Deep Multi Agent Reinforcement Learning for Autonomous Driving

by

Sushrut Bhalla

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2020

© Sushrut Bhalla 2020

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Deep Learning and back-propagation have been successfully used to perform centralized training with communication protocols among multiple agents in a cooperative Multi-Agent Deep Reinforcement Learning (MARL) environment. In this work, I present techniques for centralized training of MARL agents in large scale environments and compare my work against current state of the art techniques. This work uses model-free Deep Q-Network (DQN) as the baseline model and allows inter agent communication for cooperative policy learning. I present two novel, scalable and centralized MARL training techniques (MA-MeSN, MA-BoN), which are developed under the principle that the behavior policy and message/communication policies have different optimization criteria. Thus, this work presents models which separate the message learning module from the behavior policy learning module. As shown in the experiments, the separation of these modules helps in faster convergence in complex domains like autonomous driving simulators and achieves better results than the current techniques in literature.

Subsequently, this work presents two novel techniques for achieving decentralized execution for the communication based cooperative policy. The first technique uses behavior cloning as a method of cloning an expert cooperative policy to a decentralized agent without message sharing. In the second method, the behavior policy is coupled with a memory module which is local to each model. This memory model is used by the independent agents to mimic the communication policies of other agents and thus generate an independent behavior policy. This decentralized approach has minimal effect on degradation of the overall cumulative reward achieved by the centralized policy. Using a fully decentralized approach allows us to address the challenges of noise and communication bottlenecks in real-time communication channels. In this work, I theoretically and empirically compare the centralized and decentralized training algorithms to current research in the field of MARL.

As part of this thesis, I also developed a large scale multi-agent testing environment. It is a new OpenAI-Gym environment which can be used for large scale multi-agent research as it simulates multiple autonomous cars driving cooperatively on a highway in the presence of a bad actor. I compare the performance of the centralized algorithms to existing state-of-the-art algorithms, for ex, *DIAL* and *IMS* which are based on cumulative reward achieved per episode and other metrics. MA-MeSN and MA-BoN achieve a cumulative reward of atleast 263% higher than the reward achieved by the DIAL and IMS. I also present an ablation study of the scalability of MA-BoN and show that MA-MeSN and MA-BoN algorithms only exhibit a linear increase in inference time and number of trainable parameters compared to quadratic increase for DIAL.

Acknowledgements

I would like to thank my supervisor, Dr. Mark Crowley, who made this thesis possible by providing valuable research direction and insight in my Masters project. I would also like to thank my friends at University of Waterloo who were always open to collaborating on new ideas and always eager to flush out new research ideas. I would also like to thank my friends and family who supported and motivated me to pursue my Masters degree.

Dedication

This thesis is dedicated to the one I love.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Reinforcement Learning is the method of learning how to behave in an unknown environment using trial and error. Multi-agent Reinforcement learning extends this approach to multiple agents learning to behave in the environment and with each other. This chapter introduces the concepts of Reinforcement Learning and Multi-Agent Reinforcement Learning and other advanced topics used throughout this thesis. I start with discussing Reinforcement Learning (RL) and how Multi-Agent Reinforcement Learning (MARL) ties into the concepts of Reinforcement Learning and Cooperative Game Theory. The MARL section discusses the challenges with training multiple agents in the same environment. I also briefly describe the current approaches used in the literature today to train MARL agents. I then discuss the motivation of this thesis and the practical implications of my work. I then introduce the topic of Safe Reinforcement Learning (Safe-RL) and how the concepts and techniques in Safe-RL can be leveraged to train MARL agents in multi-agent environments.

1.1 Reinforcement Learning (RL)

Reinforcement Learning is a field of research focused on developing solutions which help an actor learn by interacting with the environment. The actor is also referred to as an agent. Learning in Reinforcement Learning algorithms happens under the pretense of a reward signal which the actor/agent learns to maximize, by taking certain actions, after partially observing the environment. The collection of actions taken by an agent in the environment, based on an agent's perception of the environment, is known as the policy of the agent. The agent's perception of the environment is known as the observation

of the environment. The observation received by the agent can include the entire state information or could only represent a partial state of the environment at any give instance of time. These different types of environments lead to different learning algorithms for fully observable environments and partially observable environments. In this thesis, we work with a partially observable environment of a highway driving simulator. During evaluation of learning algorithms, we evaluate the cumulative reward achieved by the agent's policy in a particular environment. Cumulative reward maximization leads to agents learning a self-interested policy and the agent's learning is restricted to self preservation and reward accumulation.

Reinforcement Learning algorithms can be mainly categorized into two broad categories, model-free and model-based algorithms. Model-based algorithms learn a policy in the environment by modeling the environment (and its transition dynamics) based on their observation. This leads to a complex model which might take longer to train but might achieve better performance. A model-free approach does not model the dynamics of the environment, and instead learns a policy based on raw data received from the environment. Model-free approaches are thus usually faster and easier to train. In this thesis, I focus on using model-free approaches.

As this thesis focuses on multi-agent systems with multiple agents learning in conjunction with other agents to learn a cooperative behavior, I will focus most of my discussion on the properties of different reinforcement learning algorithms in multi-agent systems. The greedy learning algorithm of a single agent can restrict the agent's policy, or behavior policy, from exploring a cooperative policy with other agents in the environment. This is because the single agent reinforcement learning algorithms try to extract maximum reward from the environment without regard for other entities in the environment. These entities in multi-agent systems could be agents which could cooperate to achieve a more suitable behavior policy in such environments. Single-agent reinforcement learning is thus restricted to competitive games only where an agent learns to interact against adversaries. Cooperation in multi-agent reinforcement learning can still be achieved if the reward signal provided by the environment explicitly encourages cooperation among agents. However, the requirement of a reward signal from the environment which helps agents learn an inter-agent cooperative policy is an unsatisfactory pretense and thus these methods could not be easily scaled to environments requiring cooperative behavior policies from multiple agents. In this thesis, I present techniques to train multiple agents with a cooperative policy which scales to large scale complex domains.

1.2 Motivation

There are many real-world multi-agent environments with an infinite horizon which are hard for a single agent to solve using an independent policy such as autonomous driving. Autonomous driving requires algorithms whose learning is scalable to a large scale and provides minute control over various aspects of the model. It also requires the model to be modularized enough so it could be validated for safety. Most current autonomous driving research focuses on modeling the road environment consisting of only human drivers. All other drivers on the road are considered as part of the environment to avoid modelling of each driver's behavior. Under this simplification, the autonomous car is treated as a single agent in the environment and it learns a self-interested behavior policy. Such policies are plausible but exhibit an overly pessimistic behavior on the road to maintain high safety standards. However, with increasing congestion and autonomous vehicles on the road, a greedy policy would lower the quality of traffic and diminish the throughput of traffic on the roads. A shared cooperative policy among multiple cars might also be better at achieving the goal of better traffic control and higher traffic throughput. The communication between agents needed for cooperation among cars is also not guaranteed to be available and thus agents need a method of cooperative without communication.

1.3 Multi Agent Reinforcement Learning (MARL)

Multi Agent Reinforcement Learning is a field of research for learning behavior policies for multiple agents using reinforcement learning techniques. Training multiple agents in continuously evolving environments using a self-interested agent will lead to sub-optimal results or worse, the training will fail to converge as shown in the experiments of this thesis. A typical approach for learning policy for self-interested agents is using a well-known algorithm, DQN (Deep Q-Network). This is a model free learning algorithm which uses the underlying transition dynamics of the environment to learn a value function, $Q(o)$, of the environment. A $Q-$value model maps the observation (o) of the agent to the predicted cumulative reward based on the agent's policy. The $Q(o)$ value model relies on the transition dynamics of the environment. A stationary transition dynamics implies, the stochastic transition probability $T_{time}(s, s')$ of the environment is constant over time. A non-stationary transition dynamics implies, the stochastic transition probability $T_{time}(s, s')$ for any 2 given states is not constant over time. The policy learnt using many model-free and model-based independent reinforcement learning algorithms claims the transition probability of the environment should be stationary. In a Multi Agent environment, where

multiple agents are learning simultaneously, the transition dynamics of the environment are governed by the evolving policies of the multiple agents. The evolving policies of agents over time, leads to a non-stationary transition dynamics of the environment and thus traditional single-agent reinforcement learning algorithms fail to converge [41].

To overcome the problem of non-stationarity in the environment during training of multiple agents, the current literature proposes multiple methods, including memory based models, communication between agents, shared policy independent agents and independent agents with gradient information sharing. Memory based models use a memory module to save the state transitions in the environment in a long term memory. During training, the agents can retrieve the transitions from the memory module and replay them for training. The advantages of using a long term memory module is that the agents can iteratively learn a better policy while the transition dynamics of the environment are stationary. Communications can be used between agents is a centralized MARL training method where agents communicate with other agents in the environments. The message passed between agents can communicate an agent's intent at a given state based on its current policy. The intent of the message is used by the receiving agent as a guide to improve its decision making. The information sharing between the agents reduces the non-stationarity in the transition dynamics of the environment and improves the learning process. In many environments, research has shown that having a shared policy function for all agents in a multi-agent environment can lead to a convergence in MARL. This approach has only been tested in limited scenarios and based on the literature review and my experimentation does not scale well to larger environments. The message shared between the agents is generated using the policy network and trained using policy gradients. This approach leads to each agent having the additional responsibility of learning a good policy in the environment and delivering effective messages to other agents in the environment [7]. Current approaches also show a poor performance in large-scale environments with sparse rewards and a long time to horizon. My thesis works is focused on such environments.

Recent research work [11, 13] has focused on learning decentralized MARL policies by sharing the gradient information between agents during training. The direction of gradient computed by an agent to update its value/policy function approximator is used as a reference by the other agents to update its gradient to align with the original agent. This approach leads to multiple agents learning a cooperative policy and training in a fully decentralized manner. This training process can be inefficient and slow depending on the environment.

1.4 Deep Learning

Deep Learning refers to the modern approach for using deep Neural Networks to learn a functional approximation from some given inputs to some possible outputs. The neural networks can be trained in two different ways, supervised or unsupervised, methods. The supervised training methods for neural networks minimize the error between the predictions of the neural network and the true output value. The error is reduced by updating the weights of the neural network in the direction of least error. The most common optimization technique used in training neural networks is Stochastic Gradient Decent (SGD) which uses a simple step based gradient estimation based on a batch sampling of the training data. The computed gradients are applied to the weights of the neural network using backpropagation. This technique ensures that the predictions of neural network would be a step closer to the true predictions.

Neural networks are widely used in RL and MARL training methods as they are used as a good functional approximation to predict the value of a state. The value of a state in RL represents the value an agent would receive if the agent were to follow the current policy to the end of the task or episode. The value of the state is computed as a the cumulative sum of rewards from the given state to the end of the task or episode; when following the given policy. Current RL research uses neural networks to predict the mean value of a particular state; given a policy. The gradient calculations to update the neural networks are a little different than regular supervised learning and depends on the baseline training method used. The baseline methods can be categorized into two different categories, value based methods and policy based methods. The value based methods focus on learning a value function of the current state and a greedy selection policy is used to maximize the value of an agent's policy. In policy based methods, the agents learn a stochastic policy directly from the transitions from the environment. The training gradients optimize the policy directly in the direction which maximizes the cumulative value achieved by the agent.

1.5 Communication between agents

In centralized training of MARL algorithms, the focus is on communication between the agents. Agents are allowed to communicate with each other, this includes speaker and listener agents. Communication between agents can be divided based on the type of communication between agents. Agents can engage in *Cheap Talk*, in which a speaker sends a costless message to other agents and the listener agents can choose to listen to it or ignore the message before deciding on a behavior action. This approach is also considered

as "doing by talking". The information shared between the agents doesn't need to be trained. The listener agents thus necessarily don't need to formally use this information in the final decision making process. Agents can also engage in *Signalling games*, which in contrast don't include explicit messages sent by a speaker to a listener, instead include an agent indicating their intent by taking a behavior action in the environment. *Signalling games* can thus be viewed as *talking by doing*, whereas *Cheap talk* can be viewed as *doing by talking*. Another communication model is *Speech Act Theory*, which is similar to *Cheap talk*, but speaker agents send a message which is intended to convey information (which could include policy information or the agent's private observation information) to the listener and also influence change on the policy of the listener. Due to the constrained behavior of the agents, *Speech Act Theory* can only be applied in cooperative games and not in competitive games. In competitive games, agents would revert back to using *Cheap talk* to gain an advantage over other agents.

