ACTIONS SPEAK LOUDER THAN WORDS: RATE-REWARD TRADE-OFF IN MARKOV DECISION PROCESSES

Anonymous authors
Paper under double-blind review

ABSTRACT

The impact of communication on decision-making systems has been extensively studied under the assumption of dedicated communication channels. We instead consider communicating through actions, where the message is embedded into the actions of an agent which interacts with the environment in a Markov decision process (MDP) framework. We conceptualize the MDP environment as a finite-state channel (FSC), where the actions of the agent serve as the channel input, while the states of the MDP observed by another agent (i.e., receiver) serve as the channel output. Here, we treat the environment as a communication channel over which the agent communicates through its actions, while at the same time, trying to maximize its reward. We first characterize the optimal information theoretic trade-off between the average reward and the rate of reliable communication in the infinite-horizon regime. Then, we propose a novel framework to design a joint control/coding policy, termed *Act2Comm*, which seamlessly embeds messages into actions. From a communication perspective, *Act2Comm* functions as a learning-based channel coding scheme for non-differentiable FSCs under input-output constraints. From a control standpoint, *Act2Comm* learns an MDP policy that incorporates communication capabilities, though at the cost of some control performance. Overall, *Act2Comm* effectively balances the dual objectives of control and communication in this environment. Experimental results validate *Act2Comm*'s capability to enable reliable communication while maintaining a certain level of control performance.

1 INTRODUCTION

The role of communication in multi-agent systems has received significant attention as it allows agents with a partial view of the system to better coordinate and cooperate by exchanging messages in parallel to actions taken in the environment (Foerster et al., 2016; Sukhbaatar et al., 2016). However, these systems rely on dedicated channels for communication. On the other hand, explicit communication channels may not always be available, or may be complemented with other *implicit* forms of communication. Such examples are abundant in nature. Bacteria communicate through chemical molecules, known as quorum sensing (Waters & Bassler, 2005), altering their environment and behavior to achieve population-wide coordination. Ants use pheromones to encode the path to food for other ants (von Thienen et al., 2014). Non-verbal communication through gestures, gaze, and even physical appearance, is also known to play an important role in human communication (Trenholm, 2020). In the artificial realm, autonomous robots may also need to rely on implicit communications when explicit communication channels are not available. In medical nano-robots, electromagnetic communication is not feasible due to size and energy limitations, but implicit communication can be achieved through molecular communications (Weiss & Knight, 2001; Wang et al., 2023). Even for more advanced robots, electromagnetic or other types of explicit communication channels may not be available in harsh or hostile environments; for example, for robots decommissioning nuclear storage facilities, or those operating in deep space, deep ocean, or subterranean environments, e.g., tunnels and caves (Ebadi et al., 2024). Moreover, wireless signals are prone to wiretapping due to their broadcast nature (Poor & Schaefer, 2017), and can be unreliable in adversarial scenarios due to jamming (Pirayesh & Zeng, 2022; Martz et al., 2020), which are other factors limiting explicit communications.

Motivated by these challenges, this paper explores implicit communication in a Markov Decision Process (MDP) environment—*communication through actions*—which facilitates information transmission from the MDP controller to other agents that can observe the MDP states. Effectively utilizing this internal channel has the potential to reduce the dependence on dedicated communication channels. However, these communication capabilities come with inherent trade-offs. As we will demonstrate, using this channel for communication often leads to a degradation in MDP control performance. This raises a fundamental challenge in balancing control and communication objectives, underscoring the need for a cohesive design that integrates both control and communication.

An MDP consists of a controller and an environment (Puterman, 2014). At each time step, the environment is in some state s, and the controller selects some action a. Upon executing action a, the environment stochastically transitions to a new state s' and generates a reward. The controller's objective is to find an optimal policy for selecting an action at each time to maximize its accumulated reward over a given time horizon. Consider that the controller now wants to communicate with another agent in the same environment that can also observe the environment state.

In communication theory, a finite-state channel (FSC) (Gallager, 1968) is an input-output system with states, where the output depends on both the input and the current state. Messages are encoded into the input sequence, resulting in a corresponding output sequence. The receiver decodes the message from this output sequence. From this perspective, the state transition of an MDP from s to s' upon taking action a can be viewed as an FSC from the controller to the agent. We refer to this internal channel within the MDP as an *action-state channel*. To communicate through the *action-state channel*, we need an encoder that maps the message to a sequence of actions, and a decoder that translates the resulting state sequence back into the original message. However, the objective of this encoder differs from that of the controller: the encoder aims to maximize the transmission rate and reliability of its message, while the controller seeks to maximize the accumulated reward. These two objectives are generally inconsistent, and a trade-off between the two must be sought.

In this paper, we first investigate the trade-off between the capacity (i.e., the maximum achievable transmission rate) of the *action-state channel* and the MDP reward in the infinite horizon regime. We demonstrate that the capacity of this channel can be expressed in a simple form—as the conditional mutual information between the input and output conditioned on the channel state. We also show that the capacity-reward trade-off can be characterized by a convex optimization, which can be solved numerically. While the capacity-reward trade-off provides an upper bound on the practically achievable rate under certain reward constraints, solving it does not yield a practical coding scheme. We then propose a practical framework for the integrated control and communication task in the finite block-length regime. The challenge of designing such a framework is twofold: (1) balancing the control and communication performance, and (2) dealing with the non-differentiability of the *action-state channel*. To tackle these issues, we propose *Act2Comm*, a transformer-based coding scheme in which encoder and decoder are trained iteratively.

Contributions. The main contributions of this paper are summarized as follows: (1) We introduce a novel paradigm of communicating through actions within an MDP environment, framing it as an integrated control and communication problem. (2) We derive the capacity of the action-state channel, and characterize the capacity-reward trade-off as a convex optimization. (3) We propose *Act2Comm*, a practical transformer-based coding scheme to learn a policy that optimizes communication performance while maintaining a specified level of MDP reward. *Act2Comm* can be of independent interest for designing practical channel coding schemes over other non-differentiable FSC scenarios.

2 RELATED WORK

Communication plays a significant role in MDPs, especially in multi-agent reinforcement learning (RL), where agents exchange messages over dedicated or noisy links to achieve a common goal (Wang et al., 2020; Chen et al., 2024; Tung et al., 2021). This is known as *emergent communications* (Boldt & Mortensen, 2024), but this framework relies on explicit communications over dedicated channels. Implicit communication through actions is considered by Knepper et al. (2017) and Tian et al. (2019). The latter also trains a policy, but it focuses on the multi-agent scenarios, and encour-

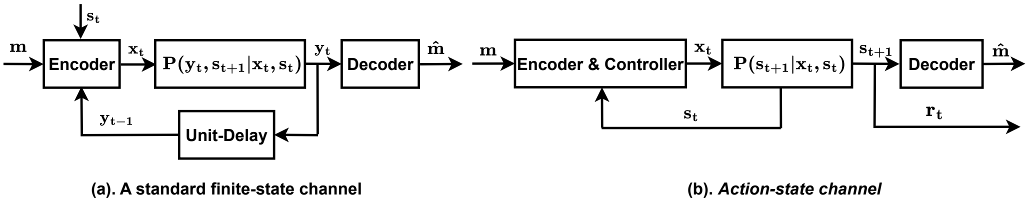


Figure 1: From a standard finite-state channel to an *action-state channel*.

ages communication by appropriately changing the reward function. We do not explicitly specify the communicated information, and instead, take a more fundamental approach by characterizing the information theoretic limits of communication and designing a practical coding policy.

Sokota et al. (2022) explored a similar concept of communication via MDPs. In their study, the receiver can observe the entire trajectory, including both the action and state sequences. This enables the controller to encode (compress) messages into the action sequence, and the receiver can subsequently decode the messages from the trajectory. Essentially, this is a randomized source coding problem. However, in most practical scenarios, while the MDP state is a physical signal observable by the receiver, the controller's actions are usually not observable to other agents. Therefore, in our work, we assume the receiver can only observe the state sequence. This shifts the problem from source coding to channel coding. Karabag et al. (2019) also examined a similar system, but their focus was on developing policies that restrict the observer's ability to infer transition probabilities.

FSC represents a general class of communication channels, and its study has been a long-standing problem in information and coding theory. Blackwell et al. (1958) studied the capacity of indecomposable FSCs without feedback. Subsequent studies in the non-feedback setting include Verdu & Han (1994) and Goldsmith & Varaiya (1996). The capacity of FSCs with feedback was examined by Massey (1990) and Permuter et al. (2009). More recently, Shemuel et al. (2022) explored the capacity of FSCs with feedback and state information at the encoder. However, these results express capacity in multi-letter forms, relying on the entire input and output sequences as their lengths approach infinity. Although Sabag et al. (2017) provided a single-letter upper bound for the feedback capacity of unifilar FSCs, exact single-letter expressions for FSC capacity are generally unknown. The action-state channel studied in this paper is a special FSC with state and feedback at the encoder. Utilizing the unique structure of this channel, we derive a single-letter expression for its capacity.

Machine learning has recently advanced traditional channel coding schemes by replacing linear operations with trainable non-linear neural networks, including Turbo autoencoder (Jiang et al., 2019), DeepPolar (Hebbar et al., 2024), KO codes (Makkuva et al., 2021), and other approaches (Jiang et al., 2020; Kim et al., 2018). However, these are designed for Gaussian channels, which are differentiable and allow joint training of the encoder and decoder. Our channel, in contrast, is non-differentiable, presenting new challenges for the design of the encoder and decoder. Channel coding for FSCs is a challenging task with limited results in the literature. Some existing work focuses only on the design of the decoder (Aharoni et al., 2023). However, the main challenge in our problem lies in designing the encoder to balance control and communication performance.

Notations: For any x_t with $t \geq 1$, $\boldsymbol{x}_{\boldsymbol{i}}^{\boldsymbol{k}}$ denotes the sequence $\{x_i, x_{i+1}, \ldots, x_k\}$, where $\boldsymbol{x}_{\boldsymbol{1}}^{\boldsymbol{k}}$ is written as $\boldsymbol{x}^{\boldsymbol{k}}$. $|\mathcal{X}|$ denotes the cardinality of the set $\mathcal{X}$. A detailed notation table is provided in Table. 1.

3 PRELIMINARIES AND SYSTEM MODEL

Markov Decision Process (MDP). An MDP can be characterized by a tuple $(\mathcal{S}, \mathcal{X}, \boldsymbol{T}, r, \alpha)$, where $\mathcal{S}$ is the state space, $\mathcal{X}$ is the action space, $\boldsymbol{T}$ is the transition kernel, $r : \mathcal{S} \times \mathcal{X} \to \mathbb{R}$ is the bounded reward function, and α is the initial state distribution. At each time step t, taking action x_t in state s_t results in a reward $r(x_t, s_t)$ and a state transition from s_t to s_{t+1}, where s_{t+1} is sampled from the distribution $\boldsymbol{T}(\cdot|s_t, x_t)$. We assume both $\mathcal{S}$ and $\mathcal{X}$ are finite sets. A stationary deterministic policy is a mapping $\pi : \mathcal{S} \to \mathcal{X}$ that selects action x_t based on state s_t at each time t.

The objective is to find an optimal policy that maximizes the long-term average reward, as follows:

$$\textbf{P1:} \quad \max_{\pi} \lim_{N\to\infty} \frac{1}{N}\mathbb{E}\left[\sum_{t=1}^{N} r(s_t, x_t)|s_1 \sim \alpha\right]. \tag{1}$$

In this paper, we assume that the MDP is unichain; that is, any deterministic policy induces a Markov chain consisting of a single recurrent class plus some transient states. As a result, the optimality of **P1** can be achieved through a stationary deterministic policy. The set of stationary deterministic policies is denoted by Π_{SD}. It is worth noting that the set of admissible policies for an MDP is not restricted to Π_{SD}. In general, a policy can be history-dependent, determining x_t using all historical states and actions up to time t. Let Π_S and Π_H denote the sets of stationary (possibly randomized) and history-dependent policies, respectively. It is easy to see that $\Pi_{SD} \subset \Pi_S \subset \Pi_H$.

Finite-State Channel (FSC). As illustrated in Fig. 1, an FSC can be characterized by a tuple $(\mathcal{X}\times\mathcal{S}, P_{Y,S^+|X,S}, \mathcal{Y}\times\mathcal{S})$, where $\mathcal{X}$ is the input alphabet, $\mathcal{Y}$ is the output alphabet, $\mathcal{S}$ is the channel state alphabet, and $P_{Y,S^+|X,S}$ is the channel law specifying the probability of the channel outputting Y and transitioning to the new state S^+, given that the channel input is X in state S. We consider a time-invariant channel that exhibits the Markov property, which can be formally expressed as:

$$P(y_t, s_{t+1}|\boldsymbol{x^t}, \boldsymbol{s^t}, \boldsymbol{y^{t-1}}) = P_{Y,S^+|X,S}(y_t, s_{t+1}|x_t, s_t), \ \forall t. \tag{2}$$

To transmit a message m, an encoder generates a sequence of channel inputs $\boldsymbol{x}^t$ as codewords. When each x_t is input to the channel, the channel transitions to a new state s_{t+1} and produces an output y_t. The decoder then collects the output sequence $\boldsymbol{y}^t$ to reconstruct m. We suppose that the channel state is available to the encoder but not to the decoder, and that the outputs are fed back to the encoder.

Let $\mathcal{M}$ denote the set of messages, with each message m uniformly sampled from $\mathcal{M}$. The encoder is defined as a sequence of mappings, $\mathcal{E} \triangleq \{\mathcal{E}_1, \ldots, \mathcal{E}_n\}$, where each mapping $\mathcal{E}_t : \mathcal{M} \times \mathcal{S}^t \times \mathcal{X}^{t-1} \times \mathcal{Y}^{t-1} \to \mathcal{X}$ generates the channel input at time t. In other words, the channel input at time t, $x_t = \mathcal{E}_t(m, \boldsymbol{s^t}, \boldsymbol{x^{t-1}}, \boldsymbol{y^{t-1}})$, is a function of m and all the historical information available at the transmitter up to time t. The decoder is defined as the mapping, $\mathcal{D} : \mathcal{Y}^n \to \mathcal{M}$, which reconstructs the message from all n channel outputs, $\hat{m} = \mathcal{D}(\boldsymbol{y}^n)$. The pair $(\mathcal{E}, \mathcal{D})$ constitutes a code, where n represents the code length. Suppose the message set is $\mathcal{M} = \{1, 2, \ldots, 2^k\}$, then each message can be represented with k bits. The rate of the code $(\mathcal{E}, \mathcal{D})$ is defined as $R_{\mathcal{E},\mathcal{D}} = k/n$.

The probability of error $P_e^{(n)}$ for $(\mathcal{E}, \mathcal{D})$ is defined as $P_e^{(n)} = \Pr(\mathcal{D}(\boldsymbol{y}^n) \neq m | m \text{ is sent})$. A rate R is deemed achievable if there exists a code $(\mathcal{E}, \mathcal{D})$ such that the error probability of the transmission approaches zero as $n \to \infty$. Consequently, the *capacity* of the FSC is defined as the supremum of all achievable rates. In other words, channel capacity reveals the maximum rate required for error-free transmission when the code length approaches infinity. In practice, however, constructing a code with infinite code length is unfeasible. Hence practical channel coding aims to balance the trade-off between R and $P_e^{(n)}$ with a finite code length n. For instance, we design codes to maximize the rate while ensuring that the probability of error remains below a certain threshold $\sigma > 0$:

$$\textbf{P2:} \quad \max_{\mathcal{E},\mathcal{D}} R_{\mathcal{E},\mathcal{D}}, \quad \text{subject to } P_e^{(n)} \leq \sigma. \tag{3}$$

Integrated Control and Communication. We investigate a scenario in which the controller of an MDP aims not only to optimize rewards but also to facilitate communication. Assuming that the receiver can observe the state of the MDP, then the environment can be modeled as a specialized FSC, enabling communication between the transmitter (i.e., the controller) and the receiver.

