent controller has a significant jump in success ratio for hard version which can be attributed to low add-rate in it.



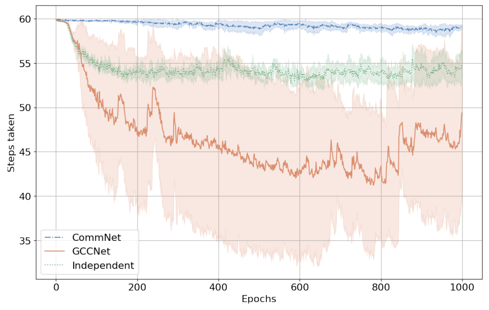


Figure 4: Average steps taken to complete the episode for Predator Prey (left) and StarCraft 10 medics explore (right). Performance of IC3Net is superior to the baselines in both cases.

4.4 StarCraft micro-management scenarios

To fully understand the scalability of our architecture in real-life and bigger scenarios, we test it on StarCraft combat and exploration tasks in partially observable settings. In StarCraft combat, two teams compete with each other on a big map where a team must kill all units of other team to win. StarCraft is a challenging RL task which has large observation-action space, a lot of different units and stochasticity as in real-life. Task can be made harder or easier by increasing or decreasing our units or enemy units or map size.

By default, game has macro-actions which allow a player to select a target unit to attack and then player's unit finds the best possible path using game's in-built path-finding system, move towards the target and attacks when the target unit is in range. We make the task harder by (i) removing macro-actions making the exploration harder (ii) making the environment partially observable by allowing units to only see what is in their limited vision and (iii) unlike previous works, initializing enemy units and our units at a random location in a fixed size square on map which introduces the component of exploration into the picture. We tackle two types of variations in StarCraft:

Explore: In this mode, we have n agents trying to explore the map and find an enemy unit. This is a direct scale-up of predator prey with more realistic and stochastic situations. To complete the task, agents must be within a particular range of enemy unit which we call *explore vision*. Once agent is within *explore vision* range of enemy unit, we $noop$ further actions. Observation for each agent is *(absolute x, absolute y)* and enemy's *(relative x, relative y, visible)* where *visible*, *relative x* and *relative y* are 0 when enemy is not in *explore vision*. Agents have 9 actions to choose from which includes 8 basic directions and one stay action. Reward structure is same as predator-prey task with only difference being that agent needs to be within *explore vision* range of the enemy unit instead of being on top of it to get non-negative reward. We use *medic* unit which doesn't attack enemy units to simulate our explore mode without any of kind of combat happening.

Combat: We test an agent's capabilities to execute complex task of combat in StarCraft which require coordination between the team, exploration of terrain, understanding of enemy units and formalism of human-like strategies. We specifically test a team of n agents trying to find and kill a team of m enemies in a partially observable environment similar to explore mode. These agents must find the team of enemy units and kill all of them first to score a win. Agent has to choose from 9 + m actions which include 9 basic actions and 1 action for attacking each of the m agents. Attack actions only work when the enemy is within the $sight$ range of the agent, otherwise it is a $noop$. Agent observes its own *(absolute x, absolute y, healthpoints + shield, weapon cooldown, previous action)* and *(relative x, relative y, visible, healthpoints + shield, weapon cooldown)* for each of the enemies. *relative x* and *relative y* are only observed when enemy is visible which is corresponded by *visible* flag. All of the observations are normalized to be in between (0, 1).

To avoid slack in finding the enemy team, we provide a negative reward r_{time} = -0.01 at each timestep when the agent is not involved in a combat. At each timestep, an agent gets as reward the difference between (i) its normalized health in current and previous timestep (ii) normalized health at previous timestep and current timestep for each of the enemies it has attacked till now. At the end of the episode, terminal reward for each agents consists of (i) all its remaining health * 3 as negative reward (ii) 5 * m + all its remaining health * 3 as positive reward if agents win (iii) normalized remaining health * 3 for all of the alive enemies as negative reward on lose. In Combat task, the group of enemies

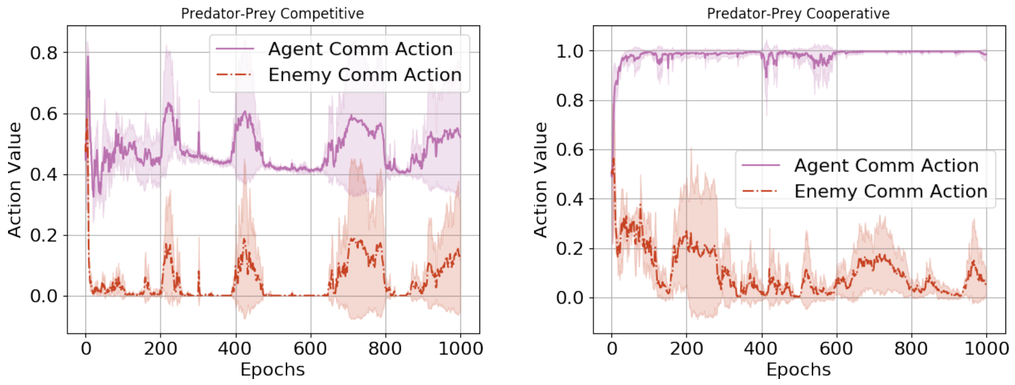


Figure 5: Communication Action testing: Plots show communication action for agents and the enemy averaged over each epoch (Left) Competitive setting for Predator-Prey task, agents only communicate until they reach the prey. Afterwards they don't communicate (Right) Cooperative setting for Predator-Prey task, agent almost always communicate to increase their own reward. In both cases, prey almost never communicates its position.