In this thesis, I focus on using *Speech Act Theory* as the communication model between agents. The message shared between agents in large scale POMDPs is supposed to be self-revealing and self-committing. The listener relies on this information from the speaker to make its own decision. In Speech-act theory, the agent takes 2 different types of actions, behavior(locutionary) action and message/non-behavior(illocutionary) action [37]. The behavior action taken by an agent affects a change in the environment and a message action does not affect any direct change in the environment. The experiments in this thesis focus on partially observable large scale environment where each individual agent must independently take its action. To achieve an optimal policy, agents need to communicate useful information regarding their private observation of the environment with each other to cooperatively find an optimal policy. The speaker agents' message actions also need to be self-revealing and self-committing so that the message received by the listener agent leads to a direct correlation to the transitions in the environment. Such correlation leads to better *Speaker Consistency* and *Instantaneous Coordination* between agents. Speaker consistency is a metric used to measure the correlation between an agent's message action and their behavior action. Instantaneous coordination is a metric which measures the correlation between the speaker's message action and the listener's behavior action. High values for the correlation between the speaker and listener thus leads to a better cooperative policy between agents in a multi-agent system.

1.6 Contributions

In this thesis, I propose four novel methods for MARL environments. I propose two centralized training algorithms for MARL environments which extend the use of communication between agents to achieve a centralized policy. I also propose two decentralized methods using memory modules and behavior cloning which can be used to during execution phase in the MARL environments. The methods and the policies trained using these methods are evaluated against current state of the art techniques in centralized training of MARL agents using communication channels. The final policy is also evaluated qualitatively and quantitatively using metrics developed by the MARL community.

The first centralized approach uses a mesh communication channel which leads to a connection between every pair of agents. The agents are free to communicate a discrete message at each time-step. Each agent only receives their private partial observation of the environment and possibly a message from the other agents in the environment. At each discrete time step, all agents perform a 2-step process of generating a message action (which gets shared with the listeners) and generating a behavior action (after listening to the speakers message action) which is performed in the environment. The speaker agent is free to decide to use cheap talk or meaningful messages which comply with speech-act theory. The listener agent is free to decide to use the message as a guide for prediction of the value/policy function in conjunction with the current observation. The discrete message is a one-hot vector of length 12 and thus agents can only send 12 different types of messages. Varying the restriction on the length of the message size leads to different policies learnt by the agents. The learning can also collapse if the length of the message is too short for the environment.

The second centralized approach employs the use of a broadcast message action which is generated by a centralized message generation model. The centralized model receives message actions from each individual agent and composes a unified message action for all agents. This unified message is used by all agents in the environments to generate a behavior action for the environment. This approach limits the size of the message shared in the environment to a constant value for all agents. This approach leads to faster inference model with reduced system memory requirements. This approach is valuable for environments which exhibit a shared resource between the agents.

The first decentralized approach uses the behavior cloning method for training. The decentralized multi-agent model is needed because in real-time the centralized message sharing is slow and can adversely affects an agent's ability to take actions in real-world MARL environments. The decentralized model allows each agent to evaluate their model

independently from other agents and generate an action prediction. The centralized trained policy from the previous two methods is used as an expert policy which is mimicked by an independent agent. The independent agent is trained using behavior cloning. This approach requires minimal parameter tuning to achieve decentralized execution in the environment.

The second decentralized approach generates a decentralized multi-agent model using the centralized approaches discussed above. This approach leads to slight degradation in the cumulative reward achieved by the previously trained centralized policy. However, the degradation in performance is controllable by varying the degree of past history of observations used during execution. The approach used in this thesis allows parallel training of the decentralized policy along with the centralized policy training.

The rest of the thesis presents the results generated from the above approaches and their comparison with current state of the art multi-agent reinforcement learning methods like DIAL [10] and IMS [39]. This work compares the algorithms on a highway multi-car driving scenario with multiple agents learning to drive with cooperation. The highway scenario is imitated with robots driving on a treadmill. I evaluate the centralized methods proposed in this thesis on the highway driving scenario along with the proposed decentralized algorithms. I evaluate the messages generated by the centralized training methods and how these messages affect the policy of the listener. I also evaluate the scalability of my approach by increasing the scale of the highway driving environment.

Chapter 2

Literature Review

Multi-Agent Reinforcement Learning (MARL) has a rich literature (particularly in the robotics domain [6]) and with the advent of Deep Learning, it is becoming a highly active research field. MARL algorithms use the independent Deep-RL approaches as baselines, such as model-free Deep Q-learning (DQN) [32], policy-based actor-critic (A2C) [31], and extend them to cooperative or competitive multi-agent environments. Independent and cooperative tabular Q-learning with multiple agents has been studied in [41]. The empirical evaluation shows that cooperative Q-learning can be achieved by sharing other agents' private observations, policies or episode information. The cooperative Q-learners are slow to learn but eventually outperform independent Q-learners in a partially observable MDP environment. Hyper Q-learning extends the idea of observation sharing to policy parameter sharing between the agents. This change allows the agents to condition their policy on the changing parameters of the other agents and thus avoid the problem of non-stationary environments in MARL [43].

2.1 Independent Policy Learners

Independent agents can also learn to behave in a cooperative manner under certain circumstances, like presence of cooperative reward or the presence of teacher feedback. Independent policy learners could still diverge in a multi-agent environment. The work in [44] shows that independent agents could be trained to cooperatively behave in a Starcraft environment when they are controlled by a single central model which can observe the full state of the game. This leads to a centralized execution policy for multiple agents which is not feasible in most real-world environment as they need to be self-reliant.

2.1.1 Deep Q-Networks with Stabalized Experience Replay

The authors in [14] successfully train multiple independent agents by implementing a stabilization on the experience replay memory. The replay memory is used by Deep Q-Network algorithm as a history of sample trajectories for training. The replay buffer stabilize the training of Deep Q-Networks as they are able to optimize on new and old samples in the same mini-batch of training. This poses a problem in multi-agent setting where there are multiple agents evolving over time. The older samples in the replay buffer represent the old transitions of other agents which are no longer valid as their policy has evolved. The stabilization of the replay buffer in a multi-agent system is done by prioritizing newer experiences in the experience buffer for training as they represent the current policy of the other agents. This allows the agent to focus on the latest transition dynamics of the other agents and stabilizes the policy optimization as it reduces the non-stationarity in the mini-batch of samples used for training. I also compare the training of our decentralized policy against the independent agents trained using Stabilized Experience Replay (**SER**).

2.2 Optimization based models

The work done in [11, 13] has shown that agents can be trained independently in a multi-agent environment by sharing the update gradients between the agents. The following parameter is added to an agent's policy network update:

$$\left(\frac{\partial V^1(\theta_i^1, \theta_i^2)}{\partial \theta_i^2}\right)^T \frac{\partial^2 V^2(\theta_i^1, \theta_i^2)}{\partial \theta_i^1 \partial \theta_i^2} \cdot \delta\eta \tag{2.1}$$

and represents the change in one agent's parameters based on the gradients of the other agent. This approach thus leads to agents updating their parameters which align with other agents and thus lead to convergence after training. The gradient sharing can also be eliminated by modeling the policy of other agents and performing local updates based on the model of other agent's policy.

2.3 Emergent Communication

Communication between multiple agents is considered as a source of intelligence in agents. The communication between agents can be supervised or unsupervised. Supervised communication between agents requires hard-coded rules which must be followed by the agents

for each message which is shared. In our scenario of training multiple autonomous cars, a hard-coded message which would help in training of the MARL system would be the location of the agents. However, generating hard-coded information is not a solution which would scale well to large domains. The agents can also learn to communicate via self-generated unsupervised messages. If trained properly, the message sharing between agents could lead to emergent communication between agents which can be exploited to train multiple agents in a centralized manner. Effective communication channels in MARL can be trained using backpropagation as shown in the papers for DIAL [10] and IMS [39].

2.3.1 Iterative Message Sharing (IMS)

The authors in [39] 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 action-values. This Iterative Message Sharing (**IMS**) is iterated P times in a single discrete time-step of the environment before the final action for all agents at that time-step is computed. The message generated by all agents is aggregated using the averaging function and passed back to the agents for re-evaluation of the agent's next message. The authors suggest that multiple iterations of message sharing are required because only one agent seems to be communicating at a single iteration of message sharing. This approach leads to a model which can be easily scaled to a variable number of agents but increasing the number of agents will also increase the cost of message sharing as the agents will require larger number of iterations to communicate effectively. The model is trained using the optimization criteria used in Deep Q-Networks(DQN). The gradients computed from $Q - error$ (error in the state value predictions) calculation is used to train the individual agents. The message action and behavior action are generated by the same network. Thus a single policy update is enough to allow their model to converge to a cooperative policy between agents.

2.3.2 Differentiable Inter-Agent Learning (DIAL)

Differentiable Inter-Agent Learning **DIAL** [10] also trains cooperative agents to communicate through back-propagation, for sequential multi-agent environments. The speaker agents send a message to the listeners. The listener agent receives the message (m_{t-1}) from the speaker and uses it to predict an action a_t. The agents use their current observation along with past messages from other agents to predict the best action. The delay in

the message received by the listener can cause a sub-optimal convergence of the cooperative policy in dynamic environments. Dynamic environments evolve over time which could lead to the message m_{t-1} being less purposeful at time t. The agents are trained in a similar fashion as IMS. The final policy predictions are used to compute the gradients for the model based on the prediction errors. The gradients help improve the shared message and policy models. Due to shared policy and message models, the agents are able to generate self-revealing messages; which help in reducing the non-stationary in the environment's transition dynamics and learn a cooperative policy.

The above approaches perform well in multi-agent environments. IMS tested their approach on a small autonomous driving test-bed with intersections. The goal was to train agents to effectively communicate to alleviate need for lights at an intersection. The tests for DIAL included the Switch-Riddle prison problem and a self developed MNIST game where agents must learn the parity of the digits they observe. These task have many multi-agent properties, however lack the property of being dynamic and large scale. I apply these algorithms to our large scale driving environment and compare the results with the novel approaches presented in this thesis. My work differs from these approaches in two ways. (a) The iterative network structure of the communication protocol is removed and replaced with a feed-forward neural network. (b) The centralized model is only used during training and a decentralized policy using a memory module is trained in parallel for execution. This is done because the communication among agents in autonomous driving environment is not guaranteed.

2.4 Decentralized Execution in Multi-Agent Systems

Work in [28, 33, 21] has shown that the MARL agents could be evaluated with discrete communication channels by using a softmax operation on the message. Recent work from [28, 12] has shown that the actor-critic algorithm could be naturally extended to truly decentralized execution by completely eliminating communication channels. The centralized training is achieved by using a shared critic for homogeneous agents or by using a separate centralized critic for every agent which maps the observation and action of every other agent to compute the state-action value function.

In this thesis, I focus on developing decentralized algorithms which derive directly from the centralized policy trained using message sharing between the agents. My work focuses on training in a centralized manner to achieve maximum cooperation among agents without providing agents with reward functions or methods which are created by developers and could be biased towards the environment or the final expected policy. My work uses the

final centralized policy to guide the decentralized policy between agents and thus achieves a better policy than the decentralized training techniques discussed in literature.

2.5 Other research on communication between agents

There is a vast literature on the emergence of communication protocols between multiple agents in the same environment [24, 33, 39, 8] which also use backpropagation to pass gradients between the continuous channels and establish an effective communication protocol. This has also been used to relay visual information between agents [8]. The multi-agent setting has also been considered in the context of game theory [25] and has shown how using the appropriate hand-crafted reward structure could lead to the agents behaving cooperatively or competitively. Similar work with explicit reward function structuring to achieve cooperation and competition is shown in [40]. Our work differs from these approaches as the multi-agent autonomous driving environment doesn't provide explicitly hand-crafted reward function for cooperation, though our proposed algorithms are extendable to cooperative reward domains. MARL has also studied in [17] which shows that parameter sharing between independent value/policy-networks for homogeneous agents helps in learning cooperative policy.

