Model-based Sparse Communication in Multi-agent Reinforcement Learning

Shuai Han
Utrecht University
Utrecht, the Netherland
s.han@uu.nl

Mehdi Dastani
Utrecht University
Utrecht, the Netherland
m.m.dastani@uu.nl

Shihan Wang
Utrecht University
Utrecht, the Netherland
s.wang2@uu.nl

Abstract

Learning to communicate efficiently is central to multi-agent reinforcement learning (MARL). Existing methods often require agents to exchange messages intensively, which abuses communication channels and leads to high communication overhead. Only a few methods target on learning sparse communication, but they allow limited information to be shared, which affects the efficiency of policy learning. In this work, we propose model-based communication (MBC), a learning framework with a decentralized communication scheduling process. The MBC framework enables multiple agents to make decisions with sparse communication. In particular, the MBC framework introduces a model-based message estimator to estimate the up-to-date global messages using past local data. A decentralized message scheduling mechanism is also proposed to determine whether a message shall be sent based on the estimation. We evaluated our method in a variety of mixed cooperative-competitive environments. The experiment results show that the MBC method shows better performance and lower channel overhead than the state-of-art baselines.

1 Introduction

Multi-agent reinforcement learning (MARL) provides powerful approaches for agents to develop effective cooperative and competitive policies. Recently these approaches have been applied in a variety of complex environments, such as traffic light control Wang et al. [2022], robotics Gu et al. [2017] and autonomous driving Wachi [2019]. Communication allows agents to share observations and intentions, thus greatly improving the efficiency and success rate for completing specific tasks Liu et al. [2020], Singh et al. [2019], Niu et al. [2021]. In communication MARL, agents communicate with each other before taking action and learn message encoding and decoding protocol with back-propagation Sukhbaatar et al. [2016]. In addition to encoding and decoding messages, agents need to learn 'when' and 'whom' to send their messages. This is known as communication scheduling Kim et al. [2019], Niu et al. [2021]. For example, IC3Net Singh et al. [2019] learns when to broadcast messages, and I2C Ding et al. [2020] and ACML Mao et al. [2020] introduce gate mechanisms to decide whether to send messages to specific agents. Moreover, FlowComm Du et al. [2021] and MAGIC Niu et al. [2021] learn to dynamically generate communication graphs to schedule messages.

However, most communication MARL methods require intensive communication among agents, which leads to high communication overhead. This issue is crucial for the real-world application of MARL methods where communication is costly Zhang and Lesser [2013]. For example, in some practical scenarios, such as unmanned aerial vehicles, reducing communication overhead is a fundamental concern due to the low-power property of the sensors Gu et al. [2020]. The existing communication MARL algorithms rarely consider the cost during training and execution, resulting in excessive and redundant communication Du et al. [2021], Zhang et al. [2020, 2019a]. On the other hand, a few existing methods for reducing communication overhead Liu et al. [2020], Zhang et al. [2020], Kim et al. [2019] do not allow agents to utilize enough information about the observations and intentions of other agents, resulting in uncompetitive performance. To our knowledge, reducing communication overhead while enabling agents to use as much information as possible to learn optimal policies is a problem that has rarely been studied.

In order to deal with this problem, we propose a novel framework in communication MARL, which we will call model-based communication (MBC). The basic idea of MBC is to enable agents to utilize the previously exchanged messages to estimate current messages that agents may exchange. The estimated messages, accessible to all individual agents, can be used by individual agents to decide whether it is needed to send a message (in case the message deviates significantly from the estimated message) or if the other agents can use their estimated message. The latter case will reduce communication overhead.

In the MBC framework, this is realized by a message estimator that is designed to be trained in a supervised manner to model the dynamics of global messages, so that agents can estimate the current messages of other agents using their previous messages. In addition, a decentralized message scheduling mechanism is introduced, which eliminates the necessity of additional communication to a central scheduler and is therefore conducive to distributed deployment. According to the scheduling in the MBC framework, each agent will send its messages only when other agents cannot estimate its messages within an error threshold. In this way, MARL agents can correct the estimation error of the global message by sending real messages with each other.

This paper makes the following contributions. 1) We propose to reduce the communication overhead by replacing receiving messages from other agents with estimating others' messages. To the best of our knowledge, this is the first work to use a model-based method to reduce the communication overhead in MARL approaches. 2) We design a decentralized message scheduling mechanism for multi-agents to correct erroneous message estimations by communicating the real messages when they cannot estimate the message accurately. 3) We verified the performance of our approach in a series of mixed cooperative-competitive environments. Our approach shows around 5.16% better performance on average than the state-of-the-art methods, around 22.33% lower communication overhead on average than the previous most communication-efficient method, and less dependence on the number of channels.

2 Related Work

Among the plethora of work on MARL Gronauer and Diepold [2022], Sharma et al. [2021], Zhu et al. [2022], we summarize the literature in three subfields, which our approach closely relates to.

Communication scheduling in MARL. This subfield addresses the problems of when to communicate and whom to address messages to. As the early works, IC3Net Singh et al. [2019] and SchedNet Kim et al. [2019] learn to decide when to broadcast individual messages. Agent-Entity Graph Agarwal et al. [2020] and G2ANet Liu et al. [2020] schedule communication among agents via a pretrained graph. Most recently, FlowComm Du et al. [2021] and MAGIC Niu et al. [2021] learn to dynamically generate graphs to determine when to communicate with whom. Communication scheduling can reduce the communication overhead in multi-agent systems, which is significant for deployment of MARL for real-world scenarios Niu et al. [2021], Liu et al. [2020], Zhang et al. [2020]. In line with these works, our scheduling approach further reduces communication load while maintaining collaboration among agents.

