Deep Multi Agent Reinforcement Learning for Autonomous Driving

Sushrut Bhalla[1] [0000−0002−4398−5052],
Sriram Ganapathi Subramanian[1] [0000−0001−6507−3049], and
Mark Crowley[1] [0000−0003−3921−4762]

University of Waterloo, Waterloo ON N2L 3G1, Canada
{sushrut.bhalla,s2ganapa,mcrowley}@uwaterloo.ca

Abstract. Deep Learning and back-propagation have been successfully used to perform centralized training with communication protocols among multiple agents in a cooperative environment. In this work, we present techniques for centralized training of Multi-Agent Deep Reinforcement Learning (MARL) using the model-free Deep Q-Network (DQN) as the baseline model and communication between agents. We present two novel, scalable and centralized MARL training techniques (MA-MeSN, MA-BoN), which achieve faster convergence and higher cumulative reward in complex domains like autonomous driving simulators. Subsequently, we present a memory module to achieve a decentralized cooperative policy for execution and thus addressing the challenges of noise and communication bottlenecks in real-time communication channels. This work theoretically and empirically compares our centralized and decentralized training algorithms to current research in the field of MARL. We also present and release a new OpenAI-Gym environment which can be used for multi-agent research as it simulates multiple autonomous cars driving on a highway. We compare the performance of our centralized algorithms to existing state-of-the-art algorithms, DIAL and IMS based on cumulative reward achieved per episode. MA-MeSN and MA-BoN achieve a cumulative reward of at-least 263% of the reward achieved by the DIAL and IMS. We also present an ablation study of the scalability of MA-BoN showing that it has a linear time and space complexity compared to quadratic for DIAL in the number of agents.

Keywords: Multi-Agent Reinforcement Learning · Autonomous Driving · Emergent Communication.

1 Introduction

Multi Agent Reinforcement Learning (MARL) is the problem of learning optimal policies for multiple interacting agents using RL. Current autonomous driving research focuses on modeling the road environment consisting of only human drivers. However, with more autonomous vehicles on the road, a shared cooperative policy among multiple cars is a necessary scenario to prepare for.

To overcome the problem of non-stationarity in the training of MARL agents, the current literature proposes the use of centralized training using message sharing between the agents [12]. The message shared between the agents is generated using the policy network and trained using policy gradients. This approach leads to sub-optimal messages being shared between agents as the message is inadvertently tied to the policy of the agent [3]. Current approaches thus show a poor performance in large-scale environments with sparse rewards and a long time to horizon as shown in our experiments section.

In this paper, we propose centralized training (MA-MeSN) algorithms for MARL environments which are a generalization of the MARL algorithms currently in literature. Our approach allows separation of policy and communication models and provides a stabilized method for training in an off-policy method. We also compare our centralized training algorithm against DIAL (Differentiable Inter-Agent Learning) [5] and IMS (Iterative Message Sharing) [15] on a large scale multi-agent highway driving simulator we developed as part of this work. We present techniques (MA-MeSN-MM) to derive a cooperative decentralized policy from the trained centralized policy (MA-MeSN). All algorithms are compared based on various metrics our treadmill driving simulator and OpenAI's multi-agent particle environments [14] for formal verification of our algorithms.

2 Related Work

MARL has a rich literature (particularly in the robotics domain [2]). Independent cooperative tabular Q-learning with multiple agents has been studied in [16]. The empirical evaluation shows that cooperative behavior policy can only be achieved by information sharing, for example, other agents' private observations, policies or episode information. Hyper Q-learning [17] extends the idea of information sharing to policy parameter sharing between the agents. This change allows the agents to condition their policy on the parameters of the other agents and thus mitigate non-stationary in MARL environments.

There is a vast literature on the emergence of communication between agents in the same environment [9, 14, 15, 4]; which propose training messages shared between agents using backpropagation. The work in [15, 5] extends the techniques of message sharing between agents to multi-agent reinforcement learning (MARL). The authors in [15] employ a message sharing protocol where an aggregated message is generated, by averaging the messages from all agents, and passing it back as an input to the agents along with their observation's hidden state representation to compute the final state-action values. This Iterative Message Sharing (**IMS**) is iterated P times before the final action for all agents is computed using ϵ-greedy method. Differentiable Inter-Agent Learning **DIAL** [5] also trains communication channels, through back-propagation, for sequential multi-agent environments. DIAL presents an on-policy training algorithm which uses the past history to generate messages for inter-agent communication. In this paper, we present a generalization of the MARL algorithms currently available in literature for centralized training. Our algorithm is able to outperform DIAL and IMS on large scale environments while achieving a better time and space complexity during training and execution.

Multi agent environments require a decentralized execution of policy by agents in the environment. Work in [11, 14, 7] has shown that the MARL agents could be evaluated with discrete communication channels by using a softmax operation on the message. This approach provides a partial decentralization of the trained centralized policy. The authors in [6] successfully train multiple independent agents by stabalizing the experience replay for multi-agent setting. The stabilization is done by prioritizing newer experiences in the experience buffer for training as they represent the current transition dynamics of the environment. We also compare the training of our decentralized policy against the independent agents trained using Stabilized Experience Replay (**SER**). Our algorithm allows the centralized trained cooperative policy to be easily extended to a decentralized setting while maintaining acceptable performance.