Chapter 3

Background

3.1 MDP and POMDP

Markov Decision Process (MDP) is way to model the environments to apply a reinforcement learning algorithm. MDPs exhibit the property that the current state depends on a finite set of past states, and the transition dynamics of the environment are stationary. The transition dynamics of the environment represent the probability of transition between two states given an action. The transition dynamics can be stochastic or deterministic. In MDPs, the agents are assumed to be able to observe the entire state of the environment, whereas in POMDP (Partially Observable-Markov Decision Process), the agents can only partially observe the state space. In MDPs, the agents can make an action prediction based on the latest observation received from the environment. In POMDPs, the agents must use the past observation history to make an action prediction. In Multi-Agent systems, I work with partially observable discrete stochastic games. Due to the large number of agents training in parallel the environment's transitions are no longer stationary. Using single agent Reinforcement learning techniques can lead to divergence of the training agent in non-stationary environments.

3.2 Deep Q-Networks (DQN)

The Deep Q-Network (**DQN**) is an extension of the tabular Q-learning algorithm which uses neural networks to approximate the action-value function $Q(o, a)$ [32]. o denotes the observation of the agent in the environment, a is the action, r is the reward received from

the environment, and t denotes the discrete time-step. γ is used as a discount factor when calculating the cumulative reward received during an episode. The value of the current state can be computed using the formula,

$$Q(o^t, a^t) = \mathop{\mathbb{E}}_{a}[r(o^t, a^t) + \gamma \mathop{\mathbb{E}}_{a}[\max_{a^{t+1}} Q^T(o^{t+1}, a^{t+1}; \theta')]] \tag{3.1}$$

Q^T (the target Q-value) is generated using the target network which computes the estimates for next state-action values and is periodically updated to the newest online network ($Q(o^t, a^t)$). DQN uses the same update rule as Gradient Q-Learning; except a separate target Q-network is used for the action-value evaluations of the next state. DQN uses Bellman equation to iteratively arrive at the optimal policy for an agent. The update rule minimizes the following error:

$$\text{TD-Error}(w) = [Q(o^t, a; \theta) - r - \gamma \text{max}_{a^{t+1}} Q^T(o^{t+1}, a^{t+1}; \theta^T)]^2 \tag{3.2}$$

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. Gradient Descent is used to train 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 Eval/online network is updated continuously and the Target Network is updated once every fixed number of iterations.

3.3 Hierarchical DQN

Hierarchical DQN [23] 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. This model allows us to split the responsibility of the agent into different models. 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 is generated by the internal critic. Thus the action-value function for the low-level controller is

$$Q^*(z_c^t, a; g) = \mathop{\mathbb{E}}_{a}[r^*(z_c^t, a; g, \theta) + \gamma r(z^t, a) + \mathop{\mathbb{E}}_{z_c^{t+1}, a}[Q^{T^*}(z_c^{t+1}, a'; g, \theta')]] \tag{3.3}$$

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 in environments with sparse and delayed feedback.

3.4 Multi-Agent Reinforcement Learning

In this thesis, I consider a general sum multi-agent partially observable 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, which can act independently of each other. The game environment transitions between states $s \in S$, and the agents observe an observation $z \in Z$. The observation z is generated using the function $z = O(s, x)$, which maps the state of the environment to each agent's 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 $x \in X$. I 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. I use the superscript t to represent the discrete time-step. All agents receive a reward from the environment based on a utility function R, which provides agents with an instantaneous reward for an action a_i. The game environment represents a Decentralized Partially Observable Markov Decision Process (DEC-POMDP) [3]. The agents can send and receive cost-less discrete messages between each other, which are bit vectors, represented as m_i^t and are considered as a communicative/messsage actions by the agent. However, the game environment does not provide a utility function for the communication actions performed by an agent. All agents that share the same observation space and have the same objective in the environment can be considered as homogeneous agents. The message generated by an agent x_i training in MARL environment at discrete time-step t is represented as m_i^t. 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.

3.5 Behavior Cloning

Behavior Cloning is a method for inverse reinforcement learning in which an agent learns a policy by following an expert act in the environment. Behavior Cloning is used to learn an approximate policy $\pi_\theta(z^t, a^t)$ for the environment based on the expert policy $\pi^*(z^t, a^t)$, through supervised learning techniques; where π_θ represents the learned policy and π^* is the optimal policy based on the demonstrations D of the expert. The learner agent receives the observation z^t from the environment and must select a greedy action a^t which maximize the log-likelihood of the learner's policy π_θ from the ground truth π^*. The objective function

can be represented as follows:

$$\pi_\theta^* = argmax_\theta \log \sum_{(a,z)\in D} \pi_\theta(a, z) \tag{3.4}$$

3.6 Communication Between Agents

Communication between agents in a multi-agent environment can be sub-divided into three different categories depending on the method and nature of information shared between the agents. The first method is cheap talk, or "doing by talking", in which agents communicate information to other agents dependent on their observation of the environment. The messages shared between the agents are not incentivised to a convey a property of the environment or the agent's policy. This type of communication between agents rarely leads to a cooperative multi-agent policy as the communication between agents is not focused on the task at hand.

The second method is signaling games, or "talking by doing", in which agents share information by performing an action in the environment. This type of communication between agents requires an environment where agents can observe agent's behavior in a temporal fashion as no actual data is being shared between the agents. This approach is very useful in multi-agent autonomous driving, as currently driving involves estimating a vehicle's future trajectory and actions based on its current behavior. This approach is widely used in this thesis to achieved decentralized execution in multi-agent environments.

The third approach in multi-agent environment is speech-act theory which can also be thought of as "doing by talking". The key difference between cheap talk and speech-act theory is that the messages shared between agents are focused on the cooperative task at hand. This approach can only be effectively used in cooperative multi-agent environments as the agents would not share helpful messages in multi-agent competitive environments. This approach is widely used in Deep-MARL literature where the message shared between agents is trained using backpropagation. This allows the speaker agents to send messages which generates a better cooperative policy. The speech-act theory suits our needs well when building cooperative centralized multi-agent policies for autonomous driving environment, where agents can share data with each other and must behave cooperatively.

Chapter 4

Methods for Scalable Multi-Agent Reinforcement Learning

In this chapter, I propose two methods for centralized training of cooperative policies in MARL domains, and present two methods for fully decentralized execution of the cooperative policy. This work is currently published in Canadian AI 2020 conference proceedings. Training in a general sum game using independent DQN leads to divergence as can be seen in the results. The online learning of agents in the environment leads to non-stationary model of the environment. Thus by learning in a multi-agent environment, the agents negatively affect the others performance, which leads to divergence.

Note that each agent in a multi-agent environment is required to learn a communication policy and an action policy. The communication policy is used to send messages to other agents and the action policy is used for in-environment actions (e.g, navigation). The DIAL and IMS algorithms allow the agents to learn a combined policy for communication and action policy using a the same TD-error generated from the environment. The proposed centralized training algorithms are designed to achieve an independent communication and action policy model. As shown in the experiments section, this distinction allows the proposed algorithms to easily learn a cooperative policy in large scale environments.

Next, to achieve partial decentralized training, using discrete messages between agents while maintaining differentiability, we introduce a Gumbel-Softmax [21] operation on the continuous message generated by each agent [33]. Gumbel-Softmax generates a continuous approximation of the categorical distribution by replacing the argmax operation in Gumbel-max trick with a Softmax operation. This approach only allows for partial decentralized execution of the policy.

To achieve fully decentralized policy, I propose two methods. The first method uses imitation learning of the final cooperative behavior policy to learn new policy which is completely independent. This can be seen as learning from a teacher or an expert. The second approach introduces per-agent memory modules. This approach eliminates all communications between agents after training has converged, while maintaining the policy achieved in the centralized methods.

All methods proposed in this thesis can be extended to use a Hierarchical Deep Q-Network ($hDQN$) model. Hierarchical DQN proposes that the final behavior policy can be split between 2 or more models. The split is achieved by logically splitting the utility function into hierarchical levels. For example, in autonomous driving environments, the higher level models can be concerned with learning a more abstract policy like navigating between different lanes, navigating to different sections of the road. The lower level model can be concerned with a more focused behavior policy to learn, ex, learning to turn the wheel to achieve the correct direction, or accelerating and decelerating to achieve a particular goal. In this thesis, all experiments performed on the treadmill driving environment use a hierarchical DQN. The individual methods detail how a model can be converted into a hierarchical model.

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 network structure where the communicative and non-communicative (dynamics of the agent) actions are generated using the same neural network. This approach forces a strong correlation between the communicative and non-communicative policies, but leads to sub-par results. I suspect that the poor performance is because of the joint objective of maximizing the non-communicative and communicative action policy's cumulative reward. Having a conflicting or uncorrelated objective leads to sub-optimal solutions.

I recognize this as a bottleneck to achieving scalable multi-agent policies and present a scalable multi-agent network structure in Fig. 4.1. I use a separate neural network for communicative and non-communicative actions. The f'' neural network maps the message received from the other agents $m_{\mathbf{-i}}$ along with its partial observation of the environment o_i to a state-action-message value function $Q(s, a, m)$. I refer to this network as the policy

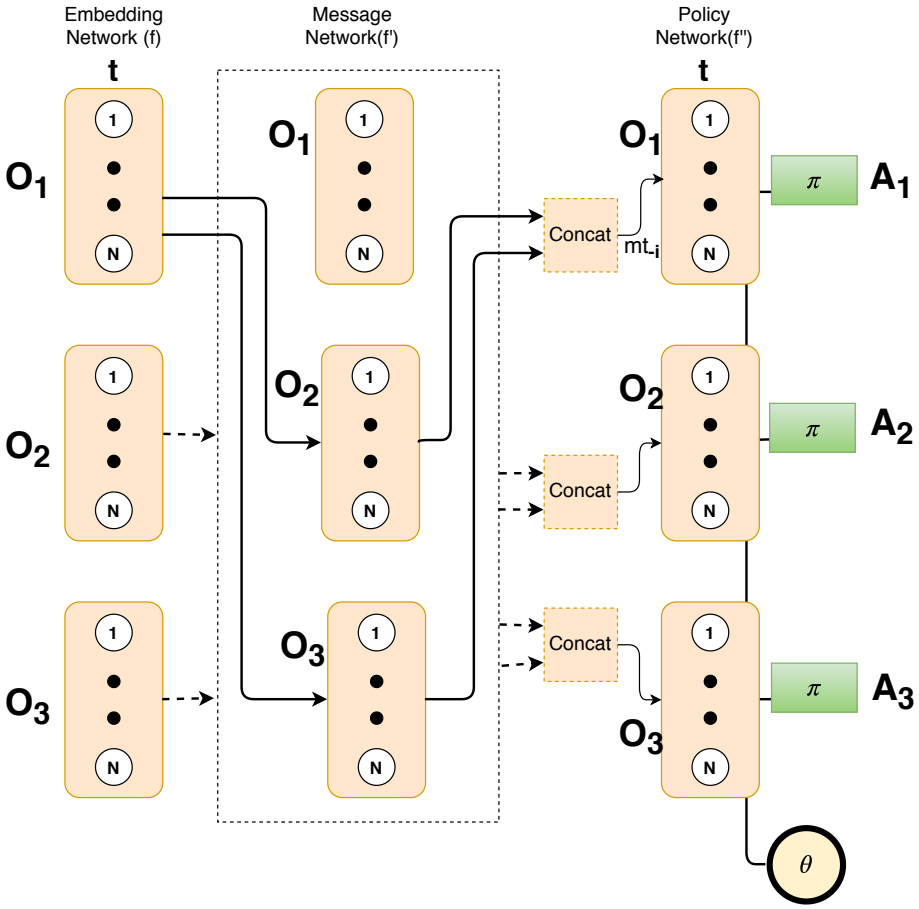


Figure 4.1: Architecture of Multi-Agent Message Sharing (MA-MeSN)

network and is shown in the far right side of Fig. 4.1. The message is generated by the other agents $x_{-\mathbf{i}}$ using the neural network approximator f' which maps the agent's private observation to a communication action $m_{-\mathbf{i}}$. I refer to this network as the message network. The message sharing interaction/negotiation can be extended to multiple iterations for faster convergence. I only allow a one time message sharing (all agents send a message and receive a message) between all agents in this algorithm. The complete architecture is shown in Fig. 4.1.

In contrast to previous work in DIAL, I optimize the message network using the joint gradients of all policy networks as shown in Alg. 1. The gradients for the online batch are combined and backpropagated in the message network. Optimizing the message network with joint gradients leads to messages which are generalizable/compatible to all agent's behavior policies. Without the joint optimization, we see a larger variance in the results of cumulative reward during evaluations.