Message aggregation in MARL. The approaches in this area address how to learn to effectively extract information from the received messages. Earlier approaches mostly average Singh et al. [2019], Zhang et al. [2019b], Sukhbaatar et al. [2016], Foerster et al. [2016] or concatenate Kim et al. [2019] the received messages to aggregate them. Some practices, such as pruning the incoming messages Du et al. [2021], Liu et al. [2020] or adding attention mechanisms to them Das et al. [2019], Jiang and Lu [2018], can alleviate this problem. Our approach builds on and adopts the multilayer nonlinear aggregation methods on received messages using the Graph Attention Networks (GATs) Casanova et al. [2018], Niu et al. [2021] or attention mechanisms Das et al. [2019]. In particular, to overcome the problem of numerous input messages, our approach uses GATs to aggregate the information from estimated global messages.

Model-based multi-agent reinforcement learning. Model-based MARL alleviates the issue of sample efficiency in model-free MARL Willemsen et al. [2021], Egorov and Shpilman [2022]. These methods typically model the environment through supervised training. After the model learns to accurately simulate the environment, the agent interacts with this model rather than the environment Willemsen et al. [2021]. Inspired by the previous Model-based MARL approaches, we propose to train a model in a supervised way to estimate the environmental dynamics. In particular, instead of modelling the dynamics of the observations (in previous works), we model the dynamics of communication messages (i.e. encoded observations). To the best of our knowledge, this is the first work to introduce the model-based learning into communication of MARL.

3 Preliminaries

3.1 Markov Game

We follow the partially observable multi-agent Markov Game Littman [1994], Niu et al. [2021] to study the communication in multi-agent reinforcement learning. A partially observable multi-agent Markov Game is defined as a tuple $< \mathcal{N}, \mathcal{S}, \mathcal{A}, \mathcal{T}, \Omega, R, \gamma >$. In this tuple, $\mathcal{N} = \{1, 2, ..., n\}$ is the set of agents, $\mathcal{S}$ is the set of global state, $\mathcal{A} = A_1 \times A_2 \times ... \times A_N$ is the set of joint actions where A_i is the set of possible actions of agent i, $\mathcal{T} : \mathcal{S} \times A_1 \times ... \times A_N \mapsto \mathcal{S}$ is the transition function, $\Omega = \Omega_1 \times \Omega_2 \times ... \times \Omega_n$ is the set of joint observations where Ω_i is the possible observations of agent i, R is the reward function, and $\gamma \in [0, 1]$ is a discounted factor. The trajectory of agent i is represented by $\tau_i^t \in T_i = (\Omega \times \mathcal{A}_i)^* \times \Omega$ where $(\Omega \times \mathcal{A}_i)^*$ represents the Kleene closure on $\Omega \times \mathcal{A}_i$. At each time step t, agent i executes an action $a_i^t \in A_i$ based on its observation $o_i^t \in \Omega_i$, and receives an individual reward r_i. Agent i is dedicated to adjust its policy π_i to maximize the individual rewards: $R_i = \sum_{t=0}^{T} \gamma^t r_i^t$, where T is the terminal time step.

3.2 MARL with Communication

When the above Markov game has been solved using multi-agent reinforcement learning with communication, agent i makes decisions based on individual policy $\pi_i : \Omega_i \times AM_i \times A_i \mapsto [0, 1]$, where AM_i is the set of aggregated messages received by the i-th agent. Individual agent i uses $f_{agg_i} : M_1, M_2, ..., M_N \mapsto AM_i$ to aggregate received messages am_i where M_j represents the set of agent j's possible sending messages. Moreover, agent j who sends message uses $f_{enc_j} : \Omega_j \mapsto M_j$ to encode local observation into message m_j. π_i, f_{agg_i} and f_{enc_j} are jointly parameterized by $\theta_{i,j} = [\theta^{\pi_i}, \theta^{f_{agg_i}}, \theta^{f_{enc_j}}]$, where θ^{π_i} is the parameters of π_i, $\theta^{f_{agg_i}}$ is the parameters of f_{agg_i}, and $\theta^{f_{enc_j}}$ is the parameters of f_{enc_j}.

In this work, we follow the previous policy gradient methods Mnih et al. [2016], Schulman et al. [2017], Niu et al. [2021] to train policy π_i, f_{agg_i} and f_{enc_j} jointly by maximizing the objective $J(\theta_{i,j}) = \mathbb{E}_{s \sim \rho, a_i \sim \pi_i}[R_i^t]$, where ρ is the initial state distribution and $R_i^t = \sum_{t'=t}^{T} \gamma^{t'-t} r_i^{t'}$ is the discounted total rewards of agent i from the time step $t \in [0, T]$. This method performs gradient ascent on $\theta_{i,j}$.

$$\nabla_{\theta_{i,j}} J(\theta_{i,j}) = \mathbb{E}_{s \sim \rho, a_i \sim \pi_i}[\sum_{t=1}^{T} \nabla_{\theta_{i,j}} \log \pi(a_i^t | \tau_i^t, am_i^t) \cdot R_i^t] \tag{1}$$

where $am_i^t = f_{agg_i}(m_1, ..., m_j, ..., m_N)$ and $m_j = f_{enc_j}(o_j^t)$. To reduce the variance, we adopt the advantage function $A_i(\tau_i^t, a_i^t) = R_i^t - V_i(\tau_i^t)$ in place of R_i^t, where V_i is the expected cumulative reward estimated by agent i.