(a) Action-state channel model: This integrated FSC is referred to as the *action-state channel*, where the state of the MDP aligns with the state of the FSC. The action and the subsequent state are viewed as the channel input and output, respectively. Upon executing x_t in state s_t, the MDP environment returns a reward r_t and transitions to a new state s_{t+1}. Here s_{t+1} functions as both the channel output and the new channel state, and $\boldsymbol{s^{t-1}}$ represents not only the historical state sequence, but also the historical feedback signal. The channel law of the *action-state channel* is given by:

$$P(s_{t+1}|\boldsymbol{x^t}, \boldsymbol{s^t}) = P_{S^+|X,S}(s_{t+1}|x_t, s_t) = \boldsymbol{T}(s_{t+1}|x_t, s_t). \tag{4}$$

This type of channel is also referred to as a POST channel in the literature (Permuter et al., 2014).

(b) Controller & Encoder: Within this framework, the MDP controller and the FSC encoder represent the two aspects of the same entity, jointly responsible for selecting an action $x_t \in \mathcal{X}$ at each time step. However, their objectives differ: the controller aims to maximize the reward, while the FSC encoder seeks to maximize the message rate. Unlike the controller, which can focus on stationary deterministic policies, the encoder must account for more complex policy forms. For any message m, the encoder described in the previous part can be viewed as a history-dependent policy for the MDP. Therefore, we consider the joint control and coding policy in its most general form. The policy $\mathcal{E}$ is represented as a sequence of mappings $\{\mathcal{E}_i : 0 \leq i \leq n-1\}$, where $\mathcal{E}_i$ is defined as $\mathcal{E}_i : \mathcal{M} \times \mathcal{S}^i \times \mathcal{X}^{i-1} \to \mathcal{X}$. Each message is transmitted via a sequence of n actions. For example, if the controller begins to transmit a message m at time t, then $x_{t+i} = \mathcal{E}_i(m, \boldsymbol{s}_t^{t+i}, \boldsymbol{x}_t^{t+i-1})$ for $0 \leq i \leq n-1$. We assume the controller always has a new message ready for transmission immediately after completing the transmission of the previous message, and each message is uniformly sampled from $\mathcal{M}$. The long-term average reward of the MDP under policy $\mathcal{E}$ is denoted by $G_{\mathcal{E}}$.

(c) Decoder: The receiver observes the state sequence associated with a message and uses it to decode the message. For example, the state sequence associated with the i-th message is $\boldsymbol{s}_{in-n+1}^{in}$. The decoder, represented as a mapping : $\mathcal{D} : \mathcal{S}^n \to \mathcal{M}$, decodes the message as: $\hat{m} = \mathcal{D}(\boldsymbol{s}_{in-n+1}^{in})$.

In this paper, we consider non-terminating MDPs over an infinite time horizon and investigate the trade-off between control and communication performance. As discussed previously, if the code length $n \to \infty$, the communication performance can be characterized by the channel capacity (i.e., the maximum achievable rate). Here, we consider a practical setting with finite code length n, where the performance of a code $(\mathcal{E}, \mathcal{D})$ is characterized by its rate $R_{\mathcal{E},\mathcal{D}}$ and the error probability $P_e^{(n)}$. We study the trade-off through the following optimization problem with constants V and $\sigma > 0$,

$$\textbf{P3:} \quad \max_{\mathcal{E},\mathcal{D}} \; R_{\mathcal{E},\mathcal{D}} \tag{5}$$

$$s.t. \; G_{\mathcal{E}} \geq V \quad \text{and} \quad P_e^{(n)} \leq \sigma. \tag{6}$$

4 THE CAPACITY-REWARD TRADE-OFF

In this section, we analyze the trade-off between the capacity of the *action-state channel* and the MDP reward. While the results may not offer direct guidance for practical coding—since the capacity is typically achievable only in the infinite-horizon regime (i.e., when the code length $n \to \infty$)—they hold substantial theoretical importance. These findings provide a fundamental understanding of this integrated control and communication system. All proofs are detailed in Appendix B.

In information theory, the capacity of an FSC is usually expressed in terms of conditional mutual information (Shemuel et al., 2024). Let X, S^+ and S denote the random variable associated with the input, output (i.e., the next state), and the current state of the *action-state channel*, respectively. Then the conditional mutual information of X and S^+ given S is defined as (Cover, 1999):

$$I(X; S^+|S) = \mathbb{E}_{p(x,s^+,s)} \left[\log \frac{p(s, s^+|s)}{p(x|s)p(s^+|s)} \right]. \tag{7}$$

Let $\pi(\cdot|s)$ denote an input distribution of the channel given that the channel state is $s \in \mathcal{S}$. For a given channel, the joint distribution $p(x, s^+, s)$ is determined by the conditional input distribution π. From the MDP perspective, $\pi(x|s)$ represents the probability of selecting action x in state s; thus, π can be viewed as a stationary randomized policy for the MDP. Let ρ_π denote the equilibrium state distribution of the MDP under policy π. We have the following result:

Theorem 1 *The capacity of the action-state channel without reward constraint is given by*

$$C = \max_{\{\pi(x|s): x \in \mathcal{X}, s \in \mathcal{S}\}} I(X; S^+|S)$$

where X, S, and S^+ follow a joint distribution given by

$$p(x, s^+, s) = \rho_\pi(s)\pi(x|s)\boldsymbol{T}(s^+|s, x), \; x \in \mathcal{X}, s, s^+ \in \mathcal{S}.$$

As previously discussed, a general encoder for an FSC generates a channel input based on all the historical state and feedback information. However, Theorem 1 reveals a surprising fact: the capacity of the *action-state channel* can be achieved by encoding messages into a stationary randomized policy for the MDP, without relying on historical information.

Theorem 1 presents the capacity of the *action-state channel* without considering the MDP reward. If we want to maintain a certain level of long-term average reward for the MDP, the capacity may generally decrease. Next, we characterize the trade-off between channel capacity and MDP reward.

Given a stationary policy π for the MDP, define $w_\pi(s,x) = \rho_\pi(s)\pi(x|s)$. Here, $w_\pi(s,x)$ represents the long-term proportion of time that the MDP is in state s and takes action x. In the literature, w_π is referred to as the occupation measure of policy π (Altman, 2021). Let $\mathcal{W}$ denote the set of all occupation measures, then $\mathcal{W}$ is the set of $w \in \mathbb{R}^{|\mathcal{S}|\times|\mathcal{X}|}$ satisfying the following equations:

$$\sum_{x\in\mathcal{X}} w(s,x) - \sum_{s'\in\mathcal{S}}\sum_{x'\in\mathcal{X}} w(s',x')\boldsymbol{T}(s|s',x') = 0,\ \forall s \in \mathcal{S}, \tag{8}$$

$$\sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}} w(s,x) = 1,\ w(s,x) \geq 0,\ \forall s \in \mathcal{S}, x \in \mathcal{X}. \tag{9}$$

Clearly, $\mathcal{W}$ is a polytope. It is well-known that there is a one-to-one mapping between $\mathcal{W}$ and Π_S. In particular, $\pi(x|s) = w_\pi(s,x)/\sum_{x'} w_\pi(s,x')$ for any $s \in \mathcal{S}, x \in \mathcal{X}$. Using this relationship, the problem of computing the capacity with reward constraint V (i.e., ensuring the long-term average reward is not less than V) reduces to a convex optimization, as stated in the following theorem:

Theorem 2 *The capacity of the action-state channel with reward constraint V is the optimal value of the following convex optimization problem:*

$$\max_{w\in\mathcal{W}} I(w,\boldsymbol{T})$$

$$s.t. \sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}} w(s,x)r(s,x) \geq V$$

where $I(w,\boldsymbol{T})$ is a concave function of $w \in \mathcal{W}$ defined as

$$I(w,\boldsymbol{T}) \triangleq \sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}} w(s,x) \sum_{s'\in\mathcal{S}} \boldsymbol{T}(s'|s,x) \log \frac{\boldsymbol{T}(s'|s,x)\sum_{x''} w(s,x'')}{\sum_{x'} \boldsymbol{T}(s'|s,x')w(s,x')}.$$

Denote by $C(V)$ the capacity of the *action-state channel* with reward constraint V. We have:

Lemma 1 *$C(V)$ is a concave function.*

Since the capacity is an upper bound for the rate of any practical coding scheme, Lemma 1 implies that the achievable region of rate-reward pairs forms a convex set.

The convex optimization problem in Theorem 2 can be efficiently solved using the gradient ascent algorithm if the gradient of the objective function has a closed-form expression (Bertsekas, 2016). Next, we derive the gradient of $I(w,\boldsymbol{T})$ with respect to w. Define

$$l(w,w_n,\boldsymbol{T}) \triangleq \sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}} w(s,x) \sum_{s'\in\mathcal{S}} \boldsymbol{T}(s'|s,x) \log \frac{\boldsymbol{T}(s'|s,x)\sum_{x''} w_n(s,x'')}{\sum_{x'} \boldsymbol{T}(s'|s,x')w_n(s,x')},\ w, w_n \in \mathcal{W}.$$

Lemma 2 *For any $w_n \in \mathcal{W}$, $l(w,w_n,\boldsymbol{T})$ is a tangent line of $I(w,\boldsymbol{T})$ at point w_n. That is,*

(i) $l(w_n,w_n,\boldsymbol{T}) = I(w_n,\boldsymbol{T})$.

(ii) $l(w,w_n,\boldsymbol{T}) \geq I(w,\boldsymbol{T})$ *for all* w.

It follows immediately from Lemma 2 that

$$\nabla I_{w_n}(s,x) \triangleq \left.\frac{\partial I(w,\boldsymbol{T})}{\partial w(s,x)}\right|_{w=w_n} = \sum_{s'\in\mathcal{S}} \boldsymbol{T}(s'|s,x) \log \frac{\boldsymbol{T}(s'|s,x)\sum_{x''} w_n(s,x'')}{\sum_{x'} \boldsymbol{T}(s'|s,x')w_n(s,x')}, \tag{10}$$

for any $w_n \in \mathcal{W}$, $s \in \mathcal{S}$, and $x \in \mathcal{X}$. The gradient $\partial I/\partial w = [\nabla I_w(s,x)]_{s,x}$ then can be used in the gradient descent method to solve the optimization problem in Theorem 2.

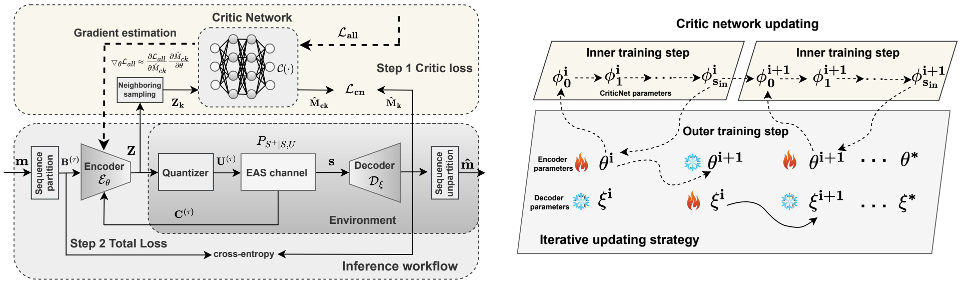


Figure 2: Left: Workflow diagram of the *Act2Comm* scheme, with the dashed line indicating the gradient flow. Right: Illustration of the iterative training strategy, incorporating a critic network.

5 ACT2COMM: A PRACTICAL CODING SCHEME

This section presents *Act2Comm*, a learning-based practical coding scheme that balances both control and communication objectives . This framework assumes a pre-determined control policy π that satisfies the reward constraint $G_\pi \geq V$, referred to as the target policy. Such a target policy can be easily derived using traditional MDP or RL algorithms. *Act2Comm* aims to learn a coding policy that: (1) closely mimics the stochastic behavior of the target policy; (2) minimizes the probability of decoding errors for a given coding rate. That is, *Act2Comm* takes a policy achieving the desired reward, and embeds messages into it with the desired reliability. For ease of reference, we denote the element of matrix $\boldsymbol{X}$ located at the i-th row and j-th column as $X[i, j]$. The sets of states and actions are indexed as $\mathcal{S} = \{0, 1, \ldots, |\mathcal{S}| - 1\}$ and $\mathcal{X} = \{0, 1, \ldots, |\mathcal{X}| - 1\}$.

Preliminary. Let $\mathcal{U} \triangleq \mathcal{X}^{|\mathcal{S}|}$, where each $u \in \mathcal{U}$ is referred to as a decision rule because the i-th element of u, can be viewed as an action for state i. Specifically, a decision rule specifies an action for each state; hence it can be used to determine an action for any given state. A control policy is thus a collection of decision rules spanning the entire time horizon. To align the coding policy with the target policy, we propose using a sequence of n decision rules as the codeword instead of n actions. To achieve this, the *action-state channel* is converted into an extended action-state (EAS) channel (Fig. 6) using Shannon's method (Shannon, 1958). In particular, we conceptually separate the encoder and controller, considering the controller as an integral component of the EAS channel. The channel state is assumed to be available at the controller but not the encoder. At each time t, the encoder selects a decision rule u_t from $\mathcal{U}$. Then the controller uses u_t and the state s_t to determine an action $x_t = u_t(s_t)$. Consequently, the EAS channel has an input alphabet $\mathcal{U}$, output alphabet $\mathcal{S}$, and channel law:

$$P_{S^+|S,U}(s_{t+1}|s_t, u_t) = P_{S^+|S,X}(s_{t+1}|s_t, u_t(s_t)) = \boldsymbol{T}(s_{t+1}|s_t, u_t(s_t)). \tag{11}$$

The EAS channel and the *action-state channel* are equivalent, and we will focus on the EAS channel to develop our coding scheme. In the rest of this section, we detail the *Act2Comm* framework in five main components: the overall workflow, block-attention feedback coding, transceiver design, joint optimization of control and communication, and the iterative training strategy.

Overall workflow. As depicted in Fig. 2, given a k-bit message $\boldsymbol{m} \in \{0, 1\}^k$, the *Act2Comm* first encodes $\boldsymbol{m}$ into a belief map $\boldsymbol{Z} \in \mathbb{R}^{|\mathcal{S}| \times \frac{k}{R}}$. This $\boldsymbol{Z}$ is subsequently mapped into a codeword $\boldsymbol{U} \in \mathcal{X}^{|\mathcal{S}| \times \frac{k}{R}}$ by a quantizer. At each time step t, the controller selects an action $x_t = \boldsymbol{U}[s_t, t]$ for state s_t, and transitions into a new state according to the channel law $P(s_{t+1}|\boldsymbol{x^t}, \boldsymbol{s^t})$. After $\frac{k}{R}$ time steps, the receiver decodes the message based on the accumulated observations $\boldsymbol{s} \in \mathcal{S}^{\frac{k}{R}}$.

Block-attention feedback coding. One of the principal innovations of *Act2Comm* is the *block-attention* coding mechanism, which reduces the coding complexity and enhances the performance.

(a) Message block: Formally, we partition message $\boldsymbol{m}$ into l blocks as $\boldsymbol{m} = [\boldsymbol{b_1}; \boldsymbol{b_2}; \ldots; \boldsymbol{b_l}]$, where each block $\boldsymbol{b_i} \in \{0, 1\}^\mu$ contains $\mu = k/l$ bits. This allows us to encode $\boldsymbol{m}$ with l coding rounds, with each round consisting of μ/R time steps. For each coding round τ $(1 \leq \tau \leq l)$, the input message block is defined as $\boldsymbol{B}^{(\tau)} \triangleq [2\boldsymbol{b_1} - 1; \ldots; 2\boldsymbol{b_\tau} - 1] \in \mathbb{R}^{\tau \times \mu}$.