4.1.1 Comparison to Previous Work

This approach has two advantages over DIAL. The messages $m^t_{-\mathbf{i}}(z^t_{-\mathbf{i}}, f(z^t_i))$ are conditioned on the entire observable state at time t, as opposed to DIAL, where messages $m^t_{-\mathbf{i}}(z^{t-1}_{-\mathbf{i}})$ are a function of the previous time-step private observation of each agent $z^{t-1}_{-\mathbf{i}}$. Sending a message in the previous discrete time observation requires the speaker to learn the stochasticity of the environment. Even with good model approximations, neural networks will generate messages $m^{t-1}_{-\mathbf{i}}$ associated with the mean of transition probabilities. On the other hand (in MA-MeSN), training the message network to generate messages $m^t_{-\mathbf{i}}$ for the current observation only requires the understanding of the agent dynamics and the utility function. Secondly, this allows for the MA-MeSN algorithm to train offline using a step based experience replay. However, optimization logic in DIAL requires sampling episodes from the online policy to train the communication channel; because the agents need to stay up-to-date with the non-stationary evolving model of the environment.

4.1.2 Hierarchical MA-MeSN Model

The MA-MeSN model uses the hierarchical DQN as the baseline model for training. The model shown in Fig. 4.1 shows the top-level DQN in the hierarchical model. The observation O_i represents the observation from the environment. The action A_i represents the meta-action generated by the agent. The action A_i is concatenated with the observation O_i of the environment and passed as input to the lower level controller DQN. The lower

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

for $i = 1, \ldots, N$ **do**
  Initialize replay memory $\mathcal{D}_\rangle : i \in \{1..N\}$ 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* $= 1, \ldots, E$ **do**
  for $t = 1, \ldots, T_{convergence}$ **do**
    for $i = 1, \ldots, N$ **do**
      Select a random action a_i^t with probability ε
      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''_θ
      $\Delta Q_{f''_i} = y_i^j - Q_{f''_i}(o_i^j, m_{-\mathbf{i}}, a; f''_{i,\theta}$
      Apply gradients $\nabla\theta_{i,f''}$ to $f''_{i,\theta}$
      for $j = 1, \ldots, N$ **do**
        $\nabla\theta_{\mathbf{j},f'} \leftarrow \nabla\theta_{\mathbf{j},f'} + \nabla\theta_{\mathbf{j},f''}$
        {Collect gradients using residual gradients from policy networks.}
      end for
      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

level DQN is an independent model and doesn't use message sharing. The lower level DQN receives a positive reward when it reaches the goal generated using action A_i and no reward otherwise. For the rest of the thesis we only deal with generating higher level goals and message sharing generated as result of it. All competing methods are upgraded to use hierarchical DQN as their baseline model instead of simple DQN. This approach significantly improves the performance for all methods. In Sec 5.3.1, I the result of using

DQN vs Hierarchical DQN and quantitatively show that hierarchical DQN can achieve a better overall policy with single agent methods.

4.2 Multi-Agent Broadcast Network (MA-BoN)

Message sharing between the agents at the current discrete time-step (MA-MeSN) allows for stability during training in large scale environments. However, the message and policy networks for all agents needs to be individually evaluated to compute the state-action value for each agent. In this method, I propose using a centralized message generation network as shown in Fig. 4.2. The neural network f' maps the shared partial observation encoding from all agents to a broadcast message bm^t. I study the properties of MA-MeSN and MA-BoN in Sec. 5.3.4 and show that this network is feasible in multi-agent general sum games.

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, which can be a bottleneck for real-time applications as further discussed in the results Sec. 5.3.3. Contrary to the iterative message sharing network I don't share the observation encoding $f(z_i)$ with the policy network g''. Empirically, feeding the observation encoding $f(z_i)$ directly to g' leads to a negative side-effect of residual block skip-connection where the optimization to the broadcast messages gets short-circuited.

The training algorithm for MA-BoN is similar to the MA-MeSN algorithm and is presented in Alg. 2. The main difference is on line 18 where the cumulative policy gradients are applied to only the broadcast network. Due to the use of a single centralized communication model in MA-BoN, this model can only be used in symmetric cooperative multi-agent environments. MA-BoN can also be decentralized by the use of a memory module $LSTM_\pi$ trained parallel to the policy network (MA-BoN-MM) as detailed in Sec. 4.5. This algorithm reduces the the training time considerably as the training algorithm exhibits a running time of $O(N^2)$, instead of $O(N^3)$ from MA-MeSN. Both the MA-MeSN and MA-BoN model use Gumbel-Softmax operation on the message policy to generate stochastic messages which can be easily converted to discrete messages for partial decentralized execution.

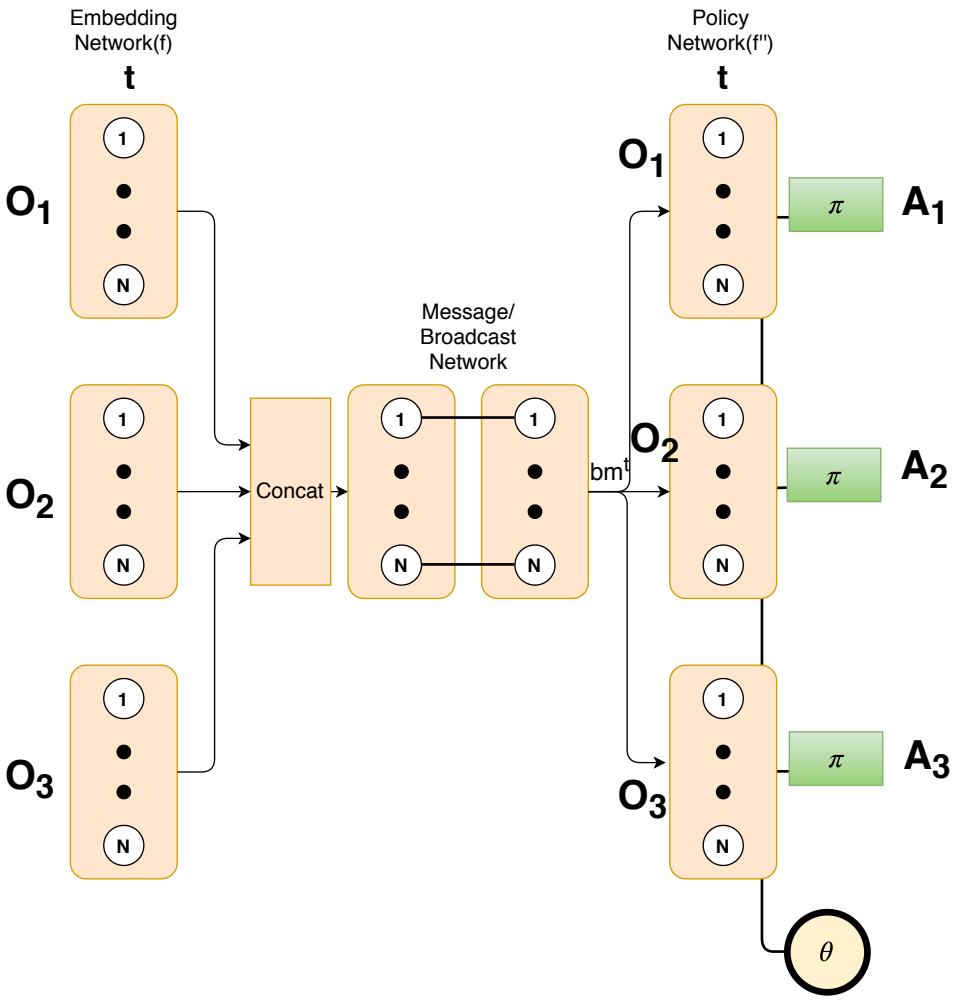


Figure 4.2: Architecture of Multi-Agent Broadcast (MA-BoN).

Algorithm 2 Multi-Agent Broadcast Network (MA-BoN)

for $i = 1, \ldots, N$ **do**
  Initialize replay memory $\mathcal{D}_\rangle : i \in \{1..N\}$ 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* $= 1, \ldots, E$ **do**
  for $t = 1, \ldots, T_{convergence}$ **do**
    for $i = 1, \ldots, N$ **do**
      Select a random action a_i^t with probability ε
      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''_θ
      $\Delta Q_{f''_i} = y_i^j - Q_{f''_i}(o_i^j, m_{-\mathbf{i}}, a; f''_{i,\theta}$
      Apply gradients $\nabla\theta_{i,f''}$ to $f''_{i,\theta}$
      Collect gradients $\nabla\theta_{\mathbf{i},f'}$ from all policy networks.
      Apply gradients $\nabla\theta_{f'}$ to f' {f' is the broadcast network.}
    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

4.2.1 Comparison to Previous Work

The work done in IMS (Iterative Message Sharing) [39] provided a method for generating a generalized (broadcast-type) message, which is shared by all agents. This approach is useful when there is a centralized communication entity. This allows for reduced communication network bandwidth. IMS uses an iterative method for computing an "iterative" broadcast message for the agents. The embedding from all agents is accumulated using weighted sum and passed back to the agents. This process is repeated multiple times to share the information between agents. Due to the summation of the agent's embeddings,

only a single message is able to be shared between the agents at a single iteration. In a simulated environment, the simulation can be paused for the model to complete its iterations. However, this approach poses a problem when operated in real-time. MA-BoN alleviates this problem by training two different models for the message/broadcast policy and the behavior policy which are trained in a tandem. The trained broadcast network generates messages which can be inclusive of all agent's personal messages. However, to compensate for the reduce communication iterations, the MA-BoN requires a bigger throughput and thus the broadcast message network's output size is 64 bits (2 bytes).

4.2.2 Hierarchical MA-BoN Model

Similar to the MA-MeSN, the MA-BoN model is also extended to use the hierarchical DQN model as the baseline. The policy network (f'') is the only network which deals with the hierarchical DQN method. The actions A_i are goals which are passed to the lower level DQN. Thus, MA-BoN is easily extendable to a hierarchical structure in terms of the policy network.

4.3 Partially decentralized MA-MeSN and MA-BoN models

The MA-MeSN and MA-BoN approaches are centralized training methods for multi-agent environments. During real-world execution, the message channels are not necessarily available. The literature provides multiple methods for developing a decentralized execution policy. A decentralized approach presented in the literature is to use a probabilistic sampling approach to generate a discrete messages which are passed between agents [34]. This approach reduces the size of the messages shared between agents, but it does not provide a fully decentralized policy. In the following two sections I present methods which allow fully decentralized execution with no message sharing between the agents.

4.4 Cooperative Distributed Behavior Cloning (CoDBC)

Behavior cloning or Imitation learning has been used in the literature to perform inverse reinforcement learning. Inverse reinforcement learning is the concept of learning a policy

directly from observations attained from executing an expert policy. In the scenario of MARL, the centralized policy is the expert policy. The decentralized policy can be trained by imitating the expert centralized policy. The CoDBC policy is trained using behavior cloning. The policy update for CoDBC is calculated by minimizing the following error function using stochastic gradient decent.

$$E(W_{CoDBC}) = \frac{1}{2}\sum_{n} |\pi_{CoDBC}(o_i^t) - \pi_{MA-MeSN}(o_i^t)|^2 \tag{4.1}$$

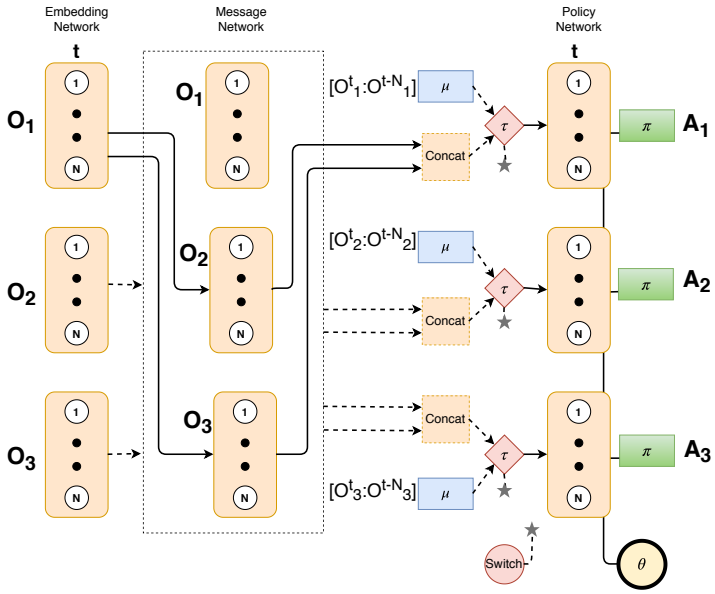
Each agent is trained separately after the centralized policy has converged. After the centralized policy has converged, the final policy is used to generate observations-action pairs for each agent. This data is used in an online manner to train the decentralized CoDBC agents. Using offline training for large scale environments leads to sub-optimal results when using inverse reinforcement learning techniques.