3 Background on Multi-Agent Reinforcement Learning

In this section we present a background on multi-agent reinforcement learning and the variables used in the paper. A short background on Deep Q-Networks [13] can be found in Appendix Sec. A.1.

In this work we consider a general sum multi-agent stochastic game G which is modeled by the tuple $G = (X, S, A, T, R, Z, O)$ with N agents, $x \in X$, in the game. The game environment presents states $s \in S$, and the agents observe an observation $z \in Z$. The observation is generated using the function $Z \equiv O(s, x)$ which maps the state of each agent to its private observation z. The game environment is modeled by the joint transition function $T(s, \mathbf{a_i}, s')$ where $\mathbf{a_i}$ represents the vector of actions for all agents $x \in X$. The dependence of the transition matrix on behavior policy of other agents gives it the non-stationary property in multi-agent environments. We use the subscript notation i to represent the properties of a single agent x, a bold subscript $\mathbf{i}$ to represent properties of all agents $x \in X$ and $-\mathbf{i}$ to represent the properties of all agents other than x_i. We use the superscript t to represent the discrete time-step. All agents share the same utility function R, which provides agents with an instantaneous reward for an action a_i. Our game environment represents a Decentralized Partially Observable Markov Decision Process (DEC-POMDP) [1]. The agents can send and receive discrete messages between each other, which are modeled based on speech act theory, represented as m_i^t. The game environment does not provide a utility function in response to the communication/message actions performed by an agent. The major challenges in the domain of multi-agent reinforcement learning include the problem of dimensionality, coordinated training, and training ambiguity. Having strong communication between agents can solve some of these problems.

4 Methods

4.1 Multi-Agent Message Sharing Network (MA-MeSN)

The DIAL and IMS methods demonstrated that emergent communication between multiple agents can be achieved by optimizing messages shared between agents using backpropagation. DIAL presents a model where the communicative actions (generated by the message policy) and non-communicative actions (generated by the behavior policy) are generated using the same model. This approach forces a strong correlation between the communicative and non-communicative actions, but leads to sub-par results. Behavior policy of the agents might be similar in a cooperative environment, but their message policy is focused on achieving high information sharing between agents. Using the same model to predict the behavior and message policy would lead to conflicting updates to the neural network due to different objectives.

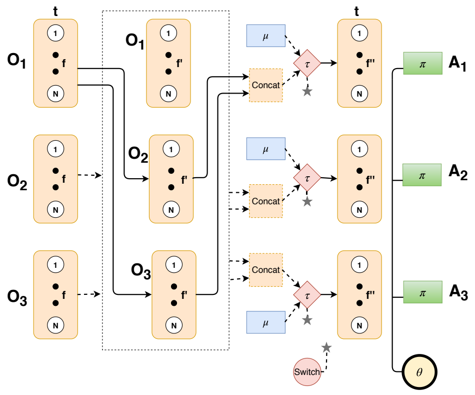


Fig. 1: Architecture for Multi-Agent Message Sharing Network (MA-MeSN)

We thus present a generalization of the communication based MARL algorithms in Fig. 1 where each agent uses a different model for message policy and behavior policy. The f'' neural network maps the message received from the other agents $m_{\mathbf{-i}}$ along with its partial observation of the environment z_i to a state-action-message value function $f'' = Q(z_i, a_i, m_i)$. We refer to this network as the (behavior) policy network. The message is generated by the other agents $x_{\mathbf{-i}}$ using the neural network approximator f'_i which maps the agent's private observation to a communication action $m_{\mathbf{-i}}$. We refer to this network as the message (policy) network. The message passing interaction/negotiation can be extended to multiple iterations for faster convergence. During the training, we allow only allow a single pass of messages between agents. In contrast to previous work in DIAL, we train the message network using the cumulative gradients of all policy networks as shown in Alg. 1. Optimizing the message network with cumulative gradients leads to messages which are generalizable to all agent policies.