3.3 Centralized Training & Decentralized Execution.

Centralized training & decentralized execution (CTDE) is a common paradigm in MARL Sunehag et al. [2018], Jaques et al. [2019], Papoudakis et al. [2021]. In this paradigm, all global information is available in training, but in execution, agents are only allowed to utilize local or received information. The existing approaches mostly use centralized value function and decentralized policies Rashid et al. [2018], Foerster et al. [2018], Peng et al. [2021]. When it comes to communication, the encoding and decoding of messages are performed decentralized in execution Singh et al. [2019], Ding et al. [2020], Sukhbaatar et al. [2016]. Our approach follows the CTDE paradigm.

4 Methodology

In this section, we describe the detailed design of the Model-based Communication (MBC) framework. The core idea of MBC is to estimate the current global message using local historical information to reduce the requirement of receiving messages directly from other agents. This enables each agent to maintain an error-controlled overview of the global information under a partial communication setup, thus achieving high performance with low communication overhead. We first introduce the overview structure of the MBC framework, then present the detailed design of each module.

4.1 Overview of the MBC Framework

We start with an intuitive example about how an individual agent makes decisions with communication (as shown in Figure 1).

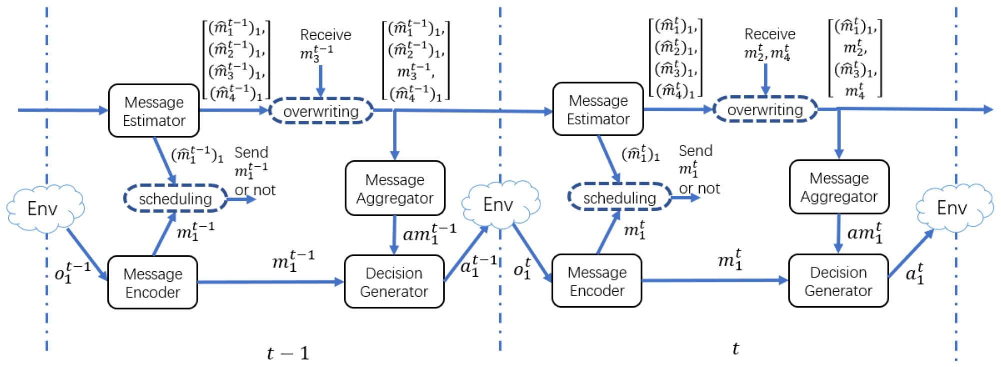


Figure 1: The demonstration of the proposed MBC framework, including the decision making process of one agent at time step t-1 and t in a four-agent MARL system. In addition to the notations mentioned in Section 3, $(\hat{m}_j^t)_i$ represents agent j's message at t estimated by agent i.

To simplify the description, we demonstrate the procedure in a four-agent setting. At time step $t-1$, agent 1 uses ***Decision Generator*** to take action a_1^{t-1} based on the local encoding m_1^{t-1} as well as the aggregated message am_1^{t-1}. The m_1^{t-1} comes from its ***Message Encoders*** by encoding the local observation o_1^{t-1}. The am_1^{t-1} is computed by the ***Message Aggregator*** using the overwritten message vector $[(\hat{m}_1^{t-1})_1, (\hat{m}_2^{t-1})_1, m_3^{t-1}, (\hat{m}_4^{t-1})_1]$. The components of this vector are either estimated messages computed by local agent 1 (i.e., $(\hat{m}_1^{t-1})_1, (\hat{m}_2^{t-1})_1, (\hat{m}_4^{t-1})_1$), or the real messages received from other agents (i.e., m_3^{t-1}).

To obtain this overwritten message vector at $t-1$, agent i first estimates the current global message vector $[(\hat{m}_1^{t-1})_1, (\hat{m}_2^{t-1})_1, (\hat{m}_3^{t-1})_1, (\hat{m}_4^{t-1})_1]$ via ***Message Estimator*** based on the overwritten message vector provided from the previous time. Then, the received real message will overwrite this estimated message vector to reduce the error of estimation. In this example, only $(\hat{m}_3^{t-1})_1$ is overwritten, because agent 1 only receives messages m_3^{t-1} from agent 3 at this moment.

In addition, once Message Estimator obtains the estimated global message vector, the self-estimation component $(\hat{m}_1^{t-1})_1$ will be taken out. The scheduling process then compares estimated $(\hat{m}_1^{t-1})_1$ and real local message m_1^{t-1} to decide whether to send m_1^{t-1} to other agents. This sending process, together with the overwriting, constitutes the communication before the decision.

At the next time step, agent 1 will repeat the above process. Notably, the demonstrated process in Figure 1 works on each individual agent. In other words, all agents contain the same framework, and each one holds its individual modules (i.e., Message Estimator, Message Encoder, Message Aggregator, and Decision Generator) and performs their individual scheduling and overwriting processes. The parameters of the four modules are shared by all agents in the system. Next, we present more details about how various modules and corresponding processes are modeled in the MBC framework.