4.5 Decentralized Model using Memory Modules (MM)

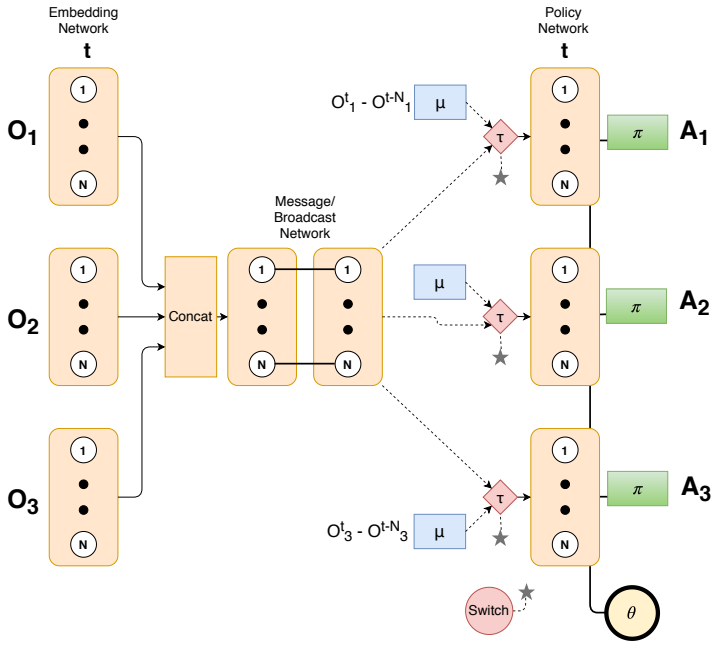
The CoDBC approach needs to be trained after the centralized model has converged. This is because, the behavior cloning equation requires a stationary policy for learning. This leads to extra training time and resources. I present an alternative technique to train a decentralized cooperative policy using memory modules. I utilize a LSTM memory module μ associated with each agent's policy network. The $LSTM_\mu$ learns a mapping from agent's private observation to the message generated by other agents in the environment. The trained memory module is then used during fully decentralized execution to simulate message generation from agents in the environment. The memory module, μ, is trained using temporal supervised learning and can generate messages which are continuous in nature as they don't need to be transported over a communication channel. The following equation shows the error function used to compute the gradients for π_μ.

$$E(W_{MM}) = \frac{1}{2}\sum_{n} |\pi_{MM}(o_i^t ... o_i^{t-N}) - \pi_{f'_{MA-MeSN}}(o_{-\mathbf{i}}^t)|^2 \tag{4.2}$$

The function $\pi_{f'_{MA-MeSN}}$ represents the message network for an agent as shown in Fig. 4.3. The term $\pi_{f'_{MA-MeSN}}(o_{-\mathbf{i}}^t$ term represents the message generated by all other



(a) MA-MeSN with Memory Modules (MA-MeSN-MM)



(b) MA-BoN with Memory Modules (MA-BoN-MM)

Figure 4.3: Architecture of the centralized models with Memory Modules.

agents in the environment. Thus the above equation 4.2 is computing the error between the message generated by other agents in the environment using the function $\pi_{f'_{MA-MeSN}}$ and the prediction of the message generated using the memory module network $\pi_{MM}(o_i^t ... o_i^{t-N})$ using this history of observations for the current agent. Essentially the MM learns the functional mapping from an agent's observation history to the message generated by the other agents.

The learning process for MM sounds counter intuitive due to the fact that the observations and actions (messages generated by the speaker agents) do not always correlate. However, in the scenario of autonomous driving, an agent can view the state transitions of other agents using its own observation. The behavior action of an agent can thus be viewed as model of signalling games (talking by doing). Talking by doing allows an agent to communicate its actions without an explicit message being shared between the agents. This information sharing is prominent in driving as an agent's movement mostly coincides with their intentions. Thus the memory module is able to learn a message generation policy which in tandem with the behavior policy of the current agent allows for a decentralized cooperative policy.

The model architecture is presented in Fig. 4.3. During training of the centralized policy, the module τ switches the input to the policy network between the output of μ module and the *Concat* module. This approach allows the policy network to generalize on both types of messages (the messages can be from other agents or simulated using the memory module μ). During decentralized execution, the communication channels between the agents are snipped and the τ module permanently switches to using the simulated messages from the memory module μ. 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).

Chapter 5

Results for Scalable Multi-Agent Reinforcement Learning Algorithms

In this chapter, I present the results and comparison of the algorithms presented in the previous chapter. This chapter first presents the multi-agent environments which were tested as part of this work. Then I present the results of using the methods presented in this thesis and compare them to the current state of the art methods in literature. I also verify my methods against OpenAI's multi-agent environment for sanity validation of the algorithms. Throughout the chapter the results are discussed and ablation studies are performed on different aspects of the algorithms presented.

5.1 Experimental Methodology

In this section I present the multi-agent treadmill driving environment used for analysis of the proposed algorithms. I also present two environments currently used for multi-agent training in the literature, the well known Predator Prey and Cooperative Communication environments. These environments are released by OpenAI as part of their multi-agent reinforcement learning research projects [28].

The goal of this work was to develop a multi-agent learning algorithm for large scale multi-agent environment. The multi-agent particle environments, currently used for MARL algorithm evaluations, do not provide us with a domain with complex physics, real-world safety constraints and a large state-space for algorithm evaluation. Thus, the following subsection details the highway driving environment using a treadmill simulator which is

used for evaluation of the algorithms presented. This environment could be considered as the baseline for future algorithm comparisons because the multi-agent particle environments have been mostly solved by current MARL research. I show in Sec. 5.2.1, 5.2.2 that all algorithms are all able to learn an optimal policy in an efficient manner in the particle environments, but DIAL and IMS fail to efficiently learn a cooperative policy in the large scale treadmill driving environment.

5.1.1 Treadmill Driving Environment

The main focus of this thesis is to develop multi-agent reinforcement learning algorithms which can be used to train multiple robots to drive in harmony on a treadmill. The software architecture of the treadmill includes a ROS based architecture which uses a publisher/subscriber model to communicate with agents. The same model is used to achieve communication between agents. The information shared between agents is limited to 2 bytes per step for every robot. The Fig. 5.1 shows the treadmill with robots driving on it. Each robot has a tag at the top for identification. A camera mounted at the top of the treadmill computes the location of the robots. This information is then transformed into a proximity based observation (which resembles data retrieved from proximity sensors) and provided to the learning agents during training and execution.

The actual training was not performed on the treadmill, rather a VREP simulator was implemented which mimics the treadmill environment with multiple agents. The VREP simulator is then wrapped with the OpenAI's Gym APIs to enable step based execution of the treadmill environment. In the environment, the treadmill is always running and thus creates an infinite highway for the robots maneuvering in the environment. The size of the treadmill is 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 the number of cooperative agents can be changed dynamically. These agents can be controlled using Deep MARL methods. The environment also contains at least one adversarial (aggressive) car. The adversary exhibits a stochastic behavior policy which tries to 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 from the environment to distinguish between their intentions (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. The agents' actions include 3 angles of steering in both directions and 3 discrete levels of

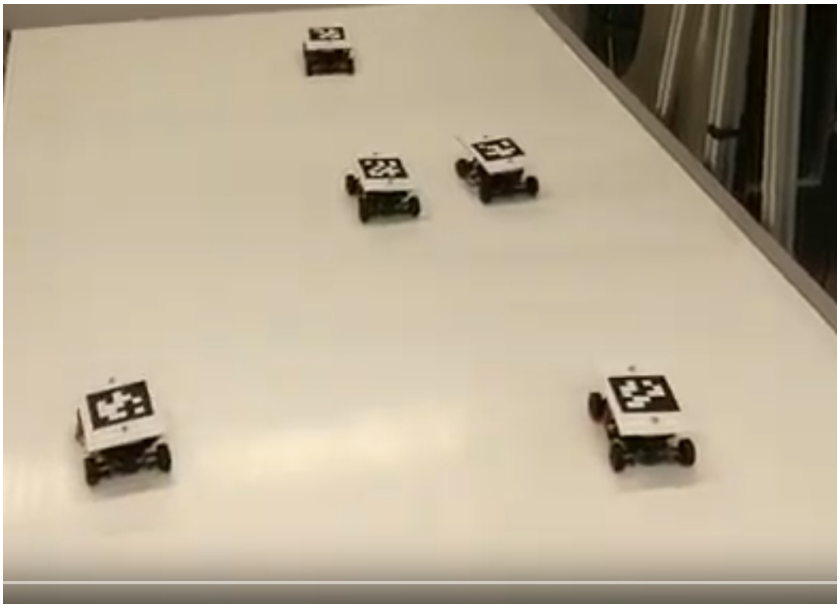


Figure 5.1: Treadmill with multiple robots driving.

acceleration/deceleration. The reward function does not provide explicit rewards for co-operation 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. The observed state space of the environment is 10^{12}.

5.1.2 Predator Prey Environment

The environment in this and the next subsection are set up with a sparse cooperative reward structure with a long time-to-horizon to show the validity of the porposed centralized algorithms (MA-MeSN, MA-BoN) in such domains. 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.

5.1.3 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 episode ends after the agent reaches one of the landmarks. 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.

5.2 Results - Particle Environments

In this section, I present the results of training my algorithms on predator prey and cooperative communication environments. This result is for validation of the algorithms presented in this thesis on multi-agent environments which have been accepted by the MARL community.

5.2.1 Centralized Training - Predator Prey

The results of MA-MeSN algorithm on the multi-agent predator prey environment are shown in Fig. 5.2. 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 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.

The centralized training methods, MA-MeSN and MA-BoN, was able to achieve good performance in this environment. DIAL and IMS are also able to achieve a better performance compared to independent agents which achieve an average reward of 0.75.

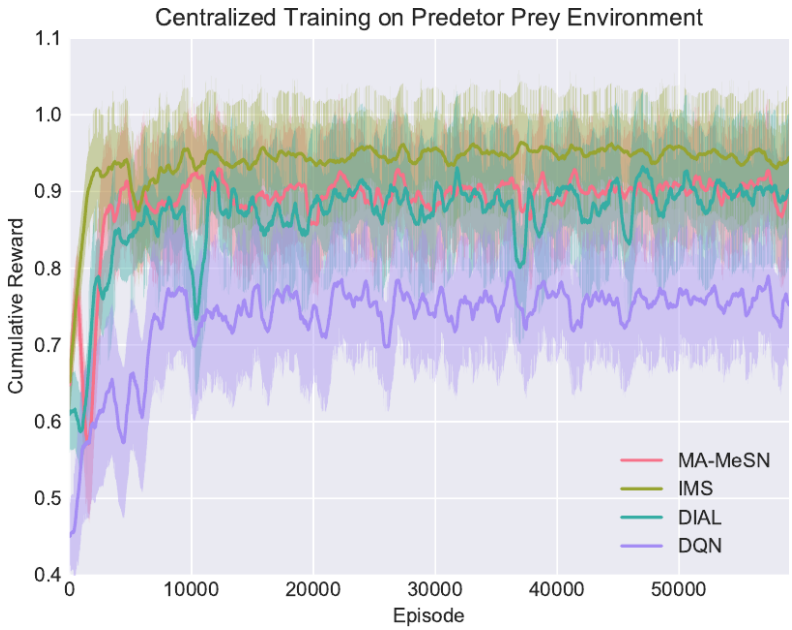


Figure 5.2: Comparison of Cumulative Reward on Predator Prey Environment.

Note, there is a slight variation in my implementation of DIAL. I employed importance sampling from the replay buffer for DIAL as DIAL's original proposal of per-episode batch training is expensive and unstable. This change actually improved DIAL's performance on long time-to-horizon, sparse reward tasks. The results clearly show that the algorithm (MA-MeSN) proposed in this thesis are acceptable on environments with sparse cooperative rewards. This experiment also validates that the algorithms presented in this thesis are valid for MARL environments.

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 average value of 0.7 achieved by the independent agents aligns with this analysis as well. 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, I find that we require $P = 3$ communication iterations for IMS with 2 predators.