(b) Feedback block: Although Theorem 1 shows that the capacity-achieving code with infinite blocklength can be history-independent, feedback has demonstrated benefits in simplifying the coding process for better performance in the practical finite blocklength regime (Kostina et al., 2017; Kim et al., 2020). Hence, for each time step t within the τ-th coding round, we introduce the feedback vector as $\boldsymbol{c}_{\boldsymbol{t}}^{(\boldsymbol{\tau})} \triangleq [s_t^{(\tau)}, x_t^{(\tau)}, s_{t+1}^{(\tau)}] \in \mathbb{R}^{1\times 3}$, which encapsulates the current state, the selected action, and the subsequent state. The feedback matrix for the τ-th round can then be given by $\boldsymbol{C}_{\boldsymbol{\tau}} = [\boldsymbol{c}_{\boldsymbol{1}}^{(\boldsymbol{\tau})}; \dots; \boldsymbol{c}_{\frac{\boldsymbol{\mu}}{\boldsymbol{R}}}^{(\boldsymbol{\tau})}] \in \mathbb{R}^{\frac{\mu}{R}\times 3}$. Consequently, for each coding round τ, we concatenate prior feedback matrices to construct a feedback block: $\boldsymbol{C}^{(\boldsymbol{\tau})} \triangleq [\boldsymbol{C}_{\boldsymbol{1}}; \dots; \boldsymbol{C}_{\boldsymbol{\tau}}] \in \mathbb{R}^{\tau\times\frac{\mu}{R}\times 3}$.

Transceiver design.

(a) Encoder: At each coding round τ, a transformer-based encoder is utilized to generate a belief matrix $\boldsymbol{Z}^{(\tau)} \in \mathbb{R}^{\tau\times\frac{\mu|\mathcal{S}|}{R}}$ using $\boldsymbol{B}^{(\boldsymbol{\tau})}$ and $\boldsymbol{C}^{(\boldsymbol{\tau})}$. The detailed architecture is provided in Fig. 9, with each component illustrated in Appendix C. The τ-th row vector of $\boldsymbol{Z}^{(\tau)}$, denoted as $\boldsymbol{z}^{(\boldsymbol{\tau})} \triangleq \boldsymbol{Z}^{(\tau)}[\tau,:] \in \mathbb{R}^{\frac{\mu|\mathcal{S}|}{R}}$, represents the belief vector derived from the τ-th coding round. After completing all l coding rounds, the selected belief vectors are combined to form the final belief map $\boldsymbol{Z} = [\boldsymbol{z}^{(\boldsymbol{1})}, \dots, \boldsymbol{z}^{(\boldsymbol{l})}] \in \mathbb{R}^{\frac{k}{\mu}\times\frac{\mu|\mathcal{S}|}{R}}$, which is subsequently reshaped into $\boldsymbol{Z} \in \mathbb{R}^{|\mathcal{S}|\times\frac{k}{R}}$. Each element of this reshaped belief map $Z[s_t, t]$ indicates the action belief at time step t given state s_t.

(b) Quantizer: *Act2Comm* employs a quantizer to generate the codeword $\boldsymbol{U} \in \mathcal{X}^{|\mathcal{S}|\times\frac{k}{R}}$ as:

$$\boldsymbol{U} = \mathcal{Q}(|\mathcal{X}| * \text{Sigmoid}(\boldsymbol{Z})), \tag{12}$$

where $\mathcal{Q} : \mathbb{R}^{|\mathcal{S}|\times\frac{k}{R}} \to \mathcal{X}^{|\mathcal{S}|\times\frac{k}{R}}$ is the quantization operation that maps the coding result to the nearest action index in the action space $\mathcal{X}$, and each element of resultant codeword $x_t = U[s_t, t]$ represents the selected action for state s_t at time step t.

(c) Decoder: Given the state observations $\boldsymbol{s} \triangleq [s_1 \dots, s_{\frac{k}{R}}] \in \mathcal{S}^{\frac{k}{R}}$, a transformer-based decoder is utilized to output logits $\hat{\boldsymbol{M}} \in \mathbb{R}^{\frac{k}{\mu}\times 2^{\mu}}$ for all blocks. After applying the softmax function, each block is predicted and subsequently transformed into the reconstructed bitstream $\hat{\boldsymbol{m}}$.

Joint optimization of control and communication. To model the trade-off between control and communication, we utilize a weighted loss function: $\mathcal{L}_{all} = \mathcal{L}_{com} + \lambda \cdot \mathcal{L}_{cont}$. The communication loss $\mathcal{L}_{com}$ is defined as the cross-entropy between the predictions from a critic network and their corresponding ground-truth, which quantifies the message decoding accuracy. To ensure control performance, we aim to make the coding policy behave closely to the target policy. Therefore, $\mathcal{L}_{cont}$ measures the "distance" between the coding policy and the target policy.

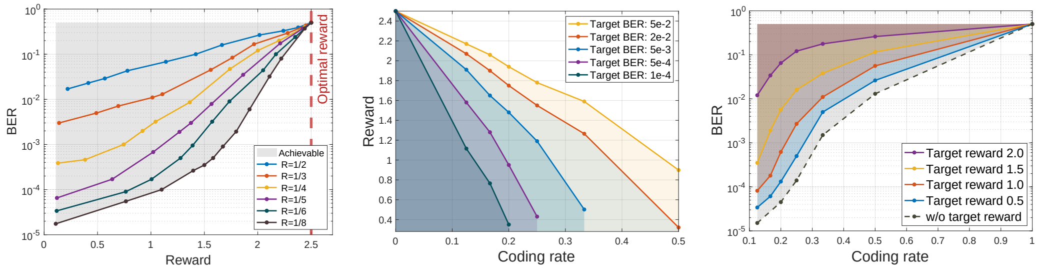
Let π denote the target policy, with $\pi(x|s)$ representing the probability of taking action x in state s. Let $f_U(x|s)$ denote the frequency of selecting x in state s across all decision rules in $\boldsymbol{U}$. We then use the mean square error (MSE) between π and f_U to measure the control loss for its stability in experiments. However, f_U is non-differentiable during the backpropagation as it is discrete. To address this issue, we estimate $f_U(x|s)$ using $\boldsymbol{Z}$ in equation 12. Let $\boldsymbol{e}$ denote the all-one row vector, and define $\Gamma_Z(\boldsymbol{T}, s, x) \triangleq \text{Sigmoid}(\gamma \cdot (|\mathcal{X}|\text{Sigmoid}(\boldsymbol{Z}[s,:]) - x\boldsymbol{e}))$. When $\gamma > 0$ is sufficiently large, $\Gamma_Z(\boldsymbol{T}, s, x)$ is a (kR)-dim vector with elements close to either 0 or 1. Additionally, $\Gamma_Z(\boldsymbol{T}, s, x)\boldsymbol{e}^\top$ approximates the number of elements in $\boldsymbol{U}[s,:]$ that are not less than x. We refer to γ as the temperature parameter and estimate $f_U(x|s)$ for $x > 0$ as follows:

$$f_U(x|s) \approx \hat{f}_U(x|s) \triangleq \frac{1}{kR}\left[\Gamma_Z(\boldsymbol{T}, s, x-1)\boldsymbol{e}^\top - \Gamma_Z(\boldsymbol{T}, s, x)\boldsymbol{e}^\top\right]. \tag{13}$$

For $x = 0$, we have $f_U(0|s) \approx \hat{f}_U(0|s) \triangleq 1 - \Gamma_Z(\boldsymbol{T}, s, 0)\boldsymbol{e}^\top/kR$. As a result, we define the control loss as $\mathcal{L}_{cont} = \mathit{MSE}(\pi, \hat{f})$.

Iterative training strategy. Given the non-differentiable nature of the EAS channel and quantizer, jointly updating the encoder and decoder is infeasible. To address this, we introduce a critic network and employ an iterative updating strategy to train *Act2Comm* effectively, with the corresponding algorithm and architectures detailed in the Appendix. C.4.

(a) Critic network As shown in Fig. 2, a critic network is introduced to estimate the gradient during gradient backpropagation for the encoder optimization, which views the EAS channel and decoder



(a) BER v.s. Reward (b) Reward v.s. Rate (c) BER v.s. Rate

Figure 3: Control-communication trade-off of *Act2Comm* in “Lucky Wheel”.

as an unknown environment. Before each update of the encoder, a critic network is trained over s_{in} inner steps to predict the logits for the neighbor belief maps of a given $\boldsymbol{Z}$. For each inner step k, the network is trained to predict the corresponding logits $\hat{\boldsymbol{M}}_{\boldsymbol{k}}$ as $\hat{\boldsymbol{M}}_{\boldsymbol{ck}}$ based on the neighbor belief maps sampled from $\boldsymbol{Z}_{\boldsymbol{k}} = \boldsymbol{Z} + \boldsymbol{W}_{\boldsymbol{k}}$, where $\boldsymbol{W}_{\boldsymbol{k}} \in \mathbb{R}^{|\mathcal{S}| \times \frac{k}{R}} \sim \mathcal{N}(0, \sigma_w^2)$ is the Gaussian noise term for neighboring sampling during the k-th inner step. The MSE loss, denoted as $\mathcal{L}_{cn}$, is utilized between $\hat{\boldsymbol{M}}_{\boldsymbol{ck}}$ and $\hat{\boldsymbol{M}}_{\boldsymbol{k}}$. With this design, we aim to obtain a precise critic network to estimate gradients from neighboring samples of $\boldsymbol{Z}$ in a given environment, thereby helping to update the encoder. Note that this extra training costs is incurred only during the offline training process, this critic network will be removed during the inference phase, as detailed in Appendix C.5-C.6..

(b) Iterative updating strategy As outlined in Fig. 2, at each update step i, we first train a critic network $\boldsymbol{\phi}^{\boldsymbol{i}}_{\boldsymbol{s_{in}}}$ using s_{in} inner steps to learn to estimate the gradient around the samples. Next, the encoder is updated to $\boldsymbol{\theta}^{\boldsymbol{i+1}}$ using the frozen decoder parameters $\boldsymbol{\xi}^{\boldsymbol{i}}$ and the learned gradient estimation. Subsequently, the decoder is directly optimized with the loss function to obtain new parameters $\boldsymbol{\xi}^{\boldsymbol{i+1}}$, while keeping the encoder frozen at $\boldsymbol{\theta}^{\boldsymbol{i+1}}$. This process iteratively alternates between encoder and decoder updates, freezing one while optimizing the other at each step.

6 EXPERIMENTAL RESULTS

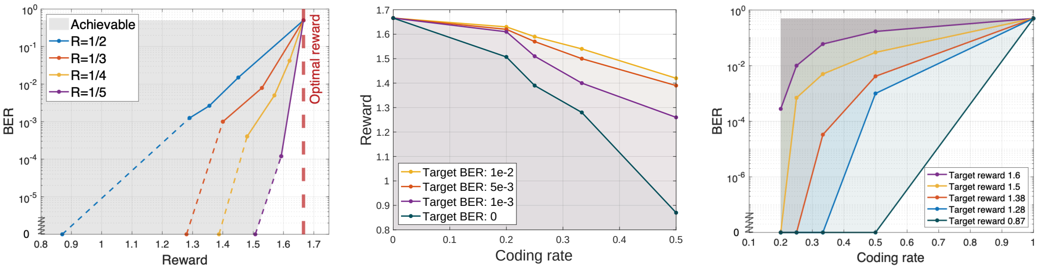
We evaluate *Act2Comm* across three distinct MDP environments, as detailed in Appendix D, with communication performance measured by the bit error rate (BER). Due to page limitations, the results of Environment 3, “Erratic robot”, are provided in the Appendix D.4.

Experiment 1: Lucky Wheel. In this game, agent keeps spinning a wheel to accumulate rewards by choosing either clockwise or counterclockwise. It is modeled as an MDP with 3 states and 2 actions, the details of the environment and experimental setting is provided in Fig. 10a in Appendix D.

We examine the trade-off among the three performance metrics in Fig. 3. We set the optimal reward-maximizing policy as the target policy, and consider different code rates. By adjusting λ, we can control how closely the coding policy approximates the target policy. The shaded regions in the figures (a)-(c) represent achievable regions of *Act2Comm* with various λ. When λ is large, regardless of the coding rate, *Act2Comm* learns a policy that mirrors the target policy for the optimal reward. In this case, all messages are mapped to the same sequence of decision rules since the target policy is stationary and deterministic. As a result, the BER is 0.5, indicating no communication capability.

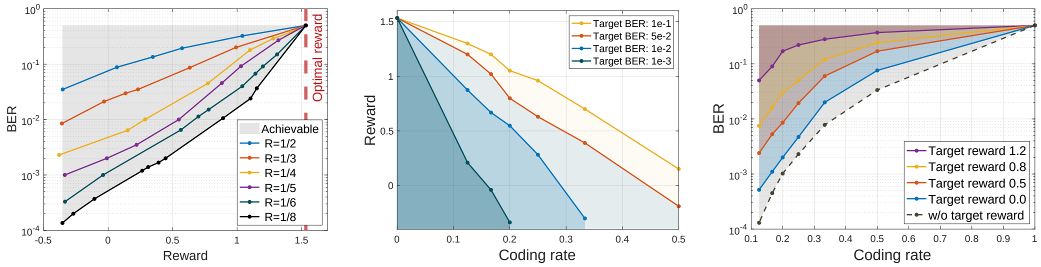
Next, we consider different target BERs, resulting in a trade-off between the code rate and reward. As shown in Fig. 3b, achieving a pre-determined BER with a higher coding rate results in a reduced reward. When targeting a lower BER, the reward decreases rapidly with the coding rate. Fig. 3c illustrates *Act2Comm*’s ability to balance BER and coding rate when ensuring a specified reward. Our results reveal that reducing the reward constraint leads to more reliable communication at the same rate. In summary, these findings demonstrate that *Act2Comm* can communicate messages through its actions at acceptable reliability while satisfying specific reward criteria.

Experiment 2: Catch the Ball. Next we consider “Catch the Ball”, which is an MDP with 27 states and 3 actions. This MDP includes a parameter p that influences its transition matrix (see the



(a) BER v.s. Reward (b) Reward v.s. Rate (c) BER v.s. Rate

Figure 4: Control-communication trade-off of *Act2Comm* in "Catch the Ball" with $p = 0$.



(a) BER v.s. Reward (b) Reward v.s. Rate (c) BER v.s. Rate

Figure 5: Control-communication trade-off of *Act2Comm* in "Catch the Ball" with $p = 0.2$.

Appendix D for details). When $p = 0$, the action-state channel is perfect as each action can be reliably inferred from the resulting state transition, but it becomes noisy for $p > 0$.

We first consider $p = 0$, where no coding is needed since there is no noise. The challenge is to maintain a certain reward while communicating. *Act2Comm* performs excellently in this environment. As shown in Fig. 4a, with a minor reduction in reward from 1.66 to 1.5, *Act2Comm* communicates at a rate of 0.2 with no error. If we relax BER to 10^{-4}, same rate can be achieved with a reward of 1.6. The trade-off between reward and rate is shown in Fig. 4b. This experiment highlights our model's capacity to enable efficient communication capabilities with minimal impact on performance. We also applied *Act2Comm* to this game with $p = 0.2$, in which the action-state channel is noisy and the coding process becomes more complex. As detailed in Fig. 5, we can observe a reduction in the coding rates for the same level of reliability due to the stochasticity in the environment.