4.2 Decision-making with Messages

This subsection describes how the Message Encoder encodes messages and how the Decision Generator generates actions. Agents in the system use a shared Message Encoder f_{enc} to encode the observations into messages. Specifically, the observation o_i^t of any agent i is initially encoded by a fully connected layer FC1, and then further encoded together with collected information from previous steps by an LSTM layer Hochreiter and Schmidhuber [1997], Hausknecht and Stone [2015] into m_i^t.

Thereafter, agents use a shared individual π to make decisions based on the encoded observations and aggregated messages. Specifically, the encoded observation m_i^t is concatenated with the aggregated message am_i^t. Then, they are input into another fully connected layer FC2 to generate individual decision a_i^t. In summary, agent i generates an action through the following equation.

$$a_i^t = f_{FC2}(m_i^t || am_i^t) \tag{2}$$

where f_{FC2} includes the fully connected layer $FC2$ and the action sampling process and $(\cdot||\cdot)$ denotes the concatenation operation.

4.3 Estimating Global Messages

This subsection describes how the Message Estimator is centrally trained and how the Message Estimator uses previous global message vector to estimate the current global messages. Message Estimator builds up a supervised learning model to estimate the up-to-date global message information based on messages from the previous step. Since the key to our idea is to reduce the communication channel overhead by using estimated messages as much as possible instead of received messages, training an accurate Message Estimator is essential.

To train an accurate Message Estimator, we seek inspiration from model-based reinforcement learning methods Janner et al. [2019], Nagabandi et al. [2018]. In the model-based MARL methods, agents model the dynamics of the observations. This dynamics, which determines the joint observation $\boldsymbol{o^t} = [o_1^t, o_2^t, ..., o_N^t]$ at time t, is modelled by the to be learned environment model f_o:

$$\boldsymbol{\hat{o}^t} = f_o(\boldsymbol{\hat{o}^{t-1}}, \boldsymbol{a^{t-1}}) \tag{3}$$

where $\boldsymbol{a^{t-1}} = [a_1^{t-1}, a_2^{t-1}, ..., a_N^{t-1}]$ is the action profile consisting of the actions of all agents at time step $t-1$. Since $\boldsymbol{m^t}$ is essentially the encoding of $\boldsymbol{o^t}$, based on Equation (3), we assume that the dynamics of the messages, which determines the joint message $\boldsymbol{m^t} = [m_1^t, m_2^t, ..., m_N^t]$ at time t, can also be estimated by the learned message model f_M.

$$\boldsymbol{\hat{m}^t} = f_M(\boldsymbol{m^{t-1}}, \boldsymbol{a^{t-1}}) \tag{4}$$

where $\boldsymbol{\hat{m}^t}$ is the estimated joint message. To obtain the joint action $\boldsymbol{a^{t-1}}$, we calculate its components separately with Equation (2), $a_i^{t-1} = f_{FC2}(m_i^{t-1}||am_i^{t-1})$, where $am_i^{t-1} = f_{agg}(\boldsymbol{m^{t-1}})$ and f_{agg} is the function of Message Aggregator. Therefore, since $\boldsymbol{a^{t-1}}$ depends on $\boldsymbol{m^{t-1}}$, we only require $\boldsymbol{m^{t-1}}$ to calculate $\boldsymbol{\hat{m}^t}$ in equation (4). In this way, we learn a model for the global dynamic in Equation (4) instead of on individual dynamic in Nagabandi et al. [2018], Kim et al. [2021]. Because in the MBC framework, an (overwritten) global message vector is always available for every agent, we can learn a global model to utilize more information as well as to train a stable message model.

During training, we collect real global messages and actions to train f_M in a supervised way. We use the Mean Square Error (MSE) between the true messages and the estimated messages as the loss function.

$$\mathcal{L}_M(\theta^{f_M}) = \mathbb{E}_{(\boldsymbol{m}^{t-1}, \boldsymbol{m}^t) \sim \mathcal{D}}[\boldsymbol{m^t} - f_M(\boldsymbol{m^{t-1}}, \boldsymbol{a^{t-1}})]^2 \tag{5}$$

where θ^{f_M} is the parameter of f_M and $\mathcal{D}$ is a buffer that stores tuples $(\boldsymbol{m}^{t-1}, \boldsymbol{m}^t)$ for $t = 1, 2, ..., T$. To better train f_M, we use $\mathcal{D}$ similar to Janner et al. [2019] to collect data as the training set during the interaction of agents with the environment. During training, $(\boldsymbol{m}^{t-1}, \boldsymbol{m}^t)$ will be sampled in batches from $\mathcal{D}$ to compute the gradient of Equation 5 and perform parameter updates on f_M.

In execution, we deploy the same parameters of the trained f_M to each agent locally. However, since the global information in Equations (4) and (2) is not always available for agent i, we use locally available information to estimate global message. Considering the general situation of Figure 1, agent i computes its own estimation on global message $\boldsymbol{(\hat{m}^t)_i} = [(\hat{m}_1^t)_i, (\hat{m}_2^t)_i, ..., (\hat{m}_N^t)_i]$ with the overwritten message vector $\boldsymbol{(\dot{m}^{t-1})_i} = [(\dot{m}_1^{t-1})_i, (\dot{m}_2^{t-1})_i, \ldots, (\dot{m}_N^{t-1})_i]$ from the last time step, where $(\dot{m}_j^{t-1})_i = (\hat{m}_j^{t-1})_i \; or \; m_j^{t-1}$ depending on whether agent i has received a real message m_j^{t-1} from j. If a message m_j^{t-1} was received, it will be used to overwrite the estimation $(\hat{m}_j^{t-1})_i$.