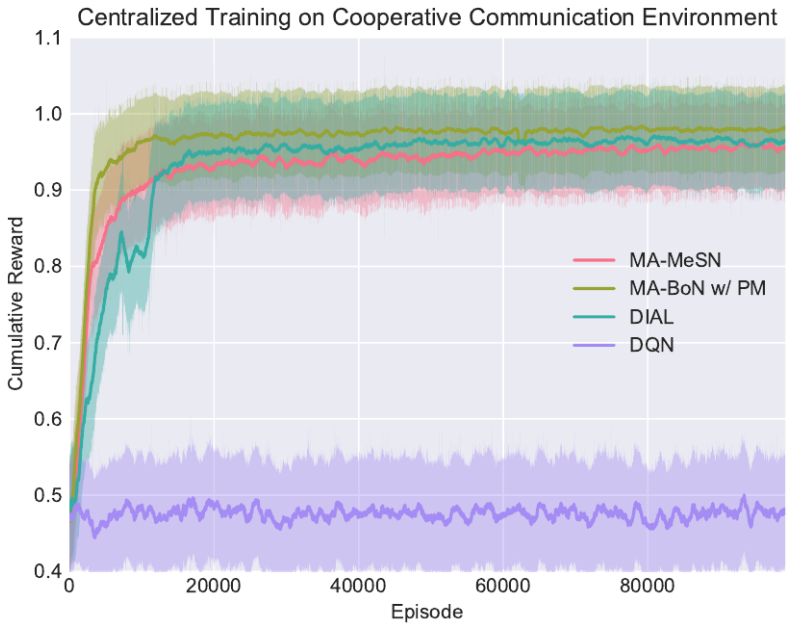


Figure 5.3: Comparison of Cumulative Reward on Cooperative Communication Environment.

5.2.2 Centralized Training - Cooperative Communication

The result of training on the cooperative communication environment is shown in Fig. 5.3. 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, I apply a softmax on the output of the speaker before sending it to the listener. The learning curves for MA-MeSN, MA-BoN and DIAL are compared against independent agents. MA-BoN and MA-MeSN don't use parameter sharing and the communication channel from the listener to the speaker is disabled.

The main focus of the training in this environment is to establish a one-way communication using only the reward received by the listener. The models backpropagate the partial gradients computed using the temporal difference at the listener side during training. This approach helps the speaker agent to optimize its messages based on the observation of the environment. All centralized training algorithms are able to converge to a nearly optimal policy. The independent DQN policy picks a random location due to the lack of communication from the speaker (devoid of the knowledge of the correct color of landmark to reach)

and thus achieves an average reward of 0.5.

5.3 Results for Centralized Training - Treadmill Driving Environment

In this section, I present the results and discussion for the centralized training algorithms, MA-MeSN and MA-BoN. In all the algorithms tested on the treadmill driving environment (MA-MeSN, MA-BoN, DIAL, IMS, independent DQN, independent DQN with SER, MA-MeSN-MM, MA-BoN-MM, CoDBC), I use a hierarchical neural network structure [23]. In this section, I present the learning curve for my algorithms on the treadmill driving environment. I also present an ablation study of MA-BoN algorithm. Then I study the inter-agent communication messages generated by the MA-MeSN algorithm and how they affect the final cooperative policy of the MARL system.

5.3.1 Hierarchical DQN in Single-Agent Treadmill Environment

Hierarchical DQN [23] extends DQN by using prior knowledge of the environment. It is a framework that integrates hierarchical value functions operating at different temporal or logical 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.

For treadmill driving environment, I find that using hierarchical DQN shows improved performance and time complexity when compared to DQN. Fig. 5.4 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 [23] along with the Double Q-learning update [45].

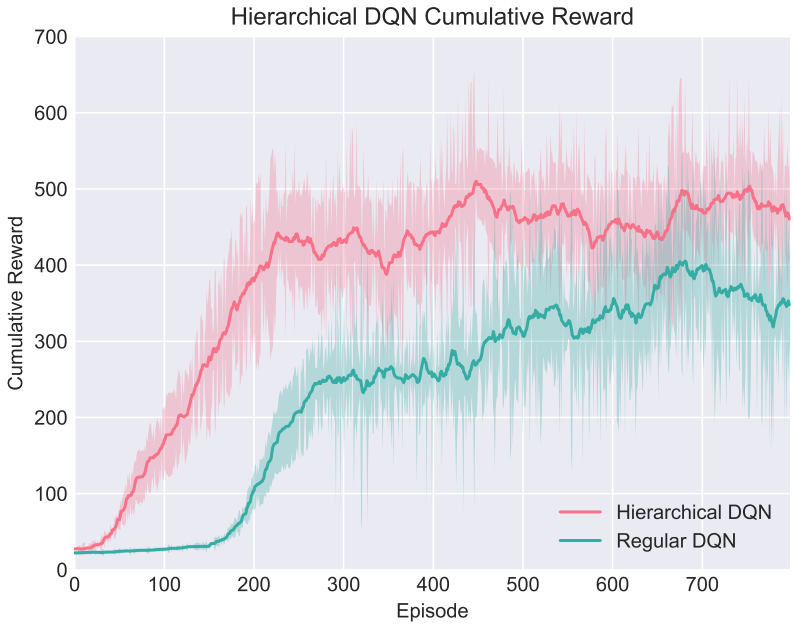


Figure 5.4: Comparison of Hierarchical DQN vs Regular DQN on driving environment with single autonomous agent.

5.3.2 Centralized training on multi-agent driving environment.

In this section, I compare the centralized training algorithms against DIAL and IMS. In all the experiments, the exploration is set to a ϵ-greedy policy with ϵ linearly decayed from 1.0 to 0.05 over the first $100K$ steps and then set to a constant value of 0.05. All experiments are run for a maximum of $4K$ episodes ($0.8M$ steps). All neural networks consist of two layers with 4096 neural units in the first layer. DIAL network consists of two layers with 6144 units in the first layer to allow for fair evaluation to other algorithms. The size of message sharing channel is set to 12 by using the second fully-connected hidden layer. The optimizer used for training is the 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, the experiments are using $P = 5$ for communication iterations through cross-validation; where P represents the number of iterations of message sharing performed at a single environment time-step.

The results for centralized training of cooperative multi-agents are shown in Fig. 5.5. All experiments are repeated 20 times and averaged to produce the learning curves. The

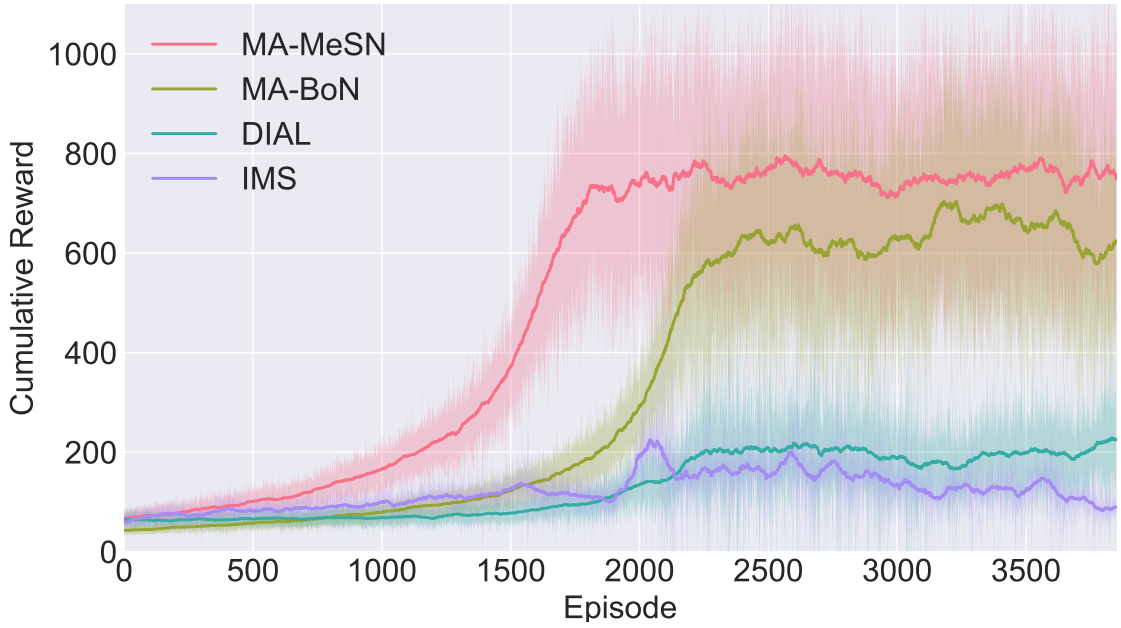


Figure 5.5: Comparison of Cumulative Reward for Centralized Training Algorithms in Driving Environment.

highest cumulative reward is achieved 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 [39]. However, to maintain fairness to other algorithms, this was left out in the experiments. Curriculum learning provides a framework for pre-training the model on simpler tasks before fine-tuning the model in the real environment.

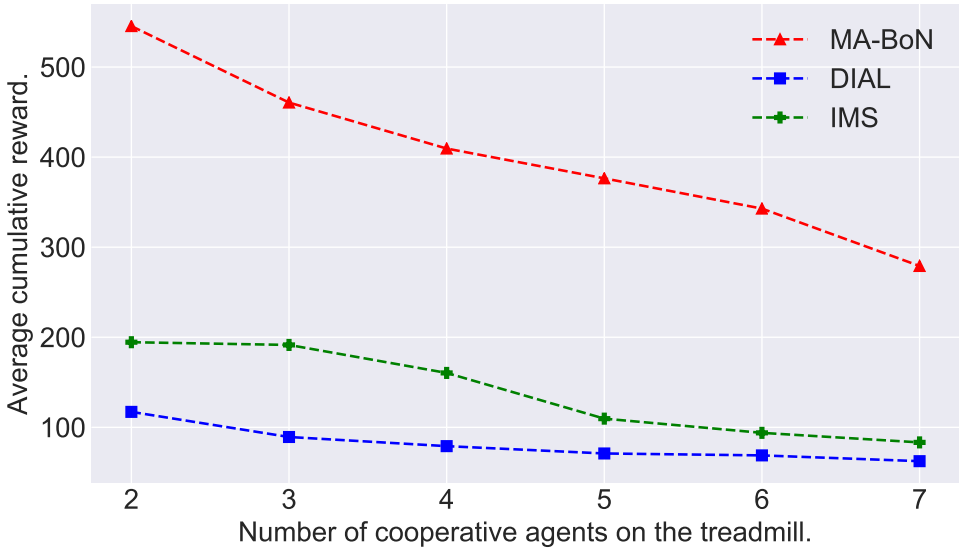
DIAL shows steady improvement in performance, however, the performance of the final policy is weak when compared to MA-MeSN, because the messages from other agents are not conditioned on the present state of the environment m_i^{t-1}. To avoid divergence in MARL, DIAL uses episodic training, which causes skewed gradient updates based on previously observed data and can hinder convergence in tasks with large state space (treadmill environment state space:10^{12}). Secondly, DIAL uses the same model for message policy and behavior policy prediction. This is a viable approach in static multi-agent environments because the agents only policy is to share messages and don't need to learn a behavior policy. MA-MeSN and MA-BoN on the other hand show strong performance when compared

to DIAL. This increase in performance can be attributed to the design of the message sharing model where MA-MeSN and MA-BoN have a designated message network which is only responsible for generating messages to be sent to other agents. The agents use the policy network to generate the behavior actions for the environment. The benefits of this method and its validity is studied in depth in Sec. 5.3.4.