7 CONCLUSION

We introduced a novel framework of *communication through actions*, a form of implicit communication from the controller of an MDP to a receiver that can observe the states. By treating the MDP environment as a communication channel, messages can be encoded into the action sequence and decoded from the state sequence. Aiming to optimize communication performance while ensuring a certain MDP reward, we formulated an integrated control and communication problem. We derived the capacity of the action-state channel and demonstrated that the trade-off between channel capacity and reward can be characterized as a convex optimization problem. We then proposed *Act2Comm*, a transformer-based framework for designing joint control and communication policies. Through experiments, we demonstrated *Act2Comm*'s capability to communicate reliably through actions while maintaining a certain level of MDP reward.

The proposed *Act2Comm* framework can be used as a plug-in component in various MDP and RL applications, enabling information transmission by learning a joint control and coding policy that closely mimics the target policy. More importantly, our study demonstrates the potential of communication through actions in multi-agent systems. While this form of implicit communication leads to some loss in control performance, it may potentially improve the overall control performance by enhancing coordination when applied to multi-agent systems where explicit communication channels are not available. This presents an interesting and challenging direction for future research.

REFERENCES

Ziv Aharoni, Bashar Huleihel, Henry D Pfister, and Haim H Permuter. Data-driven neural polar codes for unknown channels with and without memory. *arXiv preprint arXiv:2309.03148*, 2023.

Eitan Altman. *Constrained Markov decision processes*. Routledge, 2021.

Dimitri Bertsekas. *Nonlinear programming*. Athena Scientific, 2016.

David Blackwell, Leo Breiman, and Aram J Thomasian. Proof of shannon's transmission theorem for finite-state indecomposable channels. *The Annals of Mathematical Statistics*, pp. 1209–1220, 1958.

Brendon Boldt and David R Mortensen. A review of the applications of deep learning-based emergent communication. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=jesKcQxQ7j.

Jingdi Chen, Tian Lan, and Carlee Joe-Wong. Rgmcomm: Return gap minimization via discrete communications in multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 17327–17336, 2024.

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Kamak Ebadi, Lukas Bernreiter, Harel Biggie, Gavin Catt, Yun Chang, Arghya Chatterjee, Christopher E. Denniston, Simon-Pierre Deschênes, Kyle Harlow, Shehryar Khattak, Lucas Nogueira, Matteo Palieri, Pavel Petráček, Matěj Petrlík, Andrzej Reinke, Vít Krátký, Shibo Zhao, Aliakbar Agha-mohammadi, Kostas Alexis, Christoffer Heckman, Kasra Khosoussi, Navinda Kottege, Benjamin Morrell, Marco Hutter, Fred Pauling, François Pomerleau, Martin Saska, Sebastian Scherer, Roland Siegwart, Jason L. Williams, and Luca Carlone. Present and future of slam in extreme environments: The darpa subt challenge. *IEEE Transactions on Robotics*, 40:936–959, 2024. doi: 10.1109/TRO.2023.3323938.

EA Fainberg. On controlled finite state markov processes with compact control sets. *Theory of Probability & Its Applications*, 20(4):856–862, 1976.

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*, volume 29. Curran Associates, Inc., 2016. URL https://papers.nips.cc/paper_files/paper/2016/hash/c7635bfd99248a2cdef8249ef7bfbef4-Abstract.html.

Robert G Gallager. *Information theory and reliable communication*, volume 588. Springer, 1968.

A.J. Goldsmith and P.P. Varaiya. Capacity, mutual information, and coding for finite-state markov channels. *IEEE Transactions on Information Theory*, 42(3):868–886, 1996. doi: 10.1109/18.490551.

S Ashwin Hebbar, Sravan Kumar Ankireddy, Hyeji Kim, Sewoong Oh, and Pramod Viswanath. Deeppolar: Inventing nonlinear large-kernel polar codes via deep learning. *arXiv preprint arXiv:2402.08864*, 2024.

Onésimo Hernández-Lerma and Jean B Lasserre. *Discrete-time Markov control processes: basic optimality criteria*, volume 30. Springer Science & Business Media, 2012.

Yihan Jiang, Hyeji Kim, Himanshu Asnani, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Turbo autoencoder: Deep learning based channel codes for point-to-point communication channels. *Advances in neural information processing systems*, 32, 2019.

Yihan Jiang, Hyeji Kim, Himanshu Asnani, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Learn codes: Inventing low-latency codes via recurrent neural networks. *IEEE Journal on Selected Areas in Information Theory*, 1(1):207–216, 2020.

Mustafa O. Karabag, Melkior Ornik, and Ufuk Topcu. Least inferable policies for markov decision processes. In *2019 American Control Conference (ACC)*, pp. 1224–1231, 2019. doi: 10.23919/ACC.2019.8815129.

Hyeji Kim, Yihan Jiang, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Deepcode: Feedback codes via deep learning. *Advances in neural information processing systems*, 31, 2018.

Hyeji Kim, Yihan Jiang, Sreeram Kannan, Sewoong Oh, and Pramod Viswanath. Deepcode: Feedback codes via deep learning. *IEEE Journal on Selected Areas in Information Theory*, 1(1): 194–206, 2020.

Ross A. Knepper, Christoforos I. Mavrogiannis, Julia Proft, and Claire Liang. Implicit communication in a joint action. In *2017 12th ACM/IEEE International Conference on Human-Robot Interaction (HRI*, pp. 283–292, 2017.

Victoria Kostina, Yury Polyanskiy, and Sergio Verd. Joint source-channel coding with feedback. *IEEE Transactions on Information Theory*, 63(6):3502–3515, 2017.

Ashok V Makkuva, Xiyang Liu, Mohammad Vahid Jamali, Hessam Mahdavifar, Sewoong Oh, and Pramod Viswanath. Ko codes: inventing nonlinear encoding and decoding for reliable wireless communication via deep-learning. In *International Conference on Machine Learning*, pp. 7368–7378. PMLR, 2021.

Jeffrey Martz, Wesam Al-Sabban, and Ryan N Smith. Survey of unmanned subterranean exploration, navigation, and localisation. *IET Cyber-Systems and Robotics*, 2(1):1–13, 2020.

James Massey. Causality, feedback and directed information. In *Proc. Int. Symp. Inf. Theory Applic.(ISITA-90)*, pp. 303–305, 1990.

Haim Henri Permuter, Himanshu Asnani, and Tsachy Weissman. Capacity of a post channel with and without feedback. *IEEE Transactions on Information Theory*, 60(10):6041–6057, 2014. doi: 10.1109/TIT.2014.2343232.

Haim Henry Permuter, Tsachy Weissman, and Andrea J. Goldsmith. Finite state channels with time-invariant deterministic feedback. *IEEE Transactions on Information Theory*, 55(2):644–662, 2009. doi: 10.1109/TIT.2008.2009849.

Hossein Pirayesh and Huacheng Zeng. Jamming attacks and anti-jamming strategies in wireless networks: A comprehensive survey. *IEEE Communications Surveys & Tutorials*, 24(2):767–809, 2022. doi: 10.1109/COMST.2022.3159185.

H. Vincent Poor and Rafael F. Schaefer. Wireless physical layer security. *Proceedings of the National Academy of Sciences*, 114(1):19–26, 2017. doi: 10.1073/pnas.1618130114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1618130114.

Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Oron Sabag, Haim H. Permuter, and Henry D. Pfister. A single-letter upper bound on the feedback capacity of unifilar finite-state channels. *IEEE Transactions on Information Theory*, 63(3):1392–1409, 2017. doi: 10.1109/TIT.2016.2636851.

Claude E Shannon. Channels with side information at the transmitter. *IBM journal of Research and Development*, 2(4):289–293, 1958.

Eli Shemuel, Oron Sabag, and Haim H Permuter. Finite-state channels with feedback and state known at the encoder. *arXiv preprint arXiv:2212.12886*, 2022.

Eli Shemuel, Oron Sabag, and Haim H. Permuter. Finite-state channels with feedback and state known at the encoder. *IEEE Transactions on Information Theory*, 70(3):1610–1628, 2024. doi: 10.1109/TIT.2023.3336939.

Samuel Sokota, Christian A Schroeder De Witt, Maximilian Igl, Luisa M Zintgraf, Philip Torr, Martin Strohmeier, Zico Kolter, Shimon Whiteson, and Jakob Foerster. Communicating via markov decision processes. In *International Conference on Machine Learning*, pp. 20314–20328. PMLR, 2022.

Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. Learning multiagent communication with backpropagation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/55b1927fdafef39c48e5b73b5d61ea60-Paper.pdf.

Zheng Tian, Shihao Zou, Ian Davies, Tim Warr, Lisheng Wu, Haitham Bou Ammar, and Jun Wang. Learning to communicate implicitly by actions. *arXiv:cs.AI:1810.04444*, 2019. URL https://arxiv.org/abs/1810.04444.

S. Trenholm. *Thinking Through Communication: An Introduction to the Study of Human Communication (9th ed.)*. Routledge, 2020. doi: 10.4324/9781003016366.

Tze-Yang Tung, Szymon Kobus, Joan Pujol Roig, and Deniz Gündüz. Effective Communications: A Joint Learning and Communication Framework for Multi-Agent Reinforcement Learning Over Noisy Channels. *IEEE Journal on Selected Areas in Communications*, 39(8):2590–2603, August 2021. ISSN 1558-0008. doi: 10.1109/JSAC.2021.3087248. URL https://ieeexplore.ieee.org/document/9466501.

S. Verdu and Te Sun Han. A general formula for channel capacity. *IEEE Transactions on Information Theory*, 40(4):1147–1157, 1994. doi: 10.1109/18.335960.

Wolfhard von Thienen, Dirk Metzler, Dong-Hwan Choe, and Volker Witte. Pheromone communication in ants: a detailed analysis of concentration-dependent decisions in three species. *Behavioral ecology and sociobiology*, 68:1611–1627, 2014.

Jiaming Wang, Sevda Öğüt, Haitham Al Hassanieh, and Bhuvana Krishnaswamy. Towards practical and scalable molecular networks. In *Proceedings of the ACM SIGCOMM 2023 Conference*, ACM SIGCOMM '23, pp. 62–76. Association for Computing Machinery, 2023. doi: 10.1145/3603269.3604881.

Rundong Wang, Xu He, Runsheng Yu, Wei Qiu, Bo An, and Zinovi Rabinovich. Learning Efficient Multi-agent Communication: An Information Bottleneck Approach. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9908–9918. PMLR, November 2020. URL https://proceedings.mlr.press/v119/wang20i.html. ISSN: 2640-3498.

Christopher M Waters and Bonnie L Bassler. Quorum sensing: cell-to-cell communication in bacteria. *Annual Review of Cell and Developmental Biology*, 21(1):319–346, 2005.

Ron Weiss and Thomas F. Knight. Engineered communications for microbial robotics. In Anne Condon and Grzegorz Rozenberg (eds.), *DNA Computing*, pp. 1–16, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. ISBN 978-3-540-44992-8.

Michael Zhang, James Lucas, Jimmy Ba, and Geoffrey E Hinton. Lookahead optimizer: k steps forward, 1 step back. *Advances in neural information processing systems*, 32, 2019.

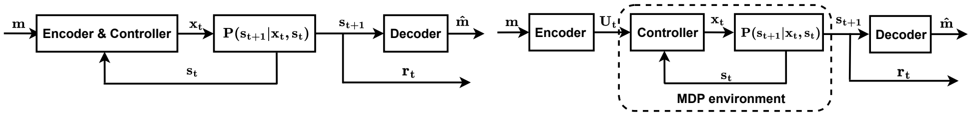
APPENDIX

A NOTATION AND DEFINITIONS

To bridge the RL and FSC areas, we provide a notation table for this paper in this section. Note that lowercase and uppercase bold letters represent vectors and matrices, respectively.

Table 1: Notation table

In general	
$\boldsymbol{x_i^k}$, $\boldsymbol{x^k}$	Sequence $\{x_i, x_{i+1}, \dots, x_k\}$, sequence $\{x_1, \dots, x_k\}$
$\lvert\mathcal{X}\rvert$	Cardinality of set $\mathcal{X}$
MDP: $(\mathcal{S}, \mathcal{X}, \boldsymbol{T}, r, \alpha)$	
s_t, x_t	MDP state and action at t
r, α	Reward function and initial state distribution
$\mathcal{S}$, $\mathcal{X}$	State and action space for MDP
$\boldsymbol{T}$, π	Transition kernel and control policy
FSC: $(\mathcal{X} \times \mathcal{S}, P_{Y,S^+\lvert X,S}, \mathcal{Y} \times \mathcal{S})$	
s_t, x_t, y_t	Channel state, input, and output at t
$\mathcal{X}$, $\mathcal{Y}$, $\mathcal{S}$	Channel input, output alphabet, and state alphabet
S^+, S	Next and current state (random variables)
$\mathcal{E}$, $\mathcal{D}$	Encoder and decoder
k, n	Message bit length and the code length
$R_{\mathcal{E},\mathcal{D}} = k/n$	Rate of code $(\mathcal{E}, \mathcal{D})$
Action-state channel: $P(s_{t+1}\lvert\boldsymbol{x^t}, \boldsymbol{s^t})$	
s_t, x_t	State and action at t
$\mathcal{S}$, $\mathcal{X}$	State alphabet, action alphabet
S^+, S	Next and current state (random variables)
$\mathcal{E}$, $\mathcal{D}$, $\boldsymbol{T}$, $G_{\mathcal{E}}$	Encoder, decoder, transition kernel, average reward
k, n, R	Message bit length, code length, and rate
$\boldsymbol{s}_{in-n+1}^{in}$	The state sequence associated with the i-th message
EAS channel for *Act2Comm*: $P(s_{t+1}\lvert\boldsymbol{x^t}, \boldsymbol{s^t})$	
s_t, x_t, u_t; π	State, action and decision rule at t; Target policy
S^+, S	Next and current state (random variables)
$\mathcal{U}$, $\mathcal{X}$	Alphabets of decision rule and actions
$\boldsymbol{T}$, $G_{\mathcal{E}}$	transition kernel and average reward
$\mathcal{E}(\cdot)$, $\mathcal{D}(\cdot)$, $\mathcal{C}(\cdot)$	Encoder, decoder, and critic network.
$\boldsymbol{\theta}$, $\boldsymbol{\xi}$, $\boldsymbol{\phi}$	Parameters of encoder, decoder, critic network
$\boldsymbol{Z}$, $\boldsymbol{U}$	Encoded belief map and codeword
$\boldsymbol{B}^{(\tau)}$, $\boldsymbol{C}^{(\tau)}$	Message and feedback block at coding round τ
$\boldsymbol{X}^{(\tau)}$	Control policy at coding round τ
$\hat{\boldsymbol{M}}$	Decoded logits
$\mathcal{Q}(\cdot)$	Quantizer
$\mathcal{L}_{all}$	Weighted loss function to update the encoder
$\mathcal{L}_{com}$, $\mathcal{L}_{cont}$	Communication loss term and Control loss term
$\mathcal{L}_{cn}$	MSE loss for critic network
$f_U(x\lvert s)$	Frequency of selecting action x in state s across $\boldsymbol{U}$
γ	Estimation temperature for control policy
$\hat{f}_U(x\lvert s)$	Estimation of $f_U(x\lvert s)$
$\boldsymbol{Z_k}$, $\boldsymbol{W_k}$	Sampled neighbors of $\boldsymbol{Z}$ and Gaussian term
$\hat{\boldsymbol{M}}_{\boldsymbol{k}}$	The corresponding logits from the frozen decoder
$\hat{\boldsymbol{M}}_{\boldsymbol{ck}}$	Predicted logits from the critic network (the k-th step)



(a). *Action-state channel* (b). *Extended action-state channel*

Figure 6: The *action-state channel* and the equivalent *extended action-state channel*.