Therefore, in execution, agent i obtains its estimation on global message with the following equation.

$$\boldsymbol{(\hat{m}^t)_i} = f_M\left(\boldsymbol{(\dot{m}^{t-1})_i}, \boldsymbol{(\hat{a}^{t-1})_i}\right) \tag{6}$$

where $\boldsymbol{(\hat{a}^{t-1})_i} = [(\hat{a}_j^{t-1})_i]$ is the joint action estimated by agent i and $(\hat{a}_j^{t-1})_i = f_{FC2}((\dot{m}_j^{t-1})_i || f_{agg}(\boldsymbol{(\dot{m}^{t-1})_i}))$. Agent i needs to estimate the joint action because it cannot observe the global action at current time step in decentralized execution. Here, agent i is able to estimate the action of agent j individually because in our framework, different agents maintain similar overwritten global message, i.e., $\boldsymbol{(\dot{m}^{t-1})_j} \approx \boldsymbol{(\dot{m}^{t-1})_i}$, and all agents share the same parameters of f_{FC2} and f_{agg}. The more components of $\boldsymbol{(\dot{m}^{t-1})_i}$ are overwritten, the more accurate this approximation is. This also means a higher communication overhead as more messages are received.

4.4 Scheduling Messages

This subsection describes how to schedule messages. The scheduling process determines whether the current local message should be sent to all other agents. The local message scheduling aims to control the estimation error within a certain boundary. This can be done by checking whether other agents are able to properly estimate the local message of agent i at this moment. In other words, if other agents can estimate the message sent by agent i at time step t within the error bound, agent i does not need to send its local encoded observation m_i^t. (Message Estimator is shared and accessible to all agents in execution). Otherwise, m_i^t needs to be sent to other agents to help them recover from the incorrect estimation.

We propose to schedule the message in a decentralized way. This allows the message dispatching module to be better deployed in a distributed manner. To do this, it is necessary to use only the local information available during the execution. As mentioned in Subsection 4.3, each agent estimates others' current messages using Equation (6) during execution. Since each agent shares the same parameters of f_M, f_{FC2}, as well as f_{agg} and maintains similar global message vector inputs, we can use agent i's estimation on itself to approximate other agents' estimation on i, i.e., $(\hat{m}_i^t)_i \approx (\hat{m}_i^t)_j$, where $j \neq i$. In other words, if agent i cannot estimate its own current message m_i^t within an error range, agent i will assume that other agents cannot estimate m_i^t neither. Then, agent i should send its message to other agents to help other agents recover from a wrong estimation.

In summary, we design the message scheduling as the following equation.

$$I_i^t = \begin{cases} 1 & \text{if } ||(\hat{m}_i^t)_i - m_i^t||_2 > \delta, \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

where I_i^t is a binary value that indicates whether agent i sends its messages to other agents at time step t, $||\cdot||_2$ indicates $L2$ norm of a vector, δ is a hyperparameter representing the threshold value. According to Equation (7), A smaller δ leads to more frequent communication, which improves performance but increases communication overhead. Conversely, a larger δ reduces communication frequency, leading to lower performance due to message estimation errors, while also decreasing communication overhead.

In addition to the scheduling approach in Equation (7), to prevent the cumulative error in message estimation, we force agents to send messages if they have not sent a message after a time window w. Besides, when $t = 0$, we set all agents to send their messages to start the calculation of Equation (6).

4.5 Aggregating Messages

This subsection describes how to aggregate messages from the current overwritten global message vector. Since each agent needs to aggregate useful information from N messages, we solve this high input dimensionality by following the ideas of GATs Casanova et al. [2018], Niu et al. [2021]. In the calculation of individual agent i, the message in layer l is computed recursively by aggregating the messages from the last layer $l-1$:

$$(m_j^{t(l)})_i = \sigma(\sum_{k \in \mathcal{N}} (\alpha_{jk})_i W^{(l)} (m_k^{t(l-1)})_i) \tag{8}$$

where $(m_j^{t(l)})_i$ is the intermediate message component on layer l computed by agent i locally at time step t, $\sigma(\cdot)$ is a nonlinear activation function (i.e. LeakyReLU Nwankpa et al. [2018] in our experiments), $\mathcal{N}$ is the set of all agents in the system, $W^{(l)}$ is a learnable weighting matrix shared among all nodes at layer l, and $(\alpha_{jk})_i$ is the weight coefficient of agent k's message for agent j, as viewed by agent i. $(\alpha_{ij})_p$ is computed by the attention mechanism Casanova et al. [2018], Niu et al. [2021]. From Equation (8), it can be seen that the f_{agg} aggregates messages recursively and that the parameters of the f_{agg} consist of the learnable $W^{(l)}$ at each layer. In L-layer GATs, the input of GATs first layer is $(m_j^{t(0)})_i = (\dot{m}_j^t)_i$, where $(\dot{m}_j^t)_i$ comes from the component of the overwritten message vector $\boldsymbol{(\dot{m}^{t-1})_i}$. The output of the final layer is the aggregated message of agent i: $am_i = (m_i^{t(L)})_i$.