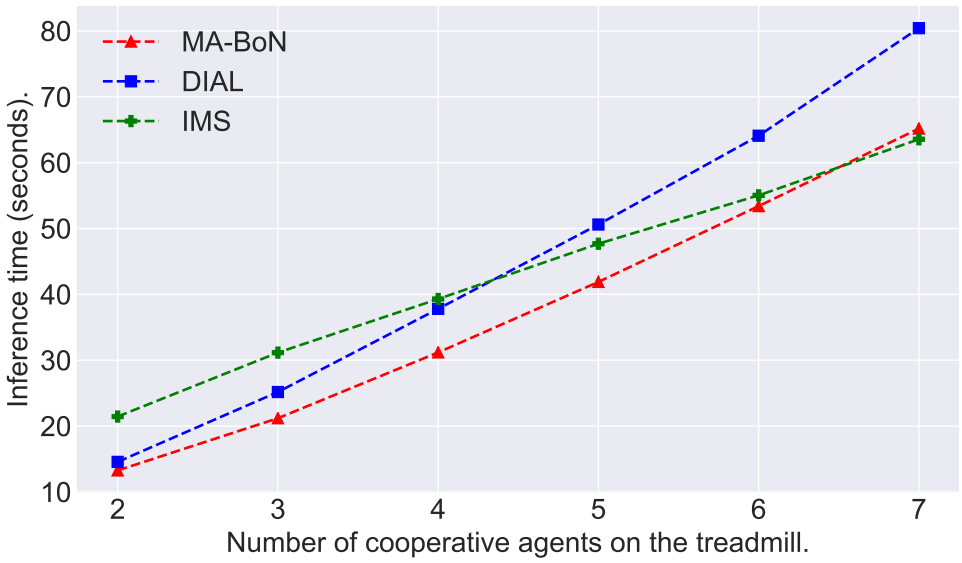
The MA-BoN and MA-MeSN use step-based replay memory (z_i^t, a_i^t, r_i^t) which provides diverse training samples for offline updates to the value function. This is a big advantage over DIAL during training as it reduces the system resource requirements and requires simpler algorithms to manage the training. MA-MeSN and MA-BoN thus achieve a stable learning curve with faster convergence properties than DIAL and IMS [32]. The MA-BoN results show comparative performance to MA-MeSN and provide the benefit of reduction in the number of communication layers needed from $|N| \times |N|$ to $|N|$. MA-BoN also allows for reduced inference time as policies for all agents can be evaluated in parallel, as opposed to $|N|$ serial evaluations for DIAL, IMS and MA-MeSN. This is further studied as part of scalability ablation study in Sec. 5.3.3.

5.3.3 Ablation Study of scalability of MA-BoN

This section demonstrates the scalability of the MA-BoN approach compared to IMS and DIAL. The ablation study of the MA-BoN method is carried out by varying the number of cars in the environment and presenting the results in Fig. 5.6. The Fig. 5.6b shows a comparison of the inference time it took to complete an episode and the Fig. 5.6a shows 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. The proposed centralized training approach of MA-BoN is able to sustain better performance compared to other approaches when the complexity of the environment is 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 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.



(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.

Figure 5.6: Scalability comparison on the treadmill environment.

5.3.4 Theoretical study of Emergent Communication

The results obtained from generating the learning curve in Fig. 5.5 show a positive correlation between message sharing and the eventual performance achieved by the centralized cooperative policy. To study the effects of the message sharing between agents, I provide quantitative results for the message policy and how it effects the behavior policy. In this section, I study the emergent communication developed during training of the centralized MARL algorithm, MA-MeSN using the techniques presented in this paper [27]. The work done in [27] shows that a better cooperative policy between agents could be achieved without explicit need for message sharing between the agents. This can happen in environments which either present a static environment or a stationary environment from the view of the speaker. This leads to the agents learning an independent policy which treats the messages received from other agents as cheap talk. In such situations, the messages shared between the agents are not useful. The second observation made in [27] is that the inter-agent communication could arise due to weight sharing between the policy network and the message network. They show that agents which use a single model for predicting the behavior policy and the message policy can lead to an positive signaling but doesn't necessarily improve the performance of the agents.

Each agent generates a message action (for communication between agents) and a behavior action (this can be an action in the environment which can be the agent's dynamics in the environment). Table 5.1 shows the results for MA-MeSN using common metrics [27] to measure the effect of these messages in our domain. There are multiple metrics used in the table.

Speaker Consistency (SC) is used to measure positive signaling as it measures the mutual information between the message m_i and the actions a_i of the agent. The mutual information between the messages and the actions is used to measure the effect the messages have on the trained policy of the MA-MeSN algorithm. We can see a slight positive correlation between the communication and environment actions (for the same agent), which indicates that the agents are sharing information which could lead to a cooperative equilibrium. The value of speaker consistency for MA-MeSN is 0.18. This value is lower than other models which share the weights of the message and policy network. In the case of MA-MeSN, the message network is independent of the policy network which leads to the message policy to focus on policy of other agents instead of its own, as the message is not consumed by the agent itself. The message is required by other agents to learn about the future actions of the speaker agent. Thus, I believe this metric does not necessarily provide a complete picture of the agent's communication policy.

The positive signaling metric also does not guarantee that other agents are listening to

Table 5.1: Summary of emergent communication metrics for MA-MeSN: speaker consistency(SC), instantaneous coordination(IC), entropy(H), Message Input Norm(MIN) and Avg. cumulative reward with white noise messages(Δr) [27].

Metric	SC	IC	H	MIN	Δr
Value	0.18	0.41	1.27	63.75	427.1

the message and actively utilizing it to achieve a cooperative behavior policy by achieving a higher cumulative reward. The metric *Instantaneous Coordination (IC)* can be used to measure the positive listening between agents. Instantaneous Coordination is computed using the mutual information between the speaker's communicative actions $m_{-\mathbf{i}}$ and the listener's environment actions a_i. MA-MeSN achieves a value of 0.41 for IC which indicates that the listener agent's policy is dependent on the messages of the speaker. This higher value of the *Instantaneous Coordination (IC)* is more valuable for measuring inter-agent communication than *Speaker Consistency (SC)* because it focuses on the eventual behavior policy of the agents (dependent on the speaker's message) to measure the effect of the inter-agent communication. Instead, speaker consistency only focuses on a single agent's correlation between its message policy and behavior policy. In cooperative environments, the agents would convey information which is useful to other agents, but this message might not align with the agent's eventual actions in the environment. This is defined as a key distinction in the communication model of multi-agent systems [37] and discussed in the background chapter of this thesis. The speaker consistency metric focuses on finding positive correlation between the speech-act theory ("doing by talking") and signaling games ("talking by doing"). In environments like autonomous driving a high value of speaker consistency is sometimes not needed or necessary to achieve optimal centralized cooperative policy.

The metric *Message Input Norm(MIN)* is used to compute the *L2-norm* of the weights of the message input layer of the listener. The MIN metric helps understand if the listener agent's model is ignoring the message it receives from the speaker agent. This verification is done to further study if the positive correlation found with the *IC* metric is not a coincidence. A large number supports the theory that the listener's behavior policy is dependent on the message. The MA-MeSN model exhibits a value of 63.75 which shows that the listener agent is not actively ignoring the message received and is a factor in the final behavior policy of the listener agent.

The Δr metric measures the difference in the average cumulative reward when the messages are replaced with white noise. This metric also allows us to evaluate the listener agent's policy's dependence on the speaker agent's messages. There is a large drop in the performance, suggesting that the agents have indeed developed emergent communication. The performance drops from an average cumulative reward of 750 to an average cumulative reward of 427.1. This metric is further proof that the inter-agent communication is a necessary part of the final cooperative policy.

The metric *Communication Message Entropy* measures the consistency of the messages when compared to the same input. The MA-MeSN model achieve a low 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. Based on the strong results of *IC*, *MIN*, Δr and entropy, the lower value of *SC* is less concerning as the model's inter-agent communication analysis supports the model's learning curve performance.

5.4 Results for Decentralized Execution - Treadmill Driving Environment

In this section, I present the results for decentralized training and execution in the multi-agent treadmill driving environment. The Fig. 5.8 shows the learning curves for the CoDBC and MM methods presented in this thesis and its comparison to the decentralized state of the art methods in the literature.

5.4.1 CoDBC - Results

In this sub-section, I compare the CoDBC method of fully decentralized execution, to the techniques like DQN with stabilized experience replay (SER), and Independent DQN. 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 curves are generated by averaging the results of 20 experiments with different seed value. The agents are trained in a sequential manner. First the centralized model MA-MeSN is trained using the method described in Sec. 4.1. Each agent in the environment is replaced with an independent agent. Essentially the centralized model for a single agent presented in Fig. 4.1 is substituted for a single neural network which does policy predictions based on the partial observation from the environment (this model doesn't receive any messages

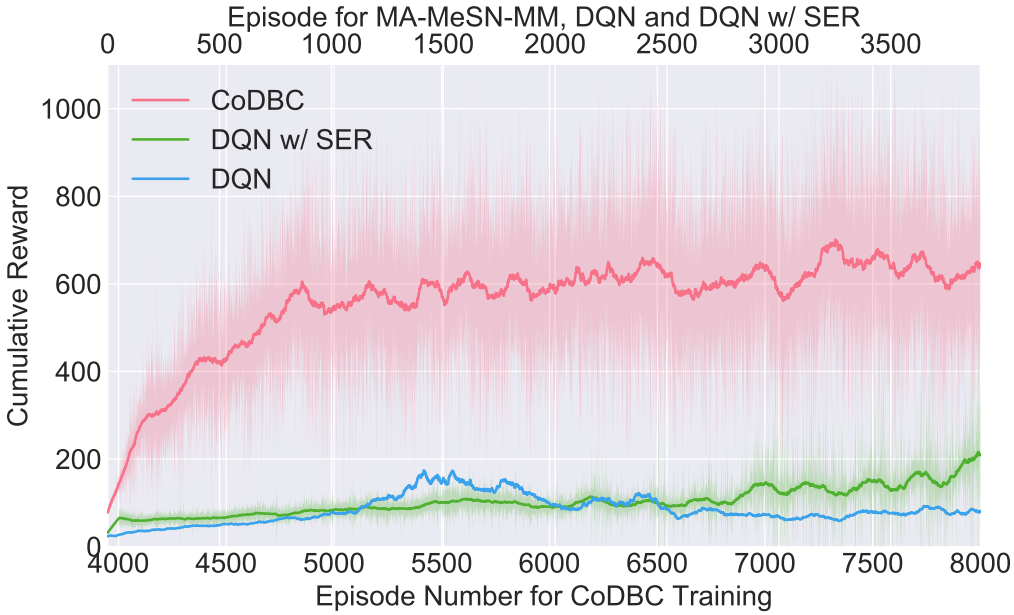


Figure 5.7: Comparison of Cumulative Reward for Decentralized Training in Multi-Agent Driving Environment.

from the speaker agents). The other agents in the environment still need a message from the independent agent. The message is generated by using the original centralized model for the replaced agent. This method allows for other agents to behave with a cooperative centralized policy and allows for the decentralized policy to be trained in a cooperative manner. The main idea of training each agent in a sequential manner is that the policy of all agents in the multi-agent environment has now converged and is thus stationary. Traditional RL methods can be used to train the single agent independent policy in the multi-agent environment now as the transition dynamics won't lead to divergence in the temporal difference updates of Q-learning. Independent policies for all agents are thus trained and the results are shared in Fig. 5.7.

The Fig. 5.7 shows the comparison of the CoDBC method to other independent approaches, for ex, independent DQN, DQN with SER (statbalized experience replay). The performance for DQN is poor compared to other decentralized techniques. This is because the treadmill environment does not explicitly reward agents for cooperation. Independent DQN agents can only converge to a cooperative policy in multi-agent systems if the system itself provides feedback for achieving cooperation. The reward function in the treadmill environment only allows the agents to learn to avoid crashes. This information is not

enough for agents to maintain safe distance from other friendly agents. Due to the lack of signalling between agents, the independent DQN policy is also not able to distinguish between cooperative agents and adversarial agents. This distinction can be achieved either with signaling between agents (through their actions or via information sharing) or directly from the environment. The independent DQN is not able to gather this information from either of the above mentioned sources. The learning curve for DQN shows good performance initially when all agents are exploring and the dynamics of the environment are perceived as stationary. However, as the exploration reduces to 0.05, the evolving policies of other agents induces non-stationarity in the environment model and the off-policy training in DQN fails.

Deep Q-Networks use an experience buffer from which samples are replayed to stabilized training in single agent environments. This approach as seen previously doesn't extend well to multi-agent environments. The issue is with the evolving policies of other agents which exhibit a non-stationary transition matrix for the environment. *DQN-SER* (DQN with Stabilized Experience Replay) [14] focuses on the evolving policies of the other agents in the off-policy training regime of DQN as the problem which needs to be solved. *DQN-SER* thus implements different stabilizing methods for the experience replay. The method implements an importance sampling procedure where the most recent samples are over-sampled compared to samples from the past. This is done by computing the weight of each sample's gradient using a linearly decaying function based on the episodes elapsed since a sample was collected. Thus, *DQN-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. Due to the recency bias in the training of *DQN-SER*, the transition matrix only focuses on the latest transitions in the environment and the non-stationarity is reduced in the training of multiple agents. The learning curve of *DQN-SER* shows a steady improvement for 4000 episodes. The agents are able to avoid divergence and improve their cooperation without any form of message sharing. However, as seen in the Fig. 5.7, the final policy for CoDBC is able to outperform both independent DQN and *DQN-SER*.