Algorithm 1 Multi-Agent Message Sharing Network (MA-MeSN)

```
for $i = 1, N$ do
    Initialize replay memory $\mathcal{D}_\rangle; where i \in \{1..N\}$ to capacity $M$
    Initialize the online and target, message and policy networks $f'_{i,\theta}, f''_{i,\theta}, f'_{i,\theta'}, f''_{i,\theta'}$
end for
for $episode = 1, E$ do
    for $t = 1, T_{convergence}$ do
        for $i = 1, N$ do
            Select a random action $a_i^t$ with probability $\varepsilon$
            Otherwise, select $a_i^t = \arg\max_a Q_{f_i''}(o_i^t, m_{-\mathbf{i}}^t, a; f_\theta'')$
            Execute action $a_i^t$, collect reward $r_i^{t+1}$ and observe next state $o_i^{t+1}$
            Store the transition $(o_i^t, a_i^t, r_i^{t+1}, o_i^{t+1})$ in $\mathcal{D}_\rangle$
            Sample mini-batch of transitions $(o_i^j, a_i^j, r_i^{j+1}, o_i^{j+1})$ from $\mathcal{D}_\rangle$
            Generate the messages from other agents $m_{-\mathbf{i}}^j = f'_{-\mathbf{i}}(o_{-\mathbf{i}}^j)$
            Set $y_i^j = \begin{cases} r_i^{j+1}, & \text{if } o_i^{j+1} \text{ is terminal} \\ r_i^{j+1} + \gamma \max_{a'} Q_{f_i''}(o_i^{j+1}, m_{-\mathbf{i}}^{j+1}, a'; f''_{i,\theta'}), & \text{otherwise} \end{cases}$
            Compute gradients using target value $y_i^j$ for policy network $f_\theta''$
            $\Delta Q_{f_i''} = y_i^j - Q_{f_i''}(o_i^j, m_{-\mathbf{i}}, a; f''_{i,\theta}$
            Apply gradients $\nabla\theta_{i,f''}$ to $f''_{i,\theta}$
            Collect gradients $\nabla\theta_{\mathbf{i},f'}$ from all policy networks
            Apply gradients $\nabla\theta_{\mathbf{i},f'}$ to $f'_{\mathbf{i}}$
        end for
        Every $C$ steps, set $\theta'_{i,f''} \leftarrow \theta_{i,f''} \forall i$
        Every $C$ steps, set $\theta'_{i,f'} \leftarrow \theta_{i,f'} \forall i$
    end for
end for
```

Comparison to Previous Work This approach has two advantages over DIAL. The messages $m_{-\mathbf{i}}^t(z_{-\mathbf{i}}^t, f(z_i^t))$ are conditioned on the entire observable state at time t, as opposed to DIAL, where messages $m_{-\mathbf{i}}^t(z_{-\mathbf{i}}^{t-1})$ are a function of the previous time-step observation of each agent $z_{-\mathbf{i}}^{t-1}$. Generating a message based on the past introduces the message network's dependency on the transition dynamics; which as discussed exhibits a non-stationary property in multi-agent environments and thus lead to divergence. On the other hand, in MA-MeSN, training the message network to generate messages $m_{-\mathbf{i}}^t$ based on the current observation reduces the dependence on the environment's transition dynamics. Secondly, this allows for our algorithm to train off-policy using a step based experience replay. Whereas DIAL requires on-policy training using recorded trajectories.

Fully Decentralized Cooperative Policy The messages shared between agents are discrete of size 2 bytes. We generate discrete message by applying Gumbel-Softmax Sampling [7] on the prediction of the message network. To achieve fully decentralized execution without message sharing, we utilize a LSTM memory module μ in conjunction with each agent's policy network. The

$LSTM_\mu$ learns a mapping from agent's private observation history to the message generated by the other agents in the environment. The model $LSTM_\mu$ mimics the message received from other agents. Thus the individual memory modules μ along with their policy network f'' can be independently used for fully decentralized execution of the learned cooperative policy (MA-MeSN-MM). The message memory module $LSTM_\mu$ is trained in a supervised fashion in parallel to the policy and message networks during centralized training.

4.2 Multi-Agent Broadcast Network (MA-BoN)

The generalization of communication based centralized MARL algorithms presented in the previous section allows us to develop communication models with distinct message types. We constraint our MA-MeSN model to a single message to rule them all approach and develop a broadcast model as shown in Fig. 2. The neural network f' (message network) maps the shared partial observation encoding from all agents to a broadcast message bm^t. We study the properties of MA-MeSN and MA-BoN in Sec. 6.3 and show that this network is feasible in multi-agent general sum games.

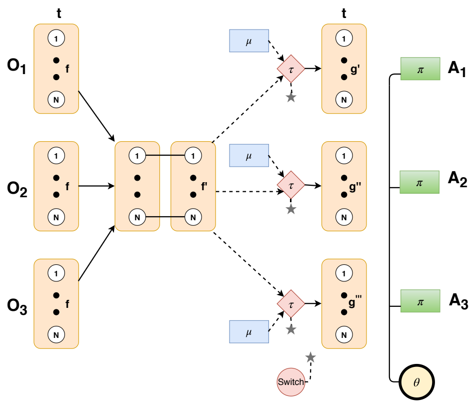


Fig. 2: Multi-Agent Broadcast Network (MA-BoN)

The NN f' learns a combined communication message as the broadcast message (bm^t). Each agent can now independently evaluate the action-value for their private observation using the function $g'(z_i^t, bm^t)$, which is a function of the complete observed state of the environment. This network also allows for parallel action-value evaluations with a single forward pass of the network and avoids the $|P|$ iterations required by IMS, and provides a linear space and runtime complexity as shown in Sec. 6.2. MA-BoN can also be decentralized by the use of a memory module $LSTM_\pi$ trained parallel to the policy network (MA-BoN-MM).

5 Experimental Methodology

In this paper, we compare our algorithms with MARL algorithms in the literature on three different MARL environments. We present the treadmill driving environment simulator in this section. The OpenAI particle environments are used to show the validity of our algorithms on public testbeds. The results can be found in Appendix Sec. B.