B TECHNICAL PROOFS

This section presents the proofs of Section 4.

B.1 PROOF OF THEOREM 1

The proof of Theorem 1 relies on converting the action-state channel to an equivalent channel. This equivalence is also stated in Section 5, as it is crucial for the design of Act2Comm. To enhance readability, we present the equivalence here as well.

Let $\mathcal{U} \triangleq \mathcal{X}^L$, where each $u \in \mathcal{U}$ is referred to as a decision rule because the i-th element of u, denoted by $u(i)$, can be viewed as an action for state i. A control policy is thus a collection of decision rules spanning the entire time horizon. To measure the similarity between the coding policy and the target policy, we propose that the codeword for a message should be a sequence of n decision rules, rather than a sequence of n actions. To this end, we convert the *action-state channel* into an extended action-state (EAS) channel, as depicted in Fig. 6, using Shannon's method (Shannon, 1958). In particular, we conceptually separate the encoder and controller, considering the controller as an integral component of the EAS channel. We then assume that the channel state is available at the controller but not the encoder. At each time t, the encoder selects a decision rule u_t from $\mathcal{U}$. Then the controller uses u_t and the state s_t to determine an action $x_t = u_t(s_t)$. Consequently, the EAS channel has an input alphabet $\mathcal{U}$, output alphabet $\mathcal{S}$, and channel law:

$$P_{S^+|S,U}(s_{t+1}|s_t, u_t) = P_{S^+|S,X}(s_{t+1}|s_t, u_t(s_t)) = \boldsymbol{T}(s_{t+1}|s_t, u_t(s_t)).$$

The EAS channel and the *action-state channel* are equivalent, and we will examine the EAS channel instead of the *action-state channel* to derive the capacity.

Assume that the initial state of the *action-state channel* is fixed to be s_1. Then it is well-known that the capacity of the EAS channel is (Gallager, 1968)

$$C(s_1) = \max_{\{p(u_i|u^{i-1})\}_{i\geq 1}} \lim_{N\to\infty} \frac{1}{N} I(U^N; S_2^{N+1}|s_1), \tag{14}$$

where $I(U^N; S_2^{N+1}|s_1)$ is the mutual information given by

$$I(U^N; S_2^{N+1}|s_1) = \sum_{u^N \in \mathcal{U}^N} \sum_{s_2^{N+1} \in \mathcal{S}^N} p(u^N) P(s_2^{N+1}|u^N, s_1) \log \frac{P(s_2^{N+1}|u^N, s_1)}{\sum_{z^N \in \mathcal{U}^N} p(z^N) P(s_2^{N+1}|z^N, s_1)}.$$

For two random variables S and X, let $H(X|S)$ denote the conditional entropy (Cover, 1999) of X given S. Then we have

$$\begin{aligned}
\lim_{N\to\infty} \frac{1}{N} I(U^N; S_2^{N+1}|s_1) &= \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \left[H(S_{i+1}|S_2^i, s_1) - H(S_{i+1}|S_2^i, U^N, s_1)\right] \\
&\overset{(a)}{=} \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \left[H(S_{i+1}|S_2^i, s_1) - H(S_{i+1}|S_2^i, U^N, X_i, s_1)\right] \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \left[H(S_{i+1}|S_i, s_1) - H(S_{i+1}|S_i, X_i, s_1)\right] \\
&= \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} I(X_i; S_{i+1}|S_i, s_1)
\end{aligned}$$

where (a) holds because X_i is determined by S^i and U_i. It follows that

$$\begin{aligned}
C &= \max_{\{p(u_i|u^{i-1})\}_{i\geq 1}} \lim_{N\to\infty} \frac{1}{N} I(U^N; S_2^{N+1}|s_1) \\
&\overset{(a)}{=} \max_{\{p(x_i|s^i,x^{i-1})\}_{i\geq 1}} \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} I(X_i; S_{i+1}|S_i, s_1) \\
&\overset{(b)}{=} \max_{\{p(x_i|s_i)\}_{i\geq 1}} \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} I(X_i; S_{i+1}|S_i, s_1) \qquad (15)
\end{aligned}$$

In equation 15, (a) follows from the equivalence between the *action-state channel* and the EAS channel. We next prove (b). Let η denote a history-dependent encoder and $\{\eta_i(x_i|s^i, x^{i-1})\}_{i\geq 1}$ denote the associated input distribution. Furthermore, let $P_i^{\eta}(s,x)$ denote the probability that $S_i = s$ and $X_i = s$ conditioned on the encoder η. Note that η can be viewed as a history-dependent policy for the MDP. Then according to MDP theory, for any history-dependent policy η, there exists a Markov policy η' such that η and η' share the same joint probability distribution of states and actions. In particular, denote by $\eta_i'(x_i|s_i)$ the probability of selecting action x_i given that the state is s_i at time i. Then η' can be seen as a Markov encoder with input distribution $\{\eta_i'(x_i|s_i)\}_{i\geq 1}$. By letting

$$\eta_i'(x_i|s_i) = \eta_i(x_i|s_i) \triangleq \sum_{s^{i-1},x^{i-1}} \eta_i(x_i|s^i, x^{i-1}) P(s^{i-1}, x^{i-1}|s_1),$$

we have

$$P_i^{\eta}(s,x|s_1) = P_i^{\eta'}(s,x|s_1),\ \forall i \geq 1, s \in \mathcal{S}, x \in \mathcal{X}$$

We omit the proof here. The interested readers are referred to Theorem 5.5.1 of (Puterman, 2014) for the formal statement and proof. Consequently, we can verify that for any history-dependent encoder η, there exists a Markov encoder η' such that they result in the same $I(X_i; S_{i+1}|S_i, s_1)$ for all i:

$$\begin{aligned}
I_\eta(X_i; S_{i+1}|S_i, s_1) &= \sum_{s_i\in\mathcal{S}} \sum_{x_i\in\mathcal{X}} P_i^{\eta}(s_i, x_i|s_1) \sum_{s_{i+1}\in\mathcal{S}} \boldsymbol{T}(s_{i+1}|s_i, x_i) \log \frac{\boldsymbol{T}(s_{i+1}|s_i, x_i)}{\sum_{x_i'} \boldsymbol{T}(s_{i+1}|s_i, x_i')\eta_i(x_i'|s_i)} \\
&= \sum_{s_i\in\mathcal{S}} \sum_{x_i\in\mathcal{X}} P_i^{\eta'}(s_i, x_i|s_1) \sum_{s_{i+1}\in\mathcal{S}} \boldsymbol{T}(s_{i+1}|s_i, x_i) \log \frac{\boldsymbol{T}(s_{i+1}|s_i, x_i)}{\sum_{x_i'} \boldsymbol{T}(s_{i+1}|s_i, x_i')\eta_i'(x_i'|s_i)}
\end{aligned}$$

We thus conclude that restricting on Markov encoders does not result in any loss of capacity.

Next, we show that, as far as finding a capacity-achieving encoder is concerned, it is enough to consider stationary encoders. To see this, we reformulate equation 15 as a dynamic programming (DP) defined as follows:

- state at time i: $s_i \in \mathcal{S}$
- action at time i given state s_i: $q_i(s_i) \in \Delta(\mathcal{X})$ with $q_i(x_i|s_i) = P(x_i|s_i)$ being the i-th element
- transition law: $P(s_{i+1}|s_i, x_i) = \sum_{x_i\in\mathcal{X}} q_i(x_i|s_i)\boldsymbol{T}(s_{i+1}|s_i, x_i)$
- reward function:
$$r(s_i, q_i(s_i)) = \sum_{x_i\in\mathcal{X}} q_i(x_i|s_i) \sum_{s_{i+1}\in\mathcal{S}} \boldsymbol{T}(s_{i+1}|s_i, x_i) \log \frac{\boldsymbol{T}(s_{i+1}|s_i, x_i)}{\sum_{x_i'} \boldsymbol{T}(s_{i+1}|s_i, x_i')q_i(x_i'|s_i)}$$

Then the capacity expression given in equation 15 can be written as

$$C = \max_{\{q_i\}_{i\geq 1}} \lim_{N\to\infty} \frac{1}{N} \sum_{i=1}^{N} \sum_{s_i\in\mathcal{S}} P(s_i) r(s_i, q_i(s_i)) \qquad (16)$$

We can think of equation 16 as a problem of maximizing the long-term average reward of the above DP over the set of Markov deterministic policies. Essentially, the DP is an MDP with finite state space, compact action space, and bounded reward function. We can easily verify the following:

1. The reward function is a continuous function of action d_i.

2. The transition law depends continuously on the action d_i.
3. Any stationary policy yields a Markov chain with one ergodic class and a possibly empty set of transient states (Assumption 1).

Then, according to MDP theory (see, e.g., (Fainberg, 1976) and (Hernández-Lerma & Lasserre, 2012)), the maximum of equation 16 can be attained by a stationary deterministic policy. Therefore, problem equation 16 is equivalent to

$$C = \max_{\{\pi(\cdot|s)\in\Delta(\mathcal{X}):s\in\mathcal{S}\}} \lim_{N\to\infty} \frac{1}{N}\sum_{i=1}^{N} I(X_i;S_{i+1}|S_i,s_1) \tag{17}$$

Note that each $\{\pi(\cdot|s)\in\Delta(\mathcal{X}) : s\in\mathcal{S}\}$ corresponds to a stationary policy for the original MDP. Under Assumption 1, any stationary policy yields a Markov chain with equilibrium state distribution $\rho_\pi(s)$. Therefore, the following probability converges to a probability that is independent of s_1:

$$\lim_{i\to\infty} P(X_i = x, S_i = s, S_{i+1} = s'|s_1) = \rho_\pi(s)\pi(x|s)\boldsymbol{T}(s'|s,x)$$

As a result, $I(X_i;S_{i+1}|S_i,s_1)$ also converges to a value that is independent of s_1. The desired result follows immediately.

B.2 PROOF OF THEOREM 2

We first show that the capacity of the *action-state channel* without reward constraint can be written as:

$$C = \max_{w\in\mathcal{W}} I(w,\boldsymbol{T}). \tag{18}$$

To see this, using Theorem 1 and the formula $\pi(x|s) = w_\pi(s,x)/\sum_{x'} w_\pi(s,x')$ yields

$$\begin{aligned} I(X;S'|S) &= \sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}} \rho_\pi(s)\pi(x|s)\sum_{s'\in\mathcal{S}} \boldsymbol{T}(s'|s,x)\log\frac{\boldsymbol{T}(s'|s,x)}{\sum_{x'}\boldsymbol{T}(s'|s,x')q(x'|s)} \\ &= \sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}} w_\pi(s,x)\sum_{s'\in\mathcal{S}} \boldsymbol{T}(s'|s,x)\log\frac{\boldsymbol{T}(s'|s,x)\sum_{x''} w_q(s,x'')}{\sum_{x'}\boldsymbol{T}(s'|s,x')w_q(s,x')}. \end{aligned}$$

Then the equivalence between equation 18 and the capacity expression given in Theorem 1 follows immediately from the one-to-one mapping between $\mathcal{W}$ and Π_S.

It is well-known that the long-term average reward of a policy can be expressed as a linear function of its occupation measure:

$$G_\pi = \lim_{N\to\infty}\frac{1}{N}\mathbb{E}_\pi\left[\sum_{t=1}^{N} r(s_t,x_t)|s_1\sim\alpha\right] = \sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}} w_\pi(s,x)r(s,x). \tag{19}$$

Since the reward constraint is linear, to show that the optimization problem is a convex optimization, it suffices to prove that $I(w,\boldsymbol{T})$ is a concave function. For any $\lambda\in[0,1]$ and $\bar{\lambda} = 1-\lambda$, let $w = \lambda w_1 + \bar{\lambda} w_2$, where $w_1, w_2 \in \mathcal{W}$. Then we have

$$\begin{aligned} &\sum_{x\in\mathcal{X}} w(s,x)\boldsymbol{T}(s'|s,x)\log\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w(s,x')}{\boldsymbol{T}(s'|s,x)\sum_{x''}w(s,x'')} \\ &= \sum_{x\in\mathcal{X}} w(s,x)\boldsymbol{T}(s'|s,x)\log\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w(s,x')}{\sum_{x''}w(s,x'')} - \sum_{x\in\mathcal{X}} w(s,x)\boldsymbol{T}(s'|s,x)\log\boldsymbol{T}(s'|s,x). \end{aligned} \tag{20}$$

The second term of equation 20 is clearly linear w.r.t. w. For the first term, an application of log-sum inequality gives

$$\begin{aligned}&\left(\sum_{x\in\mathcal{X}} w(s,x)\boldsymbol{T}(s'|s,x)\right)\log\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w(s,x')}{\sum_{x''}w(s,x'')}\\ \le&\lambda\left(\sum_{x\in\mathcal{X}} w_1(s,x)\boldsymbol{T}(s'|s,x)\right)\log\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w_1(s,x')}{\sum_{x''}w_1(s,x'')}\\ &+\bar{\lambda}\left(\sum_{x\in\mathcal{X}} w_2(s,x)\boldsymbol{T}(s'|s,x)\right)\log\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w_2(s,x')}{\sum_{x''}w_2(s,x'')}.\end{aligned}\tag{21}$$

Combining equation 20 and equation 21 yields

$$\begin{aligned}&\sum_{x\in\mathcal{X}} w(s,x)\boldsymbol{T}(s'|s,x)\log\frac{\boldsymbol{T}(s'|s,x)\sum_{x''}w(s,x'')}{\sum_{x'}\boldsymbol{T}(s'|s,x')w(s,x')}\\ \ge&\lambda\sum_{x\in\mathcal{X}} w_1(s,x)\boldsymbol{T}(s'|s,x)\log\frac{\boldsymbol{T}(s'|s,x)\sum_{x''}w_1(s,x'')}{\sum_{x'}\boldsymbol{T}(s'|s,x')w_1(s,x')}\\ &+\bar{\lambda}\sum_{x\in\mathcal{X}} w_2(s,x)\boldsymbol{T}(s'|s,x)\log\frac{\boldsymbol{T}(s'|s,x)\sum_{x''}w_2(s,x'')}{\sum_{x'}\boldsymbol{T}(s'|s,x')w_2(s,x')}.\end{aligned}\tag{22}$$

Summing both sides of the above inequality over s and s' yields

$$I(w,\boldsymbol{T})=I(\lambda w_1+\bar{\lambda}w_2,\boldsymbol{T})\ge\lambda I(w_1,\boldsymbol{T})+\bar{\lambda}I(w_2,\boldsymbol{T}).$$

We thus conclude that $I(w,\boldsymbol{T})$ is a concave function of w. This completes the proof.

B.3 PROOF OF LEMMA 1

Define

$$\mathcal{W}_V=\left\{w\in\mathcal{W}:\sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}}w(s,x)r(s,x)\ge V\right\}.$$

For any achievable V_1 and V_2, let

$$w_i=\arg\max_{w\in\mathcal{W}_{V_i}}I(w,\boldsymbol{T}),\ i=1,2.$$

Then $C(V_i)=I(w_i,\boldsymbol{T})$, $i=1,2$. For any $\theta\in[0,1]$, let $\bar{\theta}=1-\theta$. Let $V=\theta V_1+\bar{\theta}V_2$ and $w_3=\theta w_1+\bar{\theta}w_2$. Then clearly $w_3\in\mathcal{W}_V$. We thus have

$$C(V)=\max_{w\in\mathcal{W}_V}I(w,\boldsymbol{T})\ge I(w_3,\boldsymbol{T})\ge\theta I(w_1,\boldsymbol{T})+\bar{\theta}I(w_2,\boldsymbol{T})=\theta C(V_1)+\bar{\theta}C(V_2).$$

We thus conclude that $C(V)$ is concave.