is initialized together randomly in one half of the map and our agents are initialized separately in other half which makes task even harder, thus requiring communication.

Table 2: StarCraft: Win Ratio and average number of steps taken to complete episodes for explore and combat tasks. IC3Net beats the baselines with huge margin in case of exploration task where 10 medics exploring for enemy medic tasks, while it is as good as CommNet in case of 10 Marines vs 3 Zealots combat task.

	Independent		**CommNet**		**IC3Net**	
StarCraft task	**Win %**	**Steps**	**Win %**	**Steps**	**Win %**	**Steps**
Explore-10Medic 50×50	35.4± 1.7	52.7± 0.6	9.2± 3.12	58.4± 0.5	**64.2± 17.7**	**41.2± 8.4**
Explore-10Medic 75×75	9.0± 4.2	58.6± 0.9	0.2± 0.3	59.9± 0.1	**17.0± 15.2**	**57.2± 3.5**
Combat-10Mv3Ze 50×50	74.6± 5.1	35.0± 0.8	88.0± 7.2	33.3± 1.2	87.4± 1.0	33.6± 0.2

Results: Table 2 displays win percentage and average number of steps taken to complete an episode in StarCraft explore and combat tasks. We specifically test on (i) Explore task: 10 medics finding 1 enemy medic on 50×50 cell grid (ii) Same explore task on 75×75 cell grid (iii) Combat task: 10 Marines vs 3 Zealots on 50 x 50 cell grid. Maximum steps in an episode are set to 60. Results on explore task are similar to Predator-Prey as IC3Net outperforms baselines. Moving to bigger map size, we still see the performance gap but performance drops as well for all of the models. This drop can be attributed to difficulty of dynamic and stochastic environment of StarCraft as the exploration with more number of agents becomes difficult on bigger map.

On combat, IC3Netperforms comparably well to CommNet as the global reward on winning is huge and shifts the weight toward task being fully-cooperative. We don't compare with prior work on StarCraft because our environment setting is much harder and restrictive, thus, not directly comparable. To confirm that 10 marines vs 3 Zealots is hard to win, we run an experiment on reverse scenario where our agents control 3 Zealots initialized separately and enemies are 10 marines initialized together. We find that both Independent controller and IC3Net reach a success percentage of 100% easily. We find that even in this case, IC3Net converges faster than Independent Controller.

4.5 Communication Action Tests

We test working of communication action in IC3Net by using cooperation and competitive settings in Predator-Prey and StarCraft explore tasks. We create only one communication group where an agent can decide to communicate with everyone or noone and test communication (i) among a team and (ii) between a team and a single enemy in these scenarios. . Enemy shares weights with agents and acts as an agents itself and all its actions is *noop*. Enemy gets a positive reward equivalent to r_{time} = 0.05 until noone is near or on it and afterwards it gets a reward of 0.

Figure 5 shows the agents' communication action as well as enemy's communication action averaged per epoch in competitive as well as cooperative scenarios. Agents and the enemy share the weights and we test on 5×5 map size setting on Predator-Prey task. In case of competitive setting, we observe that once an agent is near the prey it stops communicating, as more people near the prey, lesser the reward. During exploration, they seldom communicate to get to enemy as fast as possible which results in agents' communication average action value to be greater than 0. In cooperative scenario, agents almost always communicate to bring more agents near prey and increase there rewards. Since prey stops getting reward once any agent comes near it, it almost never communicates. This makes sure it doesn't share its location accidentally. We test with similar competitive setting in StarCraft 10 medics finding 1 enemy medic and found similar results with more deviation.

5 Conclusions and Future Work

In this work, we introduced IC3Net which aims to solve multi-agent tasks in various cooperation settings by learning to communicate as necessary. Its continuous communication enables efficient training by backpropagation, while the discrete gating trained by reinforcement learning allows it be used even in competitive scenarios. Through our experiments, we show that IC3Net performs well in cooperative, mixed or competitive settings through multiple environments including StarCraft: Broodwars. Further, we show that agents learn to stop communication in competitive cases. In future, we would like to explore possibility of having multi-channel communication where agents can decide on which channel they want to put their information similar to communication groups but dynamic. It would be interesting to provide agents a choice of whether to listen to communication from a channel or not.