5.1 Treadmill Driving Environment

The treadmill environment simulates an infinite highway with multiple cars driving in the presence of an adversary. The highway is simulated using a treadmill, which is always running and thus creates an infinite highway. The size of the treadmill is currently kept fixed at $[100, 100]$ steps. Agents can enter or exit the treadmill from the front and back. The treadmill contains a minimum of 2 cooperative autonomous agents and at least 1 adversary agent. These agents can be controlled using Deep RL methods and the adversary (aggressive) car is controlled with a stochastic behavior policy which can cause a crash with the closest autonomous car. The cooperative autonomous vehicles can sense the closest car as part of its partial private observation of the environment, but do not receive information to distinguish between their behavior (cooperative/adversary). The agents can send messages to other agents using a discrete communication broadcast channel, to which other agents subscribe. The private reward received by an autonomous car is the normalized distance from the closest observed car and a large negative reward for a crash. The agents' actions include 3 angles of steering in 8 directions and 3 discrete levels of acceleration/deceleration. The reward function does not provide explicit rewards for cooperation between the agents or for maintaining stable emergent communications between agents. The episode is terminated when the distance between any two agents is 0 (collision is encountered). Each experiment takes $8-12$ hours of training on a GTX-960 GPU. The exploration in all experiments follows a linear decay schedule for the first 20% or $100K$ steps of the experiment (whichever is achieved earlier) and then is set to a constant value of 0.05.

6 Results and Discussions

In this section, we present the results of training our algorithms in the treadmill driving environment. In all our algorithms (MA-MeSN, MA-BoN, DIAL, IMS, independent DQN, independent DQN with SER), we use a hierarchical neural network structure [8]. We provide an evaluation of hierarchical DQN on treadmill driving environment domain in Appendix A. In this section, we focus on presenting performance results for our multi-agent algorithms on the treadmill driving simulator.

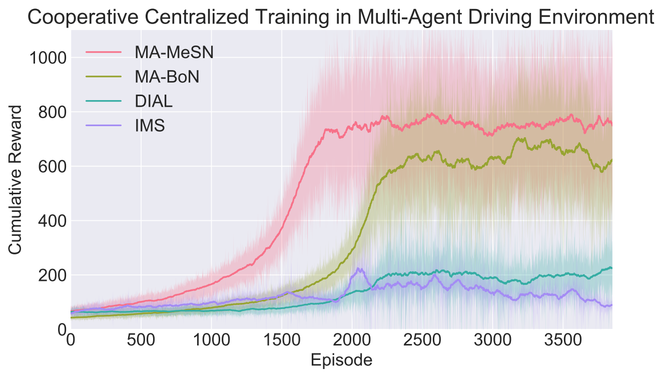


Fig. 3: Comparison of Cumulative Reward for Centralized Training Algorithms in Multi-Agent Driving Environment.

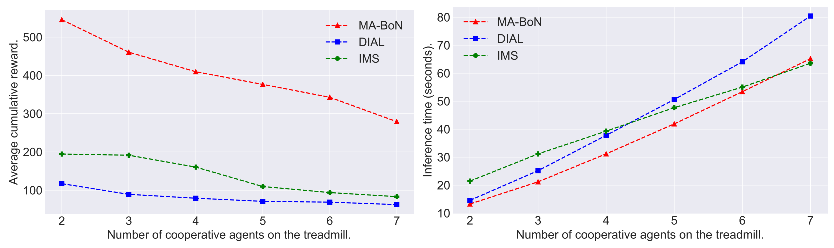
6.1 Centralized Training on Multi-Agent Driving Environment

All experiments are run for a minimum of $4K$ episodes ($0.8M$ steps). All neural networks consist of two layers with 4096 neural units in the first layer with 12 neurons in the second layer. DIAL network consists of two layers with 6144 units in the first layer to allow for fair evaluation to other algorithms. The maximum size of message shared between agents is 2 bytes. We use Adam optimizer with a learning rate of 5×10^{-4}. The batch-size for updates is 64 and the target network is updated after 200 steps, except DIAL's target network is updated after 40 episodes. For the IMS algorithm, we arrived at using $P = 5$ for communication iterations through cross-validation.

The cumulative reward achieved during centralized training of our MARL algorithms is shown in Fig. 3. All experiments are repeated 20 times and averaged to produce the learning curves. We achieve the highest cumulative reward with MA-MeSN followed by the MA-BoN algorithm. The IMS and DIAL algorithms are able to improve on the policy achieved by independent DQN, as they have the advantage of message sharing over independent DQN policy. IMS shows a slow learning curve compared to other algorithms with $P = 5$ communication iterations. IMS training also requires curriculum learning approach to train the network efficiently [15]. However, to maintain fairness, this was left out in our experiments. DIAL shows steady improvement in performance, however, the performance of the final policy is weak when compared to MA-MeSN.