5 Experiments

We evaluate MBC on three mixed cooperative-competitive environments which are widely utilized tasks in previous the state-of-the-art work Zhang et al. [2020], Niu et al. [2021] to demonstrate the superiority of MBC. All of those baselines target on solving mix cooperative-competitive tasks like ours. These include CommNet Sukhbaatar et al. [2016] that requires all agents to communicate with each other, IC3Net Singh et al. [2019] for learning when to send messages, TarMAC+IC3Net Das et al. [2019] for learning aggregated messages using attention mechanism, GA-Comm Liu et al. [2020] for learning communication graphs, MAGIC Niu et al. [2021] for centralized message scheduling, and TMC Zhang et al. [2020] for decentralized message scheduling. Among these baselines, MAGIC is the state-of-the-art method in terms of agent performance and TMC is the state-of-the-art method in terms of efficient communication channel. Our code is open source[1].

5.1 Performance

We first compare the performance of MBC with the baselines to validate the superiority of MBC in solving MARL tasks. The performance of MBC and all baselines are presented in Figure 2. The error bounds (i.e., shadow shapes) indicate the upper and lower bounds of the performance with 10 runs. In PP-grid and CN-loc, the prey and landmarks are stationary. In these environments, our approach achieves similar performance to the previous baselines, while our approach shows better sample efficiency. When it comes to the more challenging PP-loc environment in which preys are moving, MBC significantly outperforms previous methods and exhibits a smaller performance variance.

5.2 Communication Efficiency

We evaluate the communication efficiency of MBC in this subsection. We define the communication efficiency using the ratio of performance to channel overhead. The larger this ratio is, the better ability to achieve higher performance using fewer channels for the algorithm.

Figure 3 compares the communication efficiency of the algorithms. To avoid calculating negative rewards, we take the algorithms' reward increment over random policy and normalize them over CommNet in the way similar to van Hasselt et al. [2016]: $r_{normalized}(alg) = \frac{r_{alg} - r_{random}}{r_{CommNet} - r_{random}}$, where $r_{normalized}(alg)$ is normalized reward for any alg method to be compared, r_{alg} is the average reward achieved by alg method, $r_{CommNet}$ and r_{random} are the average rewards achieved by CommNet method and by a random policy, respectively. In our experiments, we consider a message

[1]`https://github.com/shan0126/Model-Based-Communication`

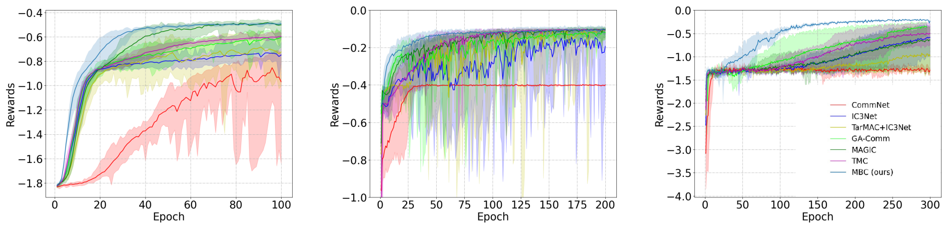

(a) PP-grid (b) CN-loc (c) PP-loc

Figure 2: Learning curves of various communication MARL algorithms in three mixed cooperative-competitive environments.

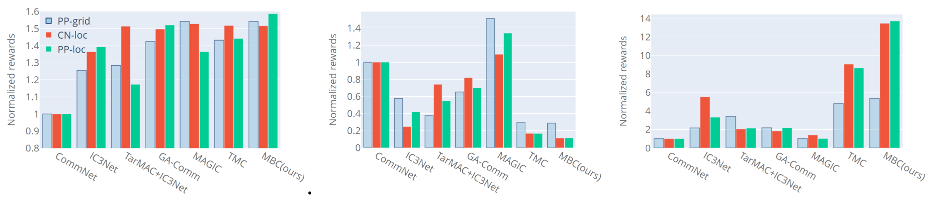

(a) Comparison on performance (b) Comparison on channels (c) Comparison on ratio of performance to channels

Figure 3: Communication efficiency comparison of various communication MARL algorithms in three mixed cooperative-competitive environments.

from one agent to another as one communication channel. The normalized channel $c_{normalized}(alg)$ for method alg is calculated by: $c_{normalized}(alg) = \frac{c_{alg}}{N(N-1)}$, where c_{alg} is the average channel consumed by alg method and N is the number of agent in the system. Figure 3(a) and 3(b) shows the performance and communication channel of each algorithm. Figure 3(c) presents the ratio of performance to channel overhead. The results show our MBC method can achieve higher performance with less channel overhead. Figure 3(c) demonstrates the significant improvement in communication efficiency of MBC compared to baselines.

As motivated in the introduction, achieving good performance with fewer channels is especially important in environments where communications are costly. Figure 4 shows the learning curves of MBC and MAGIC on CN-loc and PP-loc where the cost per communication channel is 0.001, 0.002 and 0.005 respectively. Both MBC and MAGIC use the environmental rewards to learn decisions.

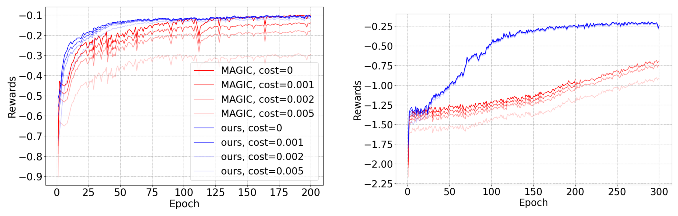
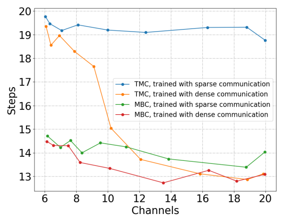

(a) CN-loc (b) PP-loc

Figure 4: The learning curves with communication overhead is counted in the reward.