B.4 PROOF OF LEMMA 2

Since $I(w,\boldsymbol{T})$ is concave w.r.t. w and $l(w,w_n,\boldsymbol{T})$ is linear w.r.t. w, to show that $l(w,w_n,\boldsymbol{T})$ is a tangent line of $I(w,\boldsymbol{T})$ at point w_n, it is enough to prove that statements (i) and (ii) hold. Statement (i) holds trivially by the definitions of the two functions.

For statement (ii), consider

$$\begin{aligned}&l(w,w_n,\boldsymbol{T})-I(w,\boldsymbol{T})\\ =&\sum_{s\in\mathcal{S}}\sum_{x\in\mathcal{X}}w(s,x)\sum_{s'\in\mathcal{S}}\boldsymbol{T}(s'|s,x)\log\frac{\sum_{x''}w_n(s,x'')}{\sum_{x'}\boldsymbol{T}(s'|s,x')w_n(s,x')}\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w(s,x')}{\sum_{x''}w(s,x'')}.\end{aligned}$$

Define

$$P_w(s'|s)=\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w(s,x')}{\sum_{x''}w(s,x'')},\ P_{w_n}(s'|s)=\frac{\sum_{x'}\boldsymbol{T}(s'|s,x')w_n(s,x')}{\sum_{x''}w_n(s,x'')}.$$

Note that $\sum_{s'} P_w(s'|s) = 1$. Hence it is indeed a conditional probability. Then

$$\begin{aligned} l(w, w_n, \boldsymbol{T}) - I(w, \boldsymbol{T}) &= \sum_{s \in \mathcal{S}} \sum_{s' \in \mathcal{S}} \left(\sum_{x \in \mathcal{X}} w(s,x) \boldsymbol{T}(s'|s,x) \right) \log \frac{P_w(s'|s)}{P_{w_n}(s'|s)} \\ &= \sum_{s \in \mathcal{S}} \sum_{x \in \mathcal{X}} w(s,x) \sum_{s' \in \mathcal{S}} P_w(s'|s) \log \frac{P_w(s'|s)}{P_{w_n}(s'|s)} \\ &\geq 0, \end{aligned}$$

where the inequality follows from the non-negativity of relative entropy. Therefore, $l(w, w_n, \boldsymbol{T})$ is a tangent line of $I(w, \boldsymbol{T})$ at point w_n.

C SYSTEM EXPLANATION AND ALGORITHM DETAILS

C.1 MODEL DESIGN

The concave optimization problem in Theorem 2 characterizes the capacity-reward tradeoff of the action-state channel. It provides an upper bound for practically achievable coding rates under a certain reward constraint. Solving the concave optimization yields the capacity and the optimal state-action distribution ω, which can be translated to a stationary policy for the MDP via the following formula:

$$\pi(a|s) = \frac{\omega(s,a)}{\sum_{a'} \omega(s,a')}.$$

Note that this policy π can not be directly used as the coding policy for communication, as the coding policy needs to generate actions based on both the state and the message. Therefore, in Section 5, we propose Act2Comm to learn a coding policy that mimics the behavior of policy π from the perspective of MDP control, with the input of message and feedback block.

By viewing this problem as a channel coding problem with finite blocklength, historical feedback proves to be beneficial from our prior experience. To effectively map messages and feedback to actions, the transformer architecture with an attention mechanism is naturally well-suited, forming the backbone for both the encoder and decoder components.

To address the challenge of non-differentiability, we propose an iterative training strategy that incorporates a critic network to train the encoder and decoder.

Instead of directly outputting actions at each time step, our encoder generates a continuous belief map, which is subsequently quantized into decision rules. This approach not only enhances performance but also simplifies the training of the critic network.

C.2 ANALYSIS OF EACH SYSTEM COMPONENT

This section provides the outline of each component within the Act2Comm scheme, as summarized in Table. 2.

- The encoder encodes a belief matrix $\boldsymbol{Z}^{(\tau)} \in \mathbb{R}^{\tau \times \frac{\mu|\mathcal{S}|}{R}}$ using message block $\boldsymbol{B}^{(\tau)}$ and feedback block $\boldsymbol{C}^{(\tau)}$ at each coding round τ. After completing all l coding rounds, a final belief map $\boldsymbol{Z} \in \mathbb{R}^{\frac{k}{\mu} \times \frac{\mu|\mathcal{S}|}{R}}$ is constructed.
- A quantizer then generates the codeword $\boldsymbol{U} \in \mathcal{X}^{|\mathcal{S}| \times \frac{k}{R}}$, where each element of the resultant codeword, $x_t = U[s_t, t]$, represents the selected action for state s_t at time step t.
- Given the action and state, EAS channel then return the next states.
- The decoder collects all states and then perform the decoding process, which outputs logits $\hat{\boldsymbol{M}} \in \mathbb{R}^{\frac{k}{\mu} \times 2^{\mu}}$ for the message.
- Since the gradient cannot propagate through the EAS and quantizer to the encoder, a critic network is introduced to link the belief map $\boldsymbol{Z}$ to the decoded logits $\hat{\boldsymbol{M}}$. Specifically, the

Table 2: Functionality of Each System Component

Model components	Function	Input	Output
Encoder	Coding the belief matrix	$\boldsymbol{B}^{(\tau)}, \boldsymbol{C}^{(\tau)}$	$\boldsymbol{Z}$
Quantizer	Generate decision rules	$\boldsymbol{Z}$	$\boldsymbol{U}$
EAS channel	Generate state from actions	$\boldsymbol{U} \rightarrow \boldsymbol{x_t}$	$\boldsymbol{s_{t+1}}$
Decoder	Decoding the message	$\boldsymbol{s}$	$\hat{\boldsymbol{M}} \rightarrow \hat{\boldsymbol{m}}$
Critic Network	Estimate decoding logits	$\boldsymbol{Z_k}$	$\hat{\boldsymbol{M}}_{\boldsymbol{ck}}$

critic network is trained to predict $\hat{\boldsymbol{M}}_k$ from $\boldsymbol{Z}_k$, producing $\hat{\boldsymbol{M}}_{ck}$ at each inner optimization step k within the total s_{in} steps. Thanks to the introduction of the critic network, the gradient can propagate through it, from logits to the belief map. As shown in the table, it effectively "links" the encoder's output to the decoder's output, thereby "skipping" the quantizer, EAS channel and decoder, which are treated as the unknown environment during the training phase.

C.3 QUANTIZER

The quantizer, $\mathcal{Q} : \mathbb{R}^{|\mathcal{S}|\times\frac{k}{R}} \rightarrow \mathcal{X}^{|\mathcal{S}|\times\frac{k}{R}}$, is designed to convert continuous coding results $|\mathcal{X}| \cdot \text{Sigmoid}(\boldsymbol{Z})$ into finite values corresponding to actions within the channel input alphabet set $\mathcal{X} = \{0, 1, \ldots, |\mathcal{X}|-1\}$. For example, if there exist 5 actions and 1 states, then each element of the coding result $|\mathcal{X}| \cdot \text{Sigmoid}(\boldsymbol{Z}) \in (0, 5)$ will be rounded down into the nearest action index. For example, if $|\mathcal{X}| \cdot \text{Sigmoid}(\boldsymbol{Z}) = [1.5, 0.8, 2.1, 3.2, 4.8]$, it will be rounded to $[1, 0, 2, 3, 4]$ as the channel input via the quantizer $\mathcal{Q}$.

Here, we adopt hard rounding, despite the availability of advanced quantization methods such as soft rounding or noise injection during training. The chosen rounding approach is intentionally straightforward, and the non-differentiability introduced by hard rounding is effectively mitigated by a critic network.

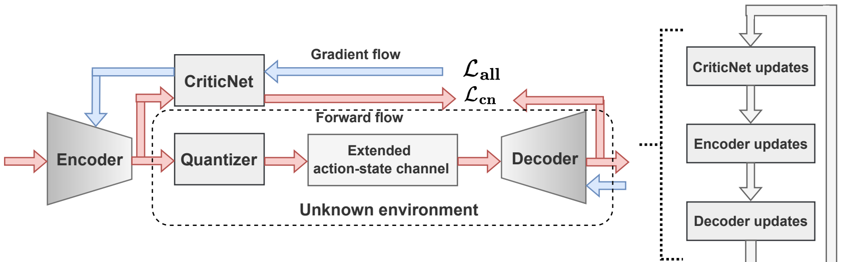
C.4 ITERATIVE TRAINING AND UPDATING

As shown in Fig. 7, a Critic network is introduced to facilitate gradient backpropagation for encoder optimization, treating the quantizer, channel, and decoder as an unknown environment. Before updating the encoder and decoder, the Critic network is trained over s_{in} steps to capture knowledge of the environment, with the encoder and decoder frozen during this phase. Subsequently, the encoder is updated with the assistance of the Critic network, aiming to jointly optimize for control and communication, while keeping the decoder frozen. Following this, the optimized encoder is frozen, and the decoder is updated based on the newly optimized encoder. With the updated decoder, the Critic network is retrained to adapt to the updated environment, preparing it to support the next round of encoder updates. This process iteratively alternates between freezing one component while optimizing the other. From the experiment, we observe that the training process is more stable when the encoder is updated once for every two updates of the decoder. The effectiveness of this approach is visualized in a Fig. 12 and Table. 5 provided in the appendix. The detailed training algorithm is presented in the Algorithm. 7

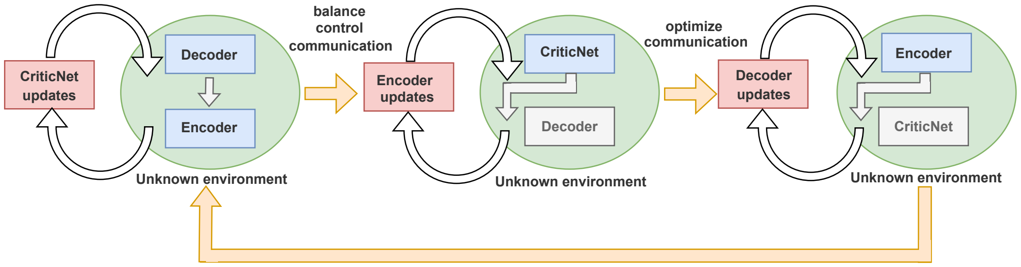
C.5 TRAINING AND INFERENCE ALGORITHM

To enhance readers' understanding of the training and inference process, we provide the pseudocode and illustration figures for *Act2Comm* (Fig. 7, Fig. 8), detailing both the training and inference phases, as shown in Algorithms 1 and 2. Furthermore, Fig. 12 visualizes the training process by tracking the loss values throughout the iterative updates. For additional details about the training, the training logs and source code are also available on the project page of this paper.

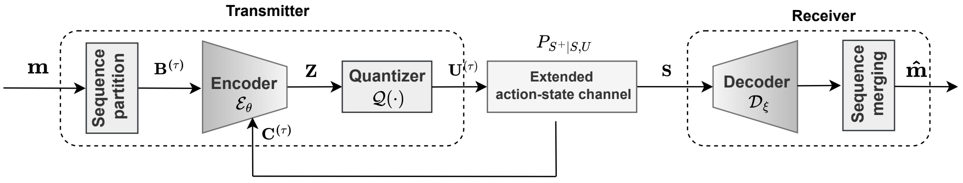
To be more specific for the training phase, we train the encoder first and then train the decoder. The critic network, with control loss, is applied to the encoder. The decoder only considers the communication loss. After training the model, the critic network is removed. Only the encoder and decoder are deployed at the transmitter and receiver for communication and control.



(a). Gradient flow of the Act2Comm



(b). Iterative training strategy



(c). Model workflow when inference

Figure 7: Illustration of the iterative training process: (a) Gradient flow in the proposed method, where blue arrows indicate the gradient flow, and red arrows represent the forward process. (b) Diagram of the update steps, where the red block represents the component being updated, the blue block represents the frozen component, and the grey block indicates an unused component. (c) Inference phase for the well-trained *Act2Comm* model, where the critic network is removed.

C.6 COMPLEXITY ANALYSIS

To analyze the complexity of the proposed method, we consider an input size of 12 bits, 4 blocks, and a rate of $R = 1/3$. The experimental results, presented in Table 3, were obtained using a single GPU-A5000 with $10,000$ runs for the ”Erratic Robot” environment.

From the table, we observe that during the training process, the main computational complexity arises from the critic network, as it needs to run around 20 iterations for each encoder update. Note that the back-propagation FLOPs can be estimated by multiplying the forward FLOPs with a factor (typically 2–3x).

During inference, the critic network is removed. We observe that the encoding process can be completed within 10 ms, while the decoding process is even faster, taking less than 3 ms for the message.

We also analyze the complexity of our approach concerning increasing state and action spaces. Specifically, based on the design of Act2Comm, the number of actions does not impact the scaling performance of the model, in terms of the computational complexity. A larger action space primarily contributes to more diversity in decision rules, without increasing the computational complexity of the approach.

However, environments with more complex action spaces may lead to a more challenging learning process, potentially affecting performance. To validate our method in such scenarios, we introduced