As results show, our generalized MARL algorithm (MA-MeSN) is able to perform superior to DIAL and IMS. The benefit of having a separate model for message policy and behavior policy prediction. The separation of message policy model and behavior policy model leads to each neural network achieving a more optimal solution than competing approaches. Whereas, DIAL and IMS constraint the training of message and behavior policy to a single neural network which produces sub-optimal results. MA-BoN also constraints the inter-agent message sharing to a single broadcast message which also leads to a sub-par result in comparison to MA-MeSN.

6.2 Ablation Study of scalability of MA-BoN



(a) Avg. cumulative reward achieved at convergence with varying number of agents in the environment.

(b) Avg. inference time of the algorithms with varying number of agents in the environment.

Fig. 4: Scalability comparison on the treadmill environment.

In this section, we demonstrate the scalability of the MA-BoN approach compared to DQN with stabilized experience replay, IMS and DIAL. We carry out an ablation study of our approach by varying the number of cars in the environment and present the results in Fig 4. The Fig 4 shows a comparison of the inference time it took to complete an episode and the average cumulative reward achieved per episode when the number of agents in the environment is increased. The results for cumulative reward comparison are computed by averaging results of 5 training runs for each algorithm with different seed values. The training of all algorithms was completed over $15,000$ episodes or $2.5M$ steps. We see that our approach MA-BoN is able to sustain better performance compared to other approaches when the complexity of the environment was increased. The inference time grows linearly for MA-BoN in comparison to the quadratic increase for DIAL. MA-BoN shows better scalability as the message generation network for each agent is optimized separately using the cumulative gradients from all agent's temporal difference loss. Thus the message is more generalizable in complex settings, while DIAL and IMS suffer from the problem of optimizing the joint objective for communicative and non-communicative policy; which leads to reduced robustness of the messages shared between agents.

6.3 Theoretical study of Emergent Communication

In this section, we study the inter-agent emergent communication achieved during training of our MARL algorithm, MA-MeSN. Table 1 shows the results for MA-MeSN using common metrics [10] to measure the effect of these messages using our domain. *Speaker consistency (SC)* is used to measure positive signaling as it measures the mutual information between the communicative m_i and behavior a_i policy of an agent. We see a small positive value of 0.18 for SC; which suggests that the objective for message policy and behavior policy are indeed different. Thus our approach of generalizing MARL algorithms with separate message and policy networks is necessary. *Instantaneous Coordination (IC)* measures the positive listening between agents, which is measure of the mutual information between the speaker's communicative actions $m_{-\mathbf{i}}$ and the listener's behavior/locomotive actions a_i. We achieve a value of 0.41 for IC which indicates that the listener agent's policy are dependent on the messages of the speaker agent, which is necessary for emergent communication. We also study the *Communication Message Entropy* which measures if the listener receives the same message for a given input. We achieve a value of 1.27 for entropy, which shows that the speaker is not using different messages **for the same input** and is rather consistent in its signals.

Table 1: Study of Emergent Communication in MA-MeSN. The table shows the results for speaker consistency, instantaneous coordination and entropy [10].

Emergent Communication Metric Used	Value
Speaker Consistency (Positive signaling)	0.18
Instantaneous Coordination (Positive Listening)	0.41
Communication Message Entropy	1.27
Message Input Norm (MIN)	63.75
Cumulative reward with white noise	319.07

To further study the effects of communication, we probe our MA-MeSN model, calculate the *L2-norm* of the fully connected weight matrix for message input for the listener agent, and report the results in Table 1. The weight matrix for the message input has an *L2-norm* much higher than 0.0, which suggests that the message indeed does get used by the listener agent's policy network. We extend our analysis of the MA-MeSN by replacing the messages received by the agents with white noise on a trained MA-MeSN model. We see a reduction in the mean cumulative reward achieved by the algorithm from 746.8 to 319.07 in the stochastic environment. The reduction in the cumulative reward shows that emergent communication did develop between agents and is an integral part of the final cooperative policy achieved.

6.4 Fully Decentralized Cooperative Policy in Driving Environment

We compare our method of using message memory models for decentralized execution (MA-MeSN-MM) with independent DQN, DQN with stabilized expe-

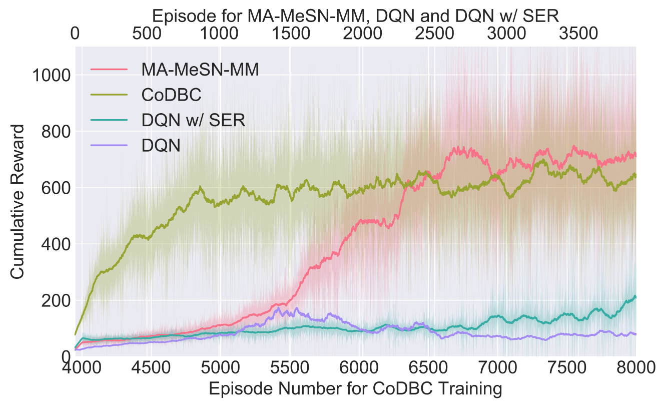


Fig. 5: Comparison of Cumulative Reward for Decentralized Training in Multi-Agent Driving Environment.

rience replay and distributed behavior cloning of centralized cooperative policy (CoDBC). CoDBC policy is trained using imitation learning of the (expert) centralized cooperative policy from MA-MeSN. All of the hyper-parameters and experimental setup are exactly the same as the experiments for the centralized training section. The learning curve for decentralized policies is shown in Fig. 5. As the treadmill environment does not explicitly reward agents for cooperation, we see poor performance from DQN and DQN with SER; however DQN with SER is more stable during training compared to DQN. DQN with SER applies a weight to each training sample's gradient. The weight is computed using a linearly decaying function based on the episodes elapsed since a sample was collected. Thus, DQN with SER is able to prioritize its training on the latest samples (which represent the latest policies of other agents) collected in the DQN's buffer and thus avoids divergence. However, the final policy achieved by DQN w/ SER is worse than MA-MeSN-MM and CoDBC.