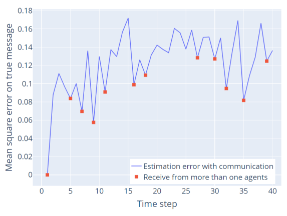
When the cost of the channel increases, the learning curve of MAGIC shifts significantly downward, whereas the learning curve of MBC only becomes slightly lower. This indicates that our approach is more stable and more efficient in environments where communication is costly. It further validate the feasibility of our method.

5.3 Channel Dependency

In real-world situations, agents may also need to accomplish tasks with restricted communication channels. Thus, we further validate the feasibility of our method by examining the influence of communication channel restriction on the performance of MBC. We compare our MBC with TMC that has the lowest communication overhead baseline in terms of channel dependency. The performance changes of two methods over different channels are shown in Figure 5(a). We limit the allowed communication channels in the training. Dense communication basically allows full communication, while sparse only allows up to 6 channels. We use these learned models to perform the task and record how many steps are needed to complete the task when we allow different numbers of communication channels. The horizontal axis of Figure 5(a) is average channel overhead per time step, and the vertical axis is the steps required for all predators to catch the prey in the PP-grid environment. The smaller the required steps, the more efficient the algorithm is in completing the task.



(a) The achieved performance changes over channels in PP-grid.



(b) Estimation error of agent with communication.

Figure 5: Performance changes over channels and estimation error changes over time steps.

As shown in Figure 5, with sparse communication training, TMC always requires more steps to complete the task than MBC in execution (compare blue and green lines). With dense communication training, TMC and MBC can achieve similar performance in the execution when there are enough communication channels. However, when there are fewer communication channels available, the efficiency of the TMC to complete the task becomes significantly lower, while the performance of MBC is only slightly reduced, as shown on the orange and red lines. In summary, MBC is more capable of communication with constraints on channels.

5.4 Interpretation on Message Scheduling

A remarkable feature of MBC is that communication is used to reduce estimation error on global message. Thus, we investigate the effects of communication (i.e. received messages) on the message estimation error. To do this, we track dynamic changes of the estimation error on global messages in one agent and plot them in Figure 5(b). As shown in the figure, the estimation error is decreased when more then one messages are received from other agents. This indicate the effectiveness of communication on correcting local estimation on global message.

6 Conclusion and future work

In this paper, we propose MBC which utilizes a message model to estimate up-to-date messages of other agents instead of always receiving messages from them. A decentralized message scheduling mechanism is also designed to correct the error of the agent's message estimation. The proposed method allows multiple agents to make collaborative decisions with sparse communication. In a variety of mixed cooperative-competitive environments, MBC shows better performance and lower communication overhead than the state-of-the-art method. The future work may include: 1) exploring alternative message prediction loss functions, e.g., predicting only messages influencing the agent's decisions, to make the message prediction more helpful for decision making; 2) developing a sparse communication learning that decides whether to send a message based on a small penalty for sending messages; 3) formulating a theoretical statement about the quality of joint decision-making based on the estimated messages never larger than δ.

References

Akshat Agarwal, Sumit Kumar, Katia P. Sycara, and Michael Lewis. Learning transferable cooperative behavior in multi-agent teams. In *Proceedings of the 19th International Conference on Autonomous Agents and Multiagent Systems*, pages 1741–1743. International Foundation for Autonomous Agents and Multiagent Systems, 2020.

Petar Veličković Guillem Cucurull Arantxa Casanova, Adriana Romero Pietro Lio, and Yoshua Bengio. Graph attention networks. *6th International Conference on Learning Representations*, 2018.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 1538–1546, 2019.

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

Yali Du, Bo Liu, Vincent Moens, Ziqi Liu, Zhicheng Ren, Jun Wang, Xu Chen, and Haifeng Zhang. Learning correlated communication topology in multi-agent reinforcement learning. In *20th International Conference on Autonomous Agents and Multiagent Systems*, pages 456–464, 2021.

Vladimir Egorov and Alexey Shpilman. Scalable multi-agent model-based reinforcement learning. In *21st International Conference on Autonomous Agents and Multiagent Systems*, pages 381–390, 2022.

Jakob Foerster, Ioannis Alexandros Assael, Nando De Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in neural information processing systems*, 29, 2016.

Jakob N. Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 2974–2982, 2018.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, 55(2):895–943, 2022.

Shixiang Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *2017 IEEE International Conference on Robotics and Automation*, pages 3389–3396, 2017.

Shushi Gu, Ye Wang, Niannian Wang, and Wen Wu. Intelligent optimization of availability and communication cost in satellite-uav mobile edge caching system with fault-tolerant codes. *IEEE Transactions on Cognitive Communications and Networking*, 6(4):1230–1241, 2020.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. In *2015 aaai fall symposium series*, 2015.

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

Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. When to trust your model: Model-based policy optimization. In *Advances in Neural Information Processing Systems 32*, pages 12498–12509, 2019.

Natasha Jaques, Angeliki Lazaridou, Edward Hughes, Çaglar Gülçehre, Pedro A. Ortega, DJ Strouse, Joel Z. Leibo, and Nando de Freitas. Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3040–3049, 2019.

Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Advances in Neural Information Processing Systems*, pages 7265–7275, 2018.

Daewoo Kim, Sangwoo Moon, David Hostallero, Wan Ju Kang, Taeyoung Lee, Kyunghwan Son, and Yung Yi. Learning to schedule communication in multi-agent reinforcement learning. In *7th International Conference on Learning Representations*, 2019.

Woojun Kim, Jongeui Park, and Youngchul Sung. Communication in multi-agent reinforcement learning: Intention sharing. In *9th International Conference on Learning Representations*, 2021.

Michael L Littman. Markov games as a framework for multi-agent reinforcement learning. In *Machine learning proceedings*, pages 157–163. 1994.

Yong Liu, Weixun Wang, Yujing Hu, Jianye Hao, Xingguo Chen, and Yang Gao. Multi-agent game abstraction via graph attention neural network. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 7211–7218, 2020.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, 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.

Hangyu Mao, Zhengchao Zhang, Zhen Xiao, Zhibo Gong, and Yan Ni. Learning agent communication under limited bandwidth by message pruning. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 5142–5149, 2020.

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *Proceedings of the 33nd International Conference on Machine Learning*, volume 48, pages 1928–1937, 2016.

Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation*, pages 7559–7566, 2018.

Yaru Niu, Rohan Paleja, and Matthew Gombolay. Multi-agent graph-attention communication and teaming. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, pages 964–973, 2021.

Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning. *CoRR*, abs/1811.03378, 2018.

Georgios Papoudakis, Filippos Christianos, and Stefano V. Albrecht. Agent modelling under partial observability for deep reinforcement learning. In *Advances in Neural Information Processing Systems 34*, pages 19210–19222, 2021.

Bei Peng, Tabish Rashid, Christian Schröder de Witt, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Boehmer, and Shimon Whiteson. FACMAC: factored multi-agent centralised policy gradients. In *Advances in Neural Information Processing Systems*, pages 12208–12221, 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. QMIX: monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 4292–4301, 2018.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

Esmaeil Seraj, Zheyuan Wang, Rohan R. Paleja, Daniel Martin, Matthew Sklar, Anirudh Patel, and Matthew C. Gombolay. Learning efficient diverse communication for cooperative heterogeneous teaming. In *21st International Conference on Autonomous Agents and Multiagent Systems*, pages 1173–1182, 2022.

Piyush K. Sharma, Rolando Fernandez, Erin G. Zaroukian, Michael R. Dorothy, Anjon Basak, and Derrik E. Asher. Survey of recent multi-agent reinforcement learning algorithms utilizing centralized training. *CoRR*, abs/2107.14316, 2021.

Amanpreet Singh, Tushar Jain, and Sainbayar Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *7th International Conference on Learning Representations*, 2019.

Sainbayar Sukhbaatar, Rob Fergus, et al. Learning multiagent communication with backpropagation. volume 29, pages 2244–2252, 2016.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinícius Flores Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 2085–2087, 2018.

Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 2094–2100, 2016.

Akifumi Wachi. Failure-scenario maker for rule-based agent using multi-agent adversarial reinforcement learning and its application to autonomous driving. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pages 6006–6012, 2019.

Zixin Wang, Hanyu Zhu, Mingcheng He, Yong Zhou, Xiliang Luo, and Ning Zhang. GAN and multi-agent DRL based decentralized traffic light signal control. *IEEE Transactions on Vehicular Technology*, 71(2):1333–1348, 2022.

Daniël Willemsen, Mario Coppola, and Guido C. H. E. de Croon. MAMBPO: sample-efficient multi-robot reinforcement learning using learned world models. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5635–5640, 2021.

Chongjie Zhang and Victor R. Lesser. Coordinating multi-agent reinforcement learning with limited communication. In *International conference on Autonomous Agents and Multi-Agent Systems*, pages 1101–1108, 2013.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. In *Advances in Neural Information Processing Systems 32*, pages 3230–3239, 2019a.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Efficient communication in multi-agent reinforcement learning via variance based control. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, pages 3230–3239, 2019b.

Sai Qian Zhang, Qi Zhang, and Jieyu Lin. Succinct and robust multi-agent communication with temporal message control. In *Advances in Neural Information Processing Systems*, volume 33, pages 17271–17282, 2020.

Weijia Zhang, Hao Liu, Jindong Han, Yong Ge, and Hui Xiong. Multi-agent graph convolutional reinforcement learning for dynamic electric vehicle charging pricing. In *The 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 2471–2481, 2022.

Changxi Zhu, Mehdi Dastani, and Shihan Wang. A survey of multi-agent reinforcement learning with communication. *CoRR*, abs/2203.08975, 2022.

EWRL Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our main claims in the abstract and introduction accurately reflect the paper's contributions and scope.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss it in 'conclusion and future work' section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: The paper does not include theoretical results.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include those information and provide the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While EWRL does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide open access to the environment and code.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the EWRL code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the EWRL code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: All those information is provided.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All the learning curves have their error bars.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).

- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The environments required to train the algorithm is provided in the code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the EWRL Code of Ethics `https://ewrl.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: research conducted in the paper conform with the EWRL Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the EWRL Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: There is no societal impact of the work performed.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [NA]

    Justification: The paper does not use existing assets.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer NA means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the EWRL Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the EWRL Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.