Algorithm 1 Training strategy of the *Act2Comm* scheme

```
Input: The initial MDP state: $s_0$, Inner optimization step: $s_{in} \in \mathbb{R}$,
Learning rates of encoder, decoder, and critic network: $\beta, \alpha, \alpha \in \mathbb{R}$, with $\alpha = 5\beta$.
Given a target policy: $\boldsymbol{\pi}$, temperature parameter $\gamma$, balanced parameter $\lambda$.
Output: Parameters of the encoder and decoder: $\boldsymbol{\theta}$, $\boldsymbol{\xi}$;
for the $m$-th step within the total training steps do
    $\mathbf{m} = [\boldsymbol{b_1}; \boldsymbol{b_2}; \dots; \boldsymbol{b_l}] \in \{0,1\}^{l\times\mu}$,    // Randomly sample and divide the message
    $\mathbf{C}^{(\mathbf{1})} \in 0^{1\times\frac{\mu}{R}\times 3}$,    // Initialize the feedback block
    for each coding round $\tau$ $(1 \le \tau \le l)$ do
        $\mathbf{B}^{(\tau)} = [2\boldsymbol{b_1} - 1; \dots; 2\boldsymbol{b_\tau} - 1] \in \mathbb{R}^{\tau\times\mu}$,    // Construct the message block
        $\mathbf{C}^{(\tau)} = [\boldsymbol{C_1}; \dots; \boldsymbol{C_\tau}] \in \mathbb{R}^{\tau\times\frac{\mu}{R}\times 3}$,    // Update the feedback block
        $\mathbf{Z}^{(\tau)} = \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{B}^{(\tau)}, \mathbf{C}^{(\tau)}) \in \mathbb{R}^{\tau\times\frac{\mu|S|}{R}}$,
        $\mathbf{z}^{(\tau)} = \boldsymbol{Z}^{(\boldsymbol{\tau})}[\tau, :] \in \mathbb{R}^{\frac{\mu|S|}{R}} \to \mathbb{R}^{|S|\times\frac{\mu}{R}}$,    // Encoding the belief map
        $\mathbf{U}^{(\tau)} = \mathcal{Q}(|\mathcal{X}|\text{Sigmoid}(\boldsymbol{z}^{(\boldsymbol{\tau})})) \in \mathcal{X}^{|S|\times\frac{\mu}{R}}$,    // Quantization for decision rules
        for $t = 1:1:\frac{\mu}{R}$ do
            $s_{t+1} = \boldsymbol{T}(s_{t+1}|s_t, \mathbf{U}^{(\tau)}[s_t, t])$,    // Go through the EAS channel for $\frac{\mu}{R}$ time steps
            $\boldsymbol{c}_{\boldsymbol{t}}^{(\boldsymbol{\tau})} = [s_t^{(\tau)}, x_t^{(\tau)}, s_{t+1}^{(\tau)}] \in \mathbb{R}^{1\times 3}$,    // Update the feedback vector
        end for
        $\boldsymbol{C_\tau} = [\boldsymbol{c}_{\boldsymbol{1}}^{(\boldsymbol{\tau})}; \dots; \boldsymbol{c}_{\boldsymbol{\frac{\mu}{R}}}^{(\boldsymbol{\tau})}] \in \mathbb{R}^{\frac{\mu}{R}\times 3}$,    // Update the feedback matrix
    end for
    $\boldsymbol{Z} = [\boldsymbol{z}^{(\mathbf{1})}; \dots; \boldsymbol{z}^{(\boldsymbol{l})}] \in \mathbb{R}^{\frac{k}{\mu}\times\frac{\mu|S|}{R}}$, $\boldsymbol{s} = [s_1; \dots; s_{\frac{k}{R}}] \in \mathcal{S}^{\frac{k}{R}} \to \mathcal{S}^{\frac{k}{\mu}\times\frac{\mu}{R}}$,
    $\hat{\boldsymbol{M}} = \mathcal{D}(\boldsymbol{s}) \in \mathbb{R}^{\frac{k}{\mu}\times 2^{\mu}}$,    // Collect codewords, states, and decode the logits
    ________________________________________    // Train the Critic network
    if $m\%2 == 0$ (each two steps) then
        for $k = 1:1:s_{in}$ do
            $\boldsymbol{Z_k} = \boldsymbol{Z} + \boldsymbol{W_k}$, with $\boldsymbol{W_k} \in \mathbb{R}^{\frac{k}{\mu}\times\frac{\mu|S|}{R}} \sim \mathcal{N}(0, \sigma_w^2)$    // Neighboring sampling
            $\mathbf{U_k} = \mathcal{Q}(|\mathcal{X}|\text{Sigmoid}(\boldsymbol{Z_k})) \in \mathcal{X}^{|S|\times\frac{k}{R}}$,
            for $t = 1:1:\frac{k}{R}$ do
                $s_{t+1} = \boldsymbol{T}(s_{t+1}|s_t, \mathbf{U_{(k)}}[s_t, t])$,    // Go through the EAS channel
            end for
            $\hat{\boldsymbol{s}}_{\boldsymbol{(k)}} = [s_1; \dots; s_{\frac{k}{R}}] \in \mathcal{S}^{\frac{k}{R}} \to \mathcal{S}^{\frac{k}{\mu}\times\frac{\mu}{R}}$,    // Collect the observed states
            $\hat{\boldsymbol{M}}_{\boldsymbol{k}} = \mathcal{D}(\hat{\boldsymbol{s}}_{\boldsymbol{(k)}}) \in \mathbb{R}^{\frac{k}{\mu}\times 2^{\mu}}$,    // Decode the logits with a frozen decoder
            $\hat{\boldsymbol{M}}_{\boldsymbol{ck}} = \mathcal{C}(\boldsymbol{Z_k}) \in \mathbb{R}^{\frac{k}{\mu}\times 2^{\mu}}$    // Predict the logits
            $\mathcal{L}_{cn} = \text{MSE}(\hat{\boldsymbol{M}}_{\boldsymbol{ck}}, \hat{\boldsymbol{M}}_{\boldsymbol{k}})$    // Compute the critic loss with MSE
            $\boldsymbol{\phi_{k+1}} = \boldsymbol{\phi_k} - \alpha\nabla_{\boldsymbol{\phi}}\mathcal{L}_{cn}$    // Update the Critic network.
        end for
        ________________________________________    // Train the Encoder with a frozen Decoder
        $\mathcal{L}_{con} = \text{MSE}(\boldsymbol{\pi}, \hat{\boldsymbol{f}})$    // Control loss term for the estimated $\hat{f}$
        $\mathcal{L}_{com} = \text{cross-entropy}(\boldsymbol{m}, \hat{\boldsymbol{M}})$    // Communication loss term
        $\boldsymbol{\theta} = \boldsymbol{\theta} - \alpha\nabla_{\boldsymbol{\theta}}(\mathcal{L}_{com} + \lambda\mathcal{L}_{con})$    // Update the encoder
    end if
    ________________________________________    // Train the Decoder with a frozen Encoder
    for each coding round and corresponding time step do
        $\mathbf{z}^{(\tau)} = \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{B}^{(\tau)}, \mathbf{C}^{(\tau)})[\tau, :] \to \boldsymbol{U}^{(\boldsymbol{\tau})}$,    // Re-encode and re-quantize the decision rules
        $s_{t+1} = \boldsymbol{T}(s_{t+1}|s_t, \mathbf{U}^{(\tau)}[s_t, t])$,    // Re-go through the EAS channel for all time steps
    end for
    $\boldsymbol{s} = [s_1; \dots; s_{\frac{k}{R}}]$, $\hat{\boldsymbol{M}} = \mathcal{D}(\boldsymbol{s})$    // Decode the observed states from a forzen encoder
    $\mathcal{L}_{com} = \text{cross-entropy}(\boldsymbol{m}, \hat{\boldsymbol{M}})$    // Communication loss term
    $\boldsymbol{\xi} = \boldsymbol{\xi} - \beta\nabla_{\boldsymbol{\xi}}\mathcal{L}_{com}$    // Update the encoder
end for
```

Algorithm 2 Inference of the *Act2Comm* scheme

Input and output for the encoding:
Input: Initial state: s_0, k-bits message $\mathbf{m}$, coding rate R, well-trained encoder: $\boldsymbol{\theta}^*$;
Output: Codeword (decision rules) $\boldsymbol{Z} \in \mathbb{R}^{|S| \times \frac{k}{R}}$.

Input and output for the decoding:
Input: Observed states $\boldsymbol{s} \in \mathcal{S}^{\frac{k}{R}}$, well-trained decoder $\boldsymbol{\xi}^*$;
Output: Codeword (decision rules) $\boldsymbol{Z} \in \mathbb{R}^{|S| \times \frac{k}{R}}$, predicted message: $\hat{\boldsymbol{m}}$.

```
Encoding phase:
$\mathbf{C}^{(\mathbf{1})} \in 0^{1 \times \frac{\mu}{R} \times 3}$,    // Initialize the feedback block
for each coding round $\tau$ $(1 \le \tau \le l)$ do
    $\mathbf{B}^{(\tau)} = [2\boldsymbol{b_1} - 1; 2\boldsymbol{b_2} - 1; \ldots; 2\boldsymbol{b_\tau} - 1] \in \mathbb{R}^{\tau \times \mu}$,    // Construct the message block
    $\mathbf{C}^{(\tau)} = [\boldsymbol{C_1}; \ldots; \boldsymbol{C_\tau}] \in \mathbb{R}^{\tau \times \frac{\mu}{R} \times 3}$,    // Update the feedback block
    $\mathbf{Z}^{(\tau)} = \mathcal{E}_{\boldsymbol{\theta}}(\mathbf{B}^{(\tau)}, \mathbf{C}^{(\tau)}) \in \mathbb{R}^{\tau \times \frac{\mu|S|}{R}}$,
    $\mathbf{z}^{(\tau)} = \boldsymbol{Z}^{(\boldsymbol{\tau})}[\tau, :] \in \mathbb{R}^{\frac{\mu|S|}{R}} \to \mathbb{R}^{|S| \times \frac{\mu}{R}}$,    // Encoding the belief map
    $\mathbf{U}^{(\tau)} = \mathcal{Q}(|\mathcal{X}|\text{Sigmoid}(\boldsymbol{Z})) \in \mathcal{X}^{|S| \times \frac{\mu}{R}}$,    // Quantization for decision rules
    for $t = 1 : 1 : \frac{\mu}{R}$ do
        $s_{t+1} = \boldsymbol{T}(s_{t+1} | s_t, \mathbf{U}^{(\tau)}[s_t, t])$,    // Go through the EAS channel for $\frac{\mu}{R}$ time steps
        $\boldsymbol{c}_{\boldsymbol{t}}^{(\boldsymbol{\tau})} = [s_t^{(\tau)}, x_t^{(\tau)}, s_{t+1}^{(\tau)}] \in \mathbb{R}^{1 \times 3}$,    // Update the feedback vector
    end for
    $\boldsymbol{C_\tau} = [\boldsymbol{c}_{\boldsymbol{1}}^{(\boldsymbol{\tau})}; \ldots; \boldsymbol{c}_{\boldsymbol{\frac{\mu}{R}}}^{(\boldsymbol{\tau})}] \in \mathbb{R}^{\frac{\mu}{R} \times 3}$,    // Update the feedback matrix
end for
$\boldsymbol{Z} = [\boldsymbol{z}^{(\mathbf{1})}; \ldots; \boldsymbol{z}^{(\boldsymbol{l})}] \in \mathbb{R}^{\frac{k}{\mu} \times \frac{\mu|S|}{R}} \to \mathbb{R}^{|S| \times \frac{k}{R}}$,    // Collect the final codeword

Decoding phase:
$\boldsymbol{s} = [s_1; \ldots; s_{\frac{k}{R}}] \in \mathcal{S}^{\frac{k}{R}} \to \mathcal{S}^{\frac{k}{\mu} \times \frac{\mu}{R}}$,    // Collect the observed states
$\hat{\boldsymbol{M}} = \mathcal{D}(\boldsymbol{s}) \in \mathbb{R}^{\frac{k}{\mu} \times 2^{\mu}}$,    // Decode the logits
$\hat{\boldsymbol{m}} = \text{argmax}(\text{Softmax}(\hat{\boldsymbol{M}}), \text{dim} = -1)$    // Decode each block labels
```

a new environment, the "Erratic Robot," which features a five-action space. Results demonstrate that our method remains effective, even in this more complex setting.

For increased state spaces, the complexity grows only in the encoder component. This occurs because the computational complexity of the encoder increases as the output matrix size expands with the number of states. However, we consider this increase acceptable within our framework. Because, this complexity growth is analogous to the increased dimensionality from lower communication rates, which Act2Comm handles effectively.

D EXPERIMENTAL DETAILS

D.1 EXPERIMENT SETUP

For the *Act2Comm* scheme, we train the model with a batch size of 4096, a learning rate of 0.001, and an Adam-based lookahead optimizer (Zhang et al., 2019). The inner-training step for the critic network is $s_{in} = 20$, with a noise variance of $\sigma_w^2 = 0.1$. Each block has a length of $\mu = 3$, and temperature parameter is as $\gamma = 10$, $\gamma = 50$, $\gamma = 100$, $\gamma = 200$. The performance presented is averaged over $20,000$ execution times. To investigate the trade-offs, we train the *Act2Comm* model with $\lambda \in [0.01, 20]$.

The detailed architecture of the *Act2Comm* scheme is provided in Fig. 9. At the transmitter, a transformer-based encoder is utilized to generate the $\boldsymbol{z}^{(\boldsymbol{\tau})}$ from $\boldsymbol{B}^{(\boldsymbol{\tau})}$ and $\boldsymbol{C}^{(\boldsymbol{\tau})}$ for each coding round. Specifically, $\boldsymbol{C}^{(\tau)}$ and $\boldsymbol{B}^{(\boldsymbol{\tau})}$ are firstly processed through multilayer perceptron (MLP) layers, then concatenated for linear projection and positional encoding operations, resulting in $\boldsymbol{F_0} \in$

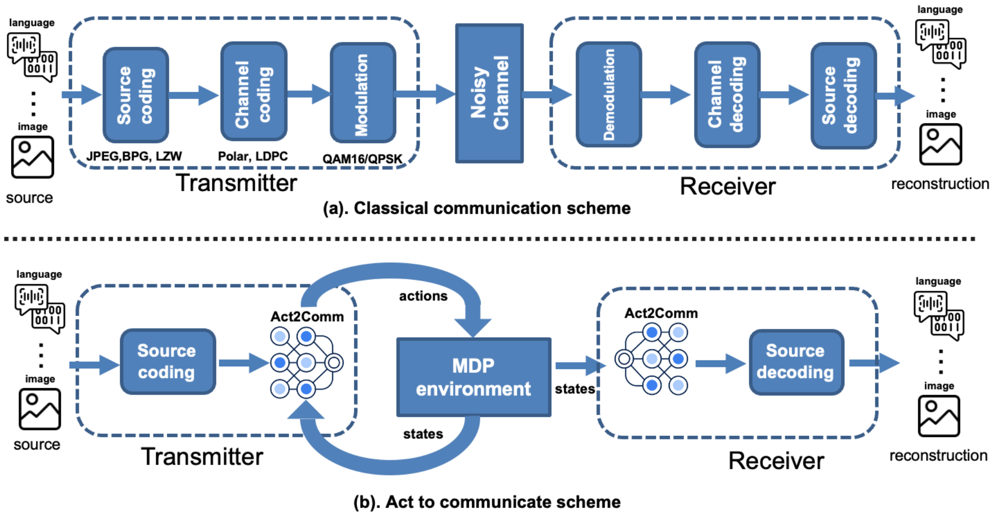


Figure 8: An example of practical implementation of the proposed *Act2Comm*.

Table 3: Number of parameters, FLOPs, and coding times for the *Act2Comm* scheme ($K = 12$, $\mu = 3$, $R = 1/3$, $s_{in} = 20$) with varying critic network training steps s_{in} for both training and inference phases. Note that the backpropagation computational complexity is approximately 2–3 times that of the forward computation.

Training Phase (forward)					
Model components	**Encoder**	**Decoder**	**CrititNet** $s_{in} = 5$	**CriticNet** $s_{in} = 10$	**CriticNet** $s_{in} = 20$
Parameters (k)	32.244	50.536	7.568	7.568	7.568
FLOPs (millions)	0.3164	0.1989	0.1454	0.2909	0.5818

Inference Phase			
Model components	**Encoder**	**Decoder**	**Critic network**
Parameters (k)	32.244	50.536	✗
FLOPs (millions)	0.3164	0.1989	✗
Coding Time (ms)	6.691	2.613	✗

$\mathbb{R}^{\tau \times d}$, where d is the hidden layer dimension. After L_t transformer layers, the resultant $\boldsymbol{F}_{\boldsymbol{L_t}} \in \mathbb{R}^{\tau \times d}$ is further transformed by a fully-connected layer into $\boldsymbol{Z}^{(\tau)}$. The decoding process begins with reshaping $\boldsymbol{S}$ into $\mathcal{S}^{\frac{k}{\mu} \times \frac{\mu}{R}}$. This reshaped $\boldsymbol{S}$ is then processed symmetrically through fully-connected layers, positional encoding, and L_r transformer layers, yielding the feature $\boldsymbol{D}_{\boldsymbol{L_r}} \in \mathbb{R}^{\frac{k}{\mu} \times d}$. Subsequently, $\boldsymbol{D}_{\boldsymbol{L_r}}$ is input into a fully-connected layer to generate the logits $\hat{\boldsymbol{M}} \in \mathbb{R}^{l \times 2^{\mu}}$ for each block. After a softmax function, we predict the label of $\boldsymbol{m}$ and transform it into bit stream $\hat{\boldsymbol{m}}$. Specifically, we set $d = 32$, $L_t = 2$ and $L_t = 4$ during the experiments.

D.2 LUCKY WHEEL

As illustrated in Fig. 10a, the wheel is evenly divided into three regions. At each time step, the player can select either action 0 or action 1, where action 0 corresponds to a clockwise rotation of the wheel, and action 1 corresponds to an anti-clockwise rotation. If action 0 is selected, there is a probability $p = 0.2$ that the pointer remains in its current region, and a probability $1 - p = 0.8$ that it moves to the next region in the clockwise direction. Similarly, if action 1 is selected, there is a probability $p = 0.2$ that the pointer remains in its current region, and a probability $1 - p = 0.8$ that it moves to the next region in the anti-clockwise direction. The rewards for the three regions are 5, -5, and 0, respectively. The player receives a reward at each time step based on the region in which the pointer is located.