While the CoDBC method outperforms DQN and DQN w/ SER, the number of episodes required to learn a cooperative policy is nearly 8000 episodes, as CoDBC needs to be run sequentially after MA-MeSN policy training has converged. Our method MA-MeSN-MM achieves decentralized cooperative policy by learning a function mapping from private observations to the messages received from other agents. The message module (MM) is trained in parallel to the policy network and thus does not require additional training after MA-MeSN has converged. This approach is ideal for real-time agents in MARL environments with a goal of cooperation as communication channels are unreliable and induce a time-latency.

7 Conclusion and Future Work

In this paper we present that generalization of the current work in MARL field leads to large improvements in the final multi-agent policy. Our approach allows for variability in the message format which is useful for various domains. MA-MeSN and MA-BoN both outperform the algorithms found in current literature based on learning curve results. Our algorithms also provide improvements in the time and space complexity over DIAL and IMS. MA-MeSN and MA-BoN are easier to train as they can be trained in an off-policy setting. We also present a decentralized model which achieves higher cumulative reward compared to some of the centralized techniques and all decentralized techniques. This paper also presents a new large scale multi-agent testing environment for further MARL research.

References

1. Bernstein, D.S., Givan, R., Immerman, N., Zilberstein, S.: The complexity of decentralized control of markov decision processes. Mathematics of operations research **27**(4), 819–840 (2002)
2. Busoniu, L., Babuska, R., De Schutter, B.: A comprehensive survey of multiagent reinforcement learning. IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008 (2008)
3. Das, A., Kottur, S., Moura, J.M.F., Lee, S., Batra, D.: Learning cooperative visual dialog agents with deep reinforcement learning pp. 2970–2979 (Oct 2017). https://doi.org/10.1109/ICCV.2017.321
4. Das, A., Kottur, S., Moura, J.M., Lee, S., Batra, D.: Learning cooperative visual dialog agents with deep reinforcement learning. arXiv preprint arXiv:1703.06585 (2017)
5. Foerster, J., Assael, I.A., de Freitas, N., Whiteson, S.: Learning to communicate with deep multi-agent reinforcement learning. In: Advances in Neural Information Processing Systems. pp. 2137–2145 (2016)
6. Foerster, J., Nardelli, N., Farquhar, G., Torr, P., Kohli, P., Whiteson, S.: Stabilising experience replay for deep multi-agent reinforcement learning. In: ICML 2017: Proceedings of the Thirty-Fourth International Conference on Machine Learning (June 2017), http://www.cs.ox.ac.uk/people/shimon.whiteson/pubs/foerstericml17.pdf
7. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144 (2016)
8. Kulkarni, T.D., Narasimhan, K., Saeedi, A., Tenenbaum, J.: Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In: Advances in neural information processing systems. pp. 3675–3683 (2016)
9. Lazaridou, A., Peysakhovich, A., Baroni, M.: Multi-agent cooperation and the emergence of (natural) language. arXiv preprint arXiv:1612.07182 (2016)
10. Lowe, R., Foerster, J., Boureau, Y.L., Pineau, J., Dauphin, Y.: On the pitfalls of measuring emergent communication. arXiv preprint arXiv:1903.05168 (2019)
11. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O.P., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments. In: Advances in Neural Information Processing Systems. pp. 6379–6390 (2017)

12. Lowe, R., WU, Y., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-agent actor-critic for mixed cooperative-competitive environments pp. 6379–6390 (2017), http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf
13. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., et al.: Human-level control through deep reinforcement learning. Nature **518**(7540), 529 (2015)
14. Mordatch, I., Abbeel, P.: Emergence of grounded compositional language in multi-agent populations. arXiv preprint arXiv:1703.04908 (2017)
15. Sukhbaatar, S., Szlam, A., Fergus, R.: Learning multiagent communication with backpropagation. In: Lee, D.D., Sugiyama, M., Luxburg, U.V., Guyon, I., Garnett, R. (eds.) Advances in Neural Information Processing Systems 29, pp. 2244–2252. Curran Associates, Inc. (2016), http://papers.nips.cc/paper/6398-learning-multiagent-communication-with-backpropagation.pdf
16. Tan, M.: Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning. pp. 330–337 (1993)
17. Tesauro, G.: Extending q-learning to general adaptive multi-agent systems. In: Advances in neural information processing systems. pp. 871–878 (2004)
18. Van Hasselt, H., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: AAAI. vol. 2, p. 5. Phoenix, AZ (2016)

A DQN and Hierarchical DQN.