References

[1] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. Openai gym. *arXiv preprint arXiv:1606.01540*, 2016.

[2] L. Bu¿oniu, R. Babu $hats$ka, and B. D. Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, March 2008. ISSN 1094-6977. doi: 10.1109/TSMCC.2007.913919.

[3] Yu-Han Chang, Tracey Ho, and Leslie Pack Kaelbling. All learning is local: Multi-agent learning in global reward games. In *Proceedings of the 16th International Conference on Neural Information Processing Systems*, NIPS'03, pages 807–814, Cambridge, MA, USA, 2003. MIT Press. URL `http://dl.acm.org/citation.cfm?id=2981345.2981446`.

[4] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2137–2145. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6042-learning-to-communicate-with-deep-multi-agent-reinforcement-learning.pdf`.

[5] Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate to solve riddles with deep distributed recurrent q-networks. *CoRR*, abs/1602.02672, 2016. URL `http://arxiv.org/abs/1602.02672`.

[6] Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *CoRR*, abs/1705.08926, 2017. URL `http://arxiv.org/abs/1705.08926`.

[7] Jayesh K. Gupta, Maxim Egorov, and Mykel J. Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *Adaptive Learning Agents Workshop*, 2017. doi: 10.1007/978-3-319-71682-4_5.

[8] 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*, pages 770–778, 2016.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[10] Martin Lauer and Martin Riedmiller. An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In *In Proceedings of the Seventeenth International Conference on Machine Learning*, pages 535–542. Morgan Kaufmann, 2000.

[11] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *CoRR*, abs/1612.07182, 2016. URL `http://arxiv.org/abs/1612.07182`.

[12] Joel Lehman, Jeff Clune, Dusan Misevic, Christoph Adami, Julie Beaulieu, Peter J. Bentley, Samuel Bernard, Guillaume Beslon, David M. Bryson, Patryk Chrabaszcz, Nick Cheney, Antoine Cully, Stéphane Doncieux, Fred C. Dyer, Kai Olav Ellefsen, Robert Feldt, Stephan Fischer, Stephanie Forrest, Antoine Frénoy, Christian Gagné, Leni K. Le Goff, Laura M. Grabowski, Babak Hodjat, Frank Hutter, Laurent Keller, Carole Knibbe, Peter Krcah, Richard E. Lenski, Hod Lipson, Robert MacCurdy, Carlos Maestre, Risto Miikkulainen, Sara Mitri, David E. Moriarty, Jean-Baptiste Mouret, Anh Nguyen, Charles Ofria, Marc Parizeau, David P. Parsons, Robert T. Pennock, William F. Punch, Thomas S. Ray, Marc Schoenauer, Eric Shulte, Karl Sims, Kenneth O. Stanley, François Taddei, Danesh Tarapore, Simon Thibault, Westley Weimer, Richard Watson, and Jason Yosinksi. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *CoRR*, abs/1803.03453, 2018. URL `http://arxiv.org/abs/1803.03453`.

[13] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6379–6390. Curran Associates, Inc., 2017. URL `http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf`.

[14] L. Matignon, G. J. Laurent, and N. L. Fort-Piat. Hysteretic q-learning :an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 64–69, Oct 2007. doi: 10.1109/IROS.2007.4399095.

[15] Laëtitia Matignon, Guillaume J. Laurent, and Nadine Le Fort-Piat. Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems. *Knowledge Engineering Review*, 27(1):1–31, March 2012. doi: 10.1017/S026988891200057. URL `https://hal.archives-ouvertes.fr/hal-00720669`.

[16] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015.

[17] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. *CoRR*, abs/1703.04908, 2017. URL `http://arxiv.org/abs/1703.04908`.

[18] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian. Deep decentralized multi-task multi-agent reinforcement learning under partial observability. *CoRR*, abs/1703.06182, 2017. URL `http://arxiv.org/abs/1703.06182`.

[19] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, November 2005. ISSN 1387-2532. doi: 10.1007/s10458-005-2631-2. URL `http://dx.doi.org/10.1007/s10458-005-2631-2`.

[20] Peng Peng, Quan Yuan, Ying Wen, Yaodong Yang, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *CoRR*, abs/1703.10069, 2017. URL `http://arxiv.org/abs/1703.10069`.

[21] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap,

Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529:484–489, 2016.

[22] Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems*, pages 2244–2252, 2016.

[23] Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 2244–2252. Curran Associates, Inc., 2016. URL `http://papers.nips.cc/paper/6398-learning-multiagent-communication-with-backpropagation.pdf`.

[24] Ming Tan. Readings in agents. chapter Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents, pages 487–494. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998. ISBN 1-55860-495-2. URL `http://dl.acm.org/citation.cfm?id=284860.284934`.

[25] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2): 26–31, 2012.

[26] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3-4):229–256, May 1992. ISSN 0885-6125. doi: 10.1007/BF00992696. URL `https://doi.org/10.1007/BF00992696`.