5.4.2 Memory Modules - Results

While the CoDBC method outperforms DQN and *DQN-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. The memory modules provide a better training procedure than the cooperative decentralized behavior cloning method (CoDBC). Behavior cloning requires the agents to converge to a stationary policy which

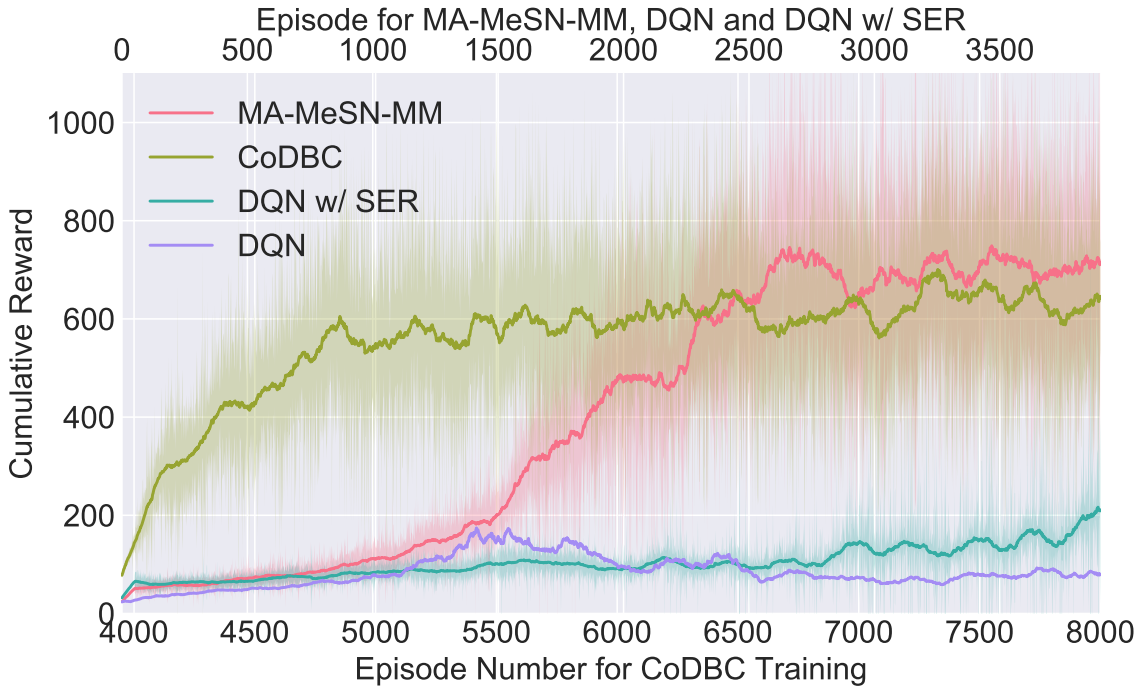


Figure 5.8: Comparison of Cumulative Reward for Decentralized Training in Multi-Agent Driving Environment.

could be replicated using an independent model. The model is also not able to converge to 100% of the performance of the centralized MA-MeSN model. The memory module presents a solution to the sequential problem and achieves a better cumulative reward than the CoDBC approach. The cumulative reward achieved during training of memory modules using the MA-MeSN-MM approach is shown in Fig. 5.8.

The method MA-MeSN-MM achieves decentralized cooperative policy by learning a function mapping from private observations to the messages received from other agents. This is different from the CoDBC approach as the independent agents in CoDBC learn a policy from the agent's observation to the centralized agent's behavior policy. The CoDBC approach thus is missing the communication from other agents to help in its training. In memory module, the neural network μ is trained to map the observation history of other agents to the message received from the agents. This assists the memory module to infer the policy of an agent as cooperative or adversary and in-turn predict the message which would be received from the cooperative agent. This approach also provides the benefit of having different models for message policy and behavior policy and does need altering of the behavior policy trained during centralized training in MA-MeSN. The same model f''

is used for behavior policy predictions. The only difference arises in the messages which are being fed to the policy network f''.

The cumulative reward curve shown in Fig. 5.8 is computed by switching the τ function to select the predicted messages using the μ model, as shown in Fig. 4.3. The memory module (MM) is trained in parallel to the policy network and thus does not require additional training after MA-MeSN has converged. The Fig. 5.8 shows that the MA-MeSN-MM approach is able to out-perform all other decentralized approaches and doesn't need to be trained in a sequential manner. 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.

Chapter 6

Conclusion and Future Work

In this thesis I proposed two novel methods for centralized training and two novel methods for decentralized training. I also present a novel multi-agent training simulator for autonomous driving on a highway in the presence of an adversary agent. The thesis presents the results of training the centralized and decentralized methods on the treadmill driving environment.

6.1 Conclusion

The main focus of this thesis was to present scalable multi-agent approaches which could be used to train in multi-agent autonomous driving environments. The first approach presented in this thesis is the MA-MeSN centralized training method (Multi-Agent Message Sharing Network). The algorithm provides an inter-agent learning algorithm when a centralized message sharing location is not available. This approach achieves the best performance in the scalable environment of multi-agent treadmill driving environment.

The second centralized method presented in this thesis is the MA-BoN method which is a specialization of the MA-MeSN method for environments which allow for a centralized message sharing location. The MA-BoN also achieves nearly the same average cumulative reward at convergence time as MA-MeSN. The slow learning curve for MA-BoN represents the restrictions on the learning process due to a single message being shared between the agents.

In this thesis, I presented techniques for training and execution of a cooperative policy which is shared among multiple agents. For the execution of the decentralized policy,

I developed and presented two novel methods, CoDBC and MM. The Cooperative Distributed Behavior Cloning (CoDBC) algorithm uses regular behavior cloning to clone the centralized policy into a decentralized policy. The Memory Module (MM) algorithms, like MA-MeSN-MM and MA-BoN-MM, show that predicting the messages from other agents based on observation history can lead to better decentralized policy.

In this thesis, the multi-agent simulator and OpenAI Gym Environment for the treadmill driving environment were developed using python. This environment included basic physics of driving and adversary agents to mimic driving on a highway. Using the approaches presented in this thesis, the message sharing between agents is only required during the training process.

The thesis also verified the algorithms on OpenAI's multi-agent particle environments which are based on multiple game theoretic models of multi-agent scenarios. The algorithms presented in this thesis were able to perform appropriately in these environments and achieve a good performance when compared to the optimal policy.

6.2 Future Work

There are many directions I want to further explore this work. The main motivation of this thesis was to develop a training algorithm which can work with multiple agents for autonomous driving. However, there can be situations where the messages between agents are not received for a certain time period or a new agent joins the group or leaves the group. A future work for this thesis is to develop an algorithm which can be trained with variable number of agents in the environment and maintain a varying policy depending on the number of robots in the environment.

I have done work in this direction. A simple approach for variable number of agents is to use safety regularization during policy training. The safety regularization can be a penalty term which depends on the number of the agents. The transfer learning approaches in reinforcement learning can also be used to train agents when there is variability in the environment.

References

[1] Pieter Abbeel and Andrew Y. Ng. Apprenticeship learning via inverse reinforcement learning. pages 1–, 2004.

[2] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

[3] Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4):819–840, 2002.

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

[5] Tim Brys, Anna Harutyunyan, Halit Bener Suay, Sonia Chernova, Matthew E Taylor, and Ann Nowé. Reinforcement learning from demonstration through shaping. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015.

[6] 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), 2008*, 2008.

[7] A. Das, S. Kottur, J. M. F. Moura, S. Lee, and D. Batra. Learning cooperative visual dialog agents with deep reinforcement learning. pages 2970–2979, Oct 2017.

[8] Abhishek Das, Satwik Kottur, José MF Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning. *arXiv preprint arXiv:1703.06585*, 2017.

[9] Javier de Lope et al. Learning autonomous helicopter flight with evolutionary reinforcement learning. In *International Conference on Computer Aided Systems Theory*, pages 75–82. Springer, 2009.

[10] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 2137–2145, 2016.

[11] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130. International Foundation for Autonomous Agents and Multiagent Systems, 2018.

[12] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *AAAI 2018: Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, February 2018.

[13] Jakob Foerster, Gregory Farquhar, Maruan Al-Shedivat, Tim Rocktäschel, Eric P Xing, and Shimon Whiteson. Dice: The infinitely differentiable monte-carlo estimator. *arXiv preprint arXiv:1802.05098*, 2018.

[14] Jakob Foerster, Nantas Nardelli, Greg Farquhar, Phil Torr, Pushmeet Kohli, and Shimon Whiteson. Stabilising experience replay for deep multi-agent reinforcement learning. In *ICML 2017: Proceedings of the Thirty-Fourth International Conference on Machine Learning*, June 2017.

[15] Peter Geibel and Fritz Wysotzki. Risk-sensitive reinforcement learning applied to control under constraints. *Journal of Artificial Intelligence Research*, 24:81–108, 2005.

[16] Alborz Geramifard, Joshua Redding, and Jonathan P How. Intelligent cooperative control architecture: a framework for performance improvement using safe learning. *Journal of Intelligent & Robotic Systems*, 72(1):83–103, 2013.

[17] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 66–83. Springer, 2017.

[18] Hado V Hasselt. Double q-learning. In *Advances in Neural Information Processing Systems*, pages 2613–2621, 2010.

[19] Matthias Heger. Consideration of risk in reinforcement learning. In *Machine Learning Proceedings 1994*, pages 105–111. Elsevier, 1994.

[20] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[21] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

[22] Landon Kraemer and Bikramjit Banerjee. Multi-agent reinforcement learning as a rehearsal for decentralized planning. *Neurocomputing*, 190:82–94, 2016.

[23] Tejas D Kulkarni, Karthik Narasimhan, Ardavan Saeedi, and Josh Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[24] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni. Multi-agent cooperation and the emergence of (natural) language. *arXiv preprint arXiv:1612.07182*, 2016.

[25] 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*, pages 464–473. International Foundation for Autonomous Agents and Multiagent Systems, 2017.

[26] 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.

[27] Ryan Lowe, Jakob Foerster, Y-Lan Boureau, Joelle Pineau, and Yann Dauphin. On the pitfalls of measuring emergent communication. *arXiv preprint arXiv:1903.05168*, 2019.

[28] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems*, pages 6379–6390, 2017.

[29] Ryan Lowe, YI WU, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. pages 6379–6390, 2017.

[30] Oliver Mihatsch and Ralph Neuneier. Risk-sensitive reinforcement learning. *Machine learning*, 49(2-3):267–290, 2002.

[31] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937, 2016.

[32] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

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

[34] Igor Mordatch and Pieter Abbeel. Emergence of grounded compositional language in multi-agent populations. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[35] Dean A Pomerleau. Efficient training of artificial neural networks for autonomous navigation. *Neural Computation*, 3(1):88–97, 1991.

[36] Makoto Sato, Hajime Kimura, and Shibenobu Kobayashi. Td algorithm for the variance of return and mean-variance reinforcement learning. *Transactions of the Japanese Society for Artificial Intelligence*, 16(3):353–362, 2001.

[37] Yoav Shoham and Kevin Leyton-Brown. *Multiagent systems: Algorithmic, game-theoretic, and logical foundations*. Cambridge University Press, 2008.

[38] Yong Song, Yi-bin Li, Cai-hong Li, and Gui-fang Zhang. An efficient initialization approach of q-learning for mobile robots. *International Journal of Control, Automation and Systems*, 10(1):166–172, 2012.

[39] 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.

[40] 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, 2017.

[41] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pages 330–337, 1993.

[42] Matthew E Taylor and AI Borealis. Improving reinforcement learning with human input. pages 5724–5728, 2018.

[43] Gerald Tesauro. Extending q-learning to general adaptive multi-agent systems. In *Advances in neural information processing systems*, pages 871–878, 2004.

[44] Nicolas Usunier, Gabriel Synnaeve, Zeming Lin, and Soumith Chintala. Episodic exploration for deep deterministic policies: An application to starcraft micromanagement tasks. *arXiv preprint arXiv:1609.02993*, 2016.

[45] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *AAAI*, volume 2, page 5. Phoenix, AZ, 2016.