A.1 Deep Q-Network (DQN)

The Deep Q-Network (**DQN**) is an off policy algorithm which uses neural networks to approximate the action-value function $Q(z, a)$ [13]. z denotes the observation of the environment, a is the action, and t denotes the discrete time-step. The value of the current state can be computed using the formula, $Q(z^t, a^t) = \mathbb{E}_a[r(z^t, a^t) + \gamma\,\mathbb{E}_a[Q^T(z^{t+1}, a^{t+1}; \theta')]]$. Q^T is generated using the target network which computes the estimates for next state-action values and is periodically updated to the newest online network $(Q(z^t, a^t))$. The update rule minimizes the following error:

$$\text{TD-Error}(w) = [Q(z^t, a; \theta) - r - \gamma Q^T(z^{t+1}, a^{t+1}; \theta^T)]^2$$

where θ and θ^T represents the weights of the online and target Q-network respectively. The DQN uses an experience replay buffer to generate batches of trajectories to use for updates of the Q-network; which makes it an off-policy algorithm. Gradient Descent is used to learn the parameters θ by iteratively minimizing the loss TD-Error. The experience replay memory (ER) and a separate target network is used to stabilize the training. The target network weights are updated C iterations to the latest weights in the evaluation network.

A.2 Hierarchical DQN

Hierarchical DQN [8] extends DQN by using prior knowledge of the environment. It is a framework that integrates hierarchical value functions operating at different temporal scales. The agent's learning goals are split into 2 levels of hierarchy: top-level (meta-controller) to generate high-level actions/goals and bottom-level (controller) to achieve the high-level goals. The meta-controller DQN is trained using the observations and reward signal from the environment and the low-level DQN is trained using the goal-observation (from meta-controller) and an intrinsic reward for actions executed in the environment generated by the internal critic. Thus the action-value function for the controller is $Q^*(z_c^t, a; g) = \mathbb{E}_a[r^*(z_c^t, a; g, \theta) + \gamma r(z^t, a) + \mathbb{E}_{z_c^{t+1}, a}[Q^{T^*}(z_c^{t+1}, a'; g, \theta')]]$ where g are the sub-goals generated by the meta-controller and z_c is the goal-observation of the controller. This algorithm has shown good performance over settings with sparse and delayed feedback.

A.3 Single-Agent Treadmill Environment Results

For treadmill driving environment, we find that using hierarchical DQN shows improved performance and time complexity when compared to DQN. Fig. 6 shows the results for comparison of Hierarchical DQN vs regular DQN with single autonomous agent training in treadmill environment. Thus, all methods applied to the driving environment use Hierarchical Deep Q-Networks [8] along with the Double Q-learning update [18].

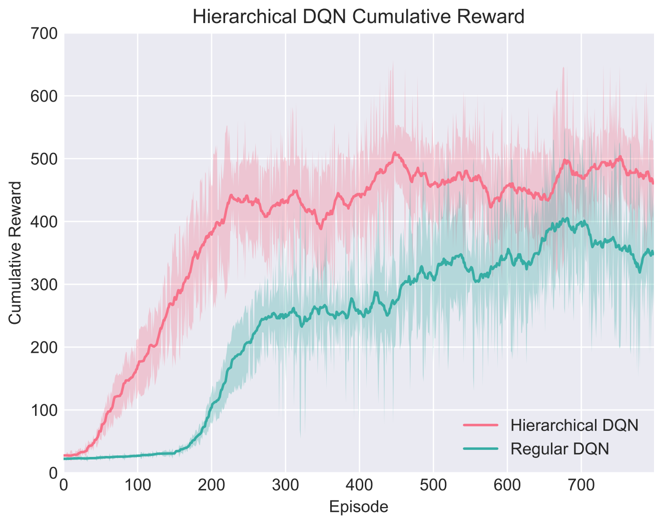


Fig. 6: Comparison of Hierarchical DQN vs Regular DQN on driving environment with single autonomous agent

B OpenAI Particle Environments

B.1 Predator Prey Environment

The environment in this and the next subsection are set up with a sparse co-operative reward structure with a long time-to-horizon to show the validity of our algorithms in such domains and is derived from the environments presented in [11]. In this multi-agent environment, all agents are homogeneous as multiple predator agents learn to capture a prey. The prey is an adversary agent and moves to avoid the predators. The predators receive a reward of 0.5 if they capture the prey independently and a reward of 1.0 when the prey is captured together in the same time-step. The partial observation of the predator only contains the location of the prey and thus agents must communicate to achieve cooperation. The episode is terminated after 1000 time-steps.

The results of our algorithm on the multi-agent predator prey environment are shown in Fig. 7. All experimental results represent the average over 5 runs and the results were smoothed using a moving average. All algorithms use a linearly decaying exploration schedule for the first $100K$ steps from 1.0 to 0.05 and we then use a constant value of 0.05 for the rest of the training. All experiments are run for $7M$ steps and $60K$ episodes. All agents in the environment use parameter sharing of weights and biases of the neural network with a size of 1 hidden layer with 256 hidden units, with a communication channel of size 8 units.

Our centralized training method MA-MeSN was able to achieve good performance in this environment as well. DIAL and IMS are also able to achieve a

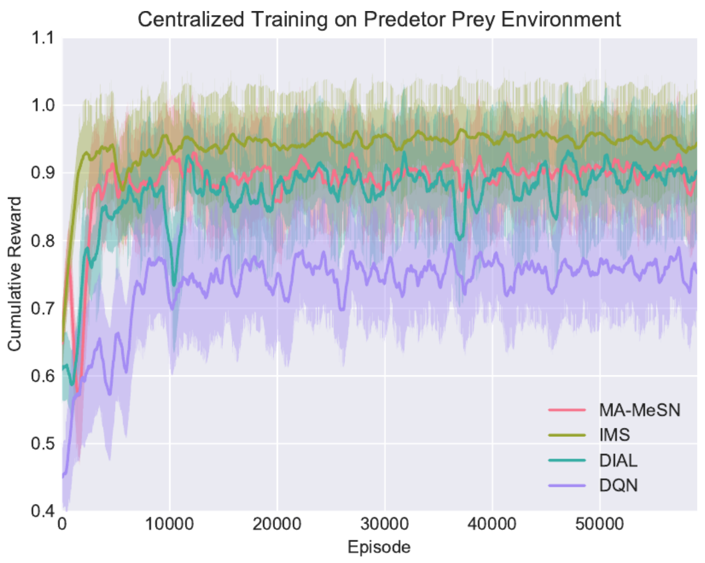


Fig. 7: Comparison of Cumulative Reward on Predator Prey Environment

better performance compared to independent agents which achieve an average reward of 0.75.

Note, we employed importance sampling from the replay buffer for DIAL as per-episode batch training is expensive and unstable. This actually improved DIAL's performance on long time-to-horizon, sparse reward tasks. The results clearly show that our algorithms are extendable to environments with sparse co-operative rewards. MA-MeSN is able to learn a centralized policy for capturing the prey with nearly the same sample complexity as the IMS algorithm. Qualitative analysis of the independent DQN shows that the agents are able to capture the prey cooperatively only when the prey is cornered by chance. The IMS shows the best sample complexity as with the reduced state space of the environment, the messages generated only need to communicate 2 different sentiments, attack or hold. Through cross-validation, we find that we require $P = 3$ communication iterations for IMS with 2 predators.

B.2 Cooperative Communication

In this multi-agent environment, the agents are heterogeneous and must learn a different policy. The environment contains a speaker agent, listener agent and 2 colored landmarks. The listener agent must travel to the correct landmark in the environment (which is assigned randomly at the beginning of the episode) while avoiding the wrong landmark. The reward to reach the correct landmark is 1.0 and 0.0 for the wrong landmark. The listeners' observation only consists of the landmarks in the environment. The speaker agent receives the full state of the environment along with the color of the correct landmark for that episode. The speaker agent must communicate with the listener agent using only 2 bit communication.

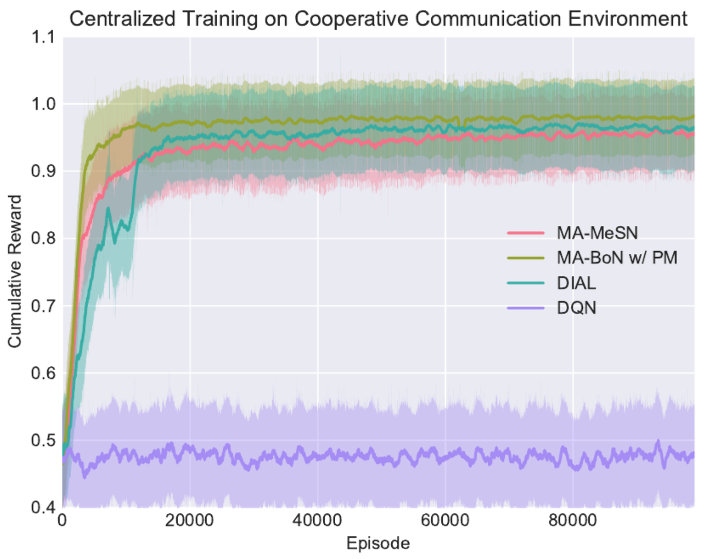


Fig. 8: Comparison of Cumulative Reward on Cooperative Communication Environment

The result of training on the cooperative communication environment is shown in Fig. 8. All experimental results represent an average over 5 runs with the same hyper-parameters as detailed in the previous section. The experiments are run for $7M$ steps and $100K$ episodes. To achieve discrete 2-bit communication between the speaker and the listener, we apply a softmax on the output of the speaker before feeding it to the listener. The learning curves for MA-MeSN, MA-BoN w/ PM and DIAL are compared against independent agents. MA-BoN and MA-MeSN don't use parameter sharing and the communication channel from the listener is multiplied by zero for this experiment. The independent DQN policy picks a random location due to the lack of communication from the speaker and thus achieves an average reward of 0.5.