Table 4: Number of parameters, FLOPs, and coding times for the *Act2Comm* scheme ($K = 12$, $\mu = 3$, $R = 1/3$, $s_{in} = 20$) with varying state numbers and action numbers, where the increased value are colored with orange and red.

Action and state number	**Encoder**			**Decoder**		
$(\lvert\mathcal{A}\rvert, \lvert\mathcal{S}\rvert)$ with $\lvert\mathcal{A}\rvert \uparrow$	$(5, 16)$	$(20, 16)$	$(40, 16)$	$(5, 16)$	$(20, 16)$	$(40, 16)$
Parameters (k)	32.244	32.244	32.244	50.536	50.536	50.536
FLOPs (millions)	0.3164	0.3164	0.3164	0.1989	0.1989	0.1989
Coding Time (ms)	6.691	6.691	6.691	2.613	2.613	2.613
$(\lvert\mathcal{A}\rvert, \lvert\mathcal{S}\rvert)$ with $\lvert\mathcal{S}\rvert \uparrow$	$(5, 16)$	$(5, 64)$	$(5, 256)$	$(5, 16)$	$(5, 64)$	$(5, 256)$
Parameters (k)	32.244	46.51	103.52	50.536	50.536	50.536
FLOPs (millions)	0.3164	0.4546	1.007	0.1989	0.1989	0.1989
Coding Time (ms)	6.691	6.69	7.04	2.613	2.613	2.613

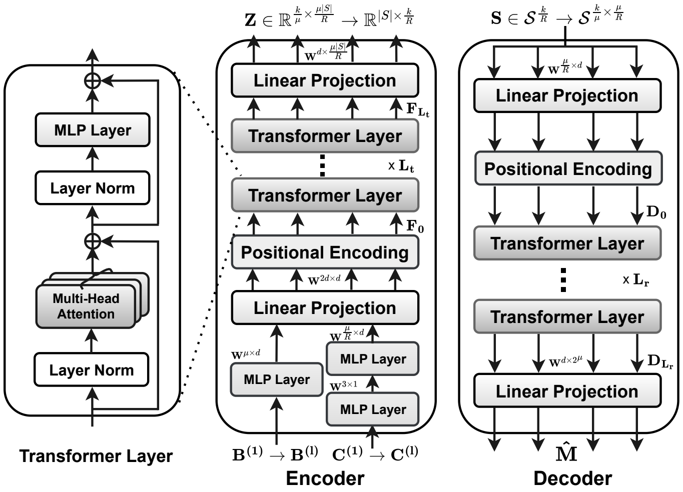

$\mathbf{Z} \in \mathbb{R}^{\frac{k}{\mu} \times \frac{\mu|S|}{R}} \to \mathbb{R}^{|S| \times \frac{k}{R}}$
$\mathbf{S} \in \mathcal{S}^{\frac{k}{R}} \to \mathcal{S}^{\frac{k}{\mu} \times \frac{\mu}{R}}$
MLP Layer
Layer Norm
Multi-Head Attention
Layer Norm
Transformer Layer
$\mathbf{W}^{d \times \frac{\mu|S|}{R}}$
Linear Projection
$\mathbf{F}_{\mathbf{L_t}}$
Transformer Layer
x $\mathbf{L_t}$
Transformer Layer
$\mathbf{F_0}$
Positional Encoding
$\mathbf{W}^{2d \times d}$
Linear Projection
$\mathbf{W}^{\mu \times d}$
$\mathbf{W}^{\frac{\mu}{R} \times d}$
MLP Layer
MLP Layer
$\mathbf{W}^{3 \times 1}$
MLP Layer
$\mathbf{B}^{(1)} \to \mathbf{B}^{(l)}$
$\mathbf{C}^{(1)} \to \mathbf{C}^{(l)}$
Encoder
$\mathbf{W}^{\frac{\mu}{R} \times d}$
Linear Projection
Positional Encoding
$\mathbf{D_0}$
Transformer Layer
x $\mathbf{L_r}$
Transformer Layer
$\mathbf{W}^{d \times 2^{\mu}}$
$\mathbf{D}_{\mathbf{L_r}}$
Linear Projection
$\hat{\mathbf{M}}$
Decoder

Figure 9: The architecture of the *Act2Comm*.

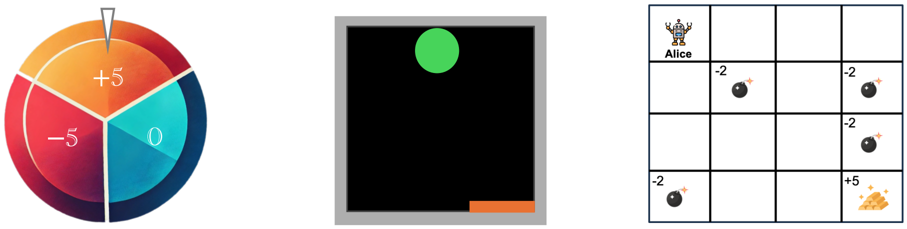
D.3 CATCH THE BALL

The "Catch the Ball" game is set in a 3×3 grid, as illustrated in Fig. 10b. A ball randomly appears at the top of the grid and descends one grid space at each time step. Meanwhile, a board at the bottom moves horizontally to catch the falling ball. Each time the board successfully catches a ball, the player receives a reward r. If the ball falls to the bottom of the grid without being caught, it disappears, resulting in a penalty of $-r$ for the player. At all other times, there are no rewards or penalties for the player. After a ball disappears or is caught, a new ball appears randomly (with equal probability) at one of the three positions at the top of the grid.

The player has three available actions to move the board: move left, move right, or remain stationary. If the player chooses to move left or right, there is a probability $p \in [0, 1)$ that the movement fails (in which case the board remains in its current position), and a probability $1-p$ that the board moves for one grid space successfully as intended. Additionally, any action that attempts to move the board outside the grid will always fail. In this experiment, we set $r = 5$ and $p = 0.8$.

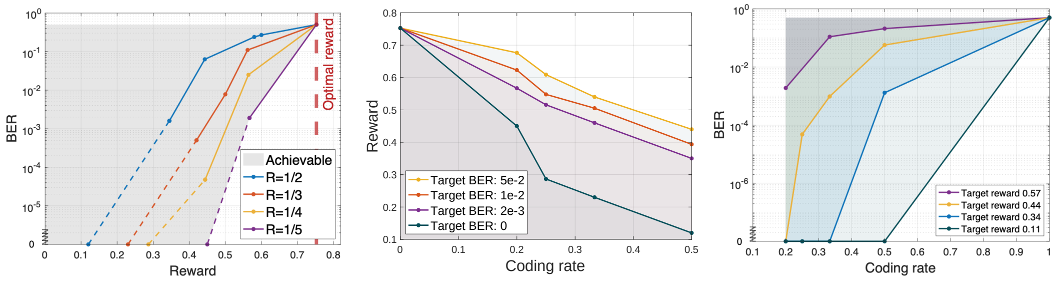
D.4 ERRATIC ROBOT.

The 'Erratic robot' game takes place on a 4×4 grid map with 16 states and 5 actions, as shown in Fig. 10c. The robot is designed to minimize the number of steps required to collect goods while avoiding obstacle points. Specifically, its primary objective is to continuously take goods from designated destination points, earning a reward of $+5$ for each successful collection. The grid also includes four obstacle points, each incurring a penalty of -2 when encountered. The robot has five available actions: move left, move right, move up, move down, or remain stationary. Due to the instability,



(a) Lucky wheel. (b) Catch the Ball. (c) Erratic robot.

Figure 10: Illustrations of experimental MDP environments



(a) BER v.s. Reward (b) Reward v.s. Rate (c) BER v.s. Rate

Figure 11: Control-communication trade-off of *Act2Comm* in “Erratic Robot”.

any movement of the robot has a probability $p \in [0, 1)$ of resulting in an additional unintended step. Actions that would move the robot outside the grid boundaries always fail. In this experiment, the probability of slippage is set to $p = 0.2$. The results are presented in the Fig. 11

As shown in the Fig. 11, the proposed *Act2cComm* scheme can achieve perfect communication with $R = 1/5$, albeit with a reduction in the average reward from the optimal value of 0.753 to 0.45. If we relax BER, the same rate can be achieved with a better reward. Similarly, for different reward targets, we can obtain different communication performances. This additional experiment also highlights the versatility of the proposed *Act2Comm* scheme across various environments.

E ADDITIONAL ABLATION EXPERIMENTS

This section presents additional ablation experiments for our proposed *Act2Comm* framework, using the lucky wheel environment as the default experimental setting with an initial state 0.

E.1 ABLATION STUDY OVER DIFFERENT MECHANISMS

To showcase the efficiency of our approach, we present the loss values monitored throughout the iterative training process in Fig. 12. While the loss value occasionally fluctuates during the training updates—primarily due to the alternating update scheme between the decoder and encoder—overall, the model exhibits a rapid and consistent convergence.

Additionally, to further validate the performance of our Critic network, we plot the estimated loss values generated by the Critic alongside the true loss during training. As shown in Fig. 12, the Critic network maintains accurate loss estimation throughout the training phase.

We present a detailed comparative analysis of each mechanism within our method, as shown in Table 5, emphasizing the specific contributions of each component to the cumulative performance improvements. Notably, utilizing the Critic network to predict the loss directly, or eliminating policy noise during inner-step training can yield failure, underscoring the critical importance of appropriate policy noise and neighboring sampling mechanisms.

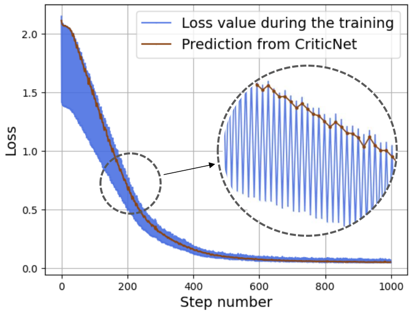
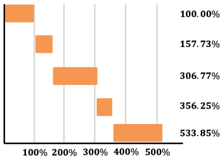
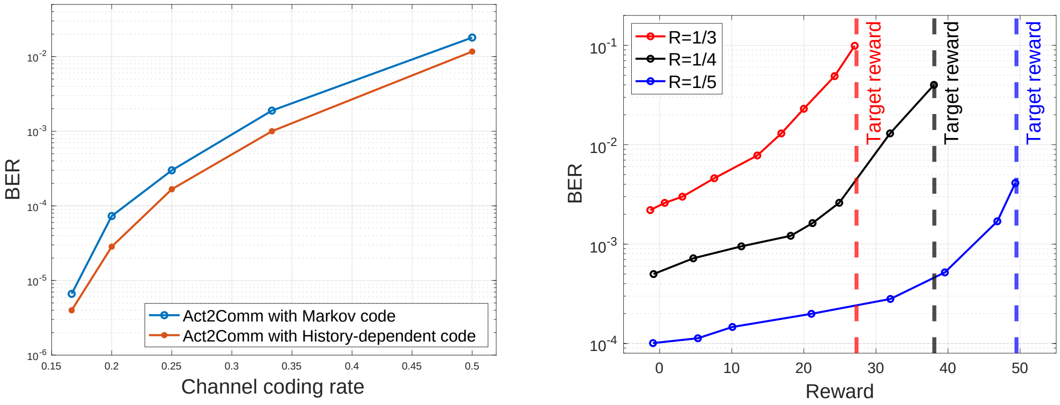


Figure 12: Loss during the training process, where the loss value decreases significantly with each decoder update, while showing a slight increase with each encoder update.



Model Variant	**BER performance vs. *Act2Comm***
Act2Comm scheme	100.00%
✗ Feedback coding	157.73%
✗ Appropriate policy noise	306.77%
✗ Block-wise coding	356.25%
✗ Lookahead optimizer	533.85%
✓ CriticNet for loss	Not work
✓ Perfect policy	Not work

Table 5: Ablation study when sequentially removing one improvement after another. Note higher BER means worse performance for a given reward here with $R = 1/4$.



(a)

(b)

Figure 13: Left: Performance for different coding scheme. Right: Approaching a given policy $\overline{\pi}$.

all these design elements collectively ensure that our proposed solution consistently delivers robust performance across a wide range of scenarios.

E.2 ABLATION STUDY OVER THE FEEDBACK DESIGN

Act2Comm supports both history-dependent coding, which encodes both message and feedback blocks, and Markov coding, which encodes only the message block. We examine *Act2Comm* scheme with both History-dependent codes and Markov codes across various coding rates R for a pure communication optimization.

The experimental results are depicted in Fig. 13a. It is evident that the BER performance improves significantly as the channel coding rate decreases. Specifically, with a coding rate $R = 1/6$, the BER reaches 10^{-6} level, indicating a significant enhancement in performance due to the increased number of channel uses. Comparing history-dependent codes with Markov codes reveals that incorporating feedback blocks can yield substantial performance improvements. Although, from a theoretical perspective, channel feedback does not enhance the capacity of a memoryless channel, the observed benefits may be attributed to the attention mechanism applied to historical states and their corresponding actions. This mechanism effectively utilizes prior coding experience to simplify the subsequent coding process.

E.3 ABLATION STUDY OVER THE MESSAGE LENGTH

Additional experiments with *Act2Comm* are conducted across different message lengths and coding rates, as presented in Table. 6. The results demonstrate that *Act2Comm* scheme consistently delivers competitive performance over different message lengths and coding rates. An interesting observation is that communication performance generally improves with longer block lengths; however, it eventually declines due to the increased learning complexity. Notably, for longer message lengths k, adapting a larger block size μ can potentially enhance performance while addressing the increased learning complexity.

Coding rate	$k = 12$	$k = 24$	$k = 36$
$R = 1/2$	$1.16e^{-2}$	$1.06e^{-2}$	$9.15e^{-3}$
$R = 1/3$	$1.06e^{-3}$	$1.02e^{-3}$	$1.14e^{-3}$
$R = 1/5$	$2.84e^{-5}$	$1.62e^{-5}$	$2.06e^{-5}$

Table 6: BER for *Act2Comm* across different message length and coding rate, where $\mu = 3$.

E.4 ABLATION STUDY OVER A SPECIFIC POLICY

In previous part of this paper, we focused solely on the optimal strategy as the target policy for training, aiming to optimize the control objective. However, our approach is versatile and can accommodate a broad spectrum of policies—ranging from optimal to specialized strategies—enabling users to employ the system to approach specific policies for their own tasks.

In the lucky wheel environment, now we consider a target policy $\bar{\boldsymbol{\pi}}$ given as:

$$\bar{\boldsymbol{\pi}} = \begin{bmatrix} \pi(x=0|s=0) & \pi(x=1|s=0) \\ \pi(x=0|s=1) & \pi(x=1|s=1) \\ \pi(x=0|s=2) & \pi(x=1|s=2) \end{bmatrix} = \begin{bmatrix} 0.7 & 0.3 \\ 0.3 & 0.7 \\ 0.7 & 0.3 \end{bmatrix}. \tag{23}$$

We evaluate the *Act2Comm* scheme with different coding rates and consider a single message transmission with an accumulated reward. The experimental results in Fig. 13b display the curve representing the lower envelope of all possible BER-reward trade-off outcomes for each coding rate. This curve illustrates that the region above it is achievable by our *Act2Comm* scheme.

Notably, our method can achieve good communication performance, approximately $4 * 10^{-1}$, $4 * 10^{-2}$, and $3 * 10^{-3}$ for $R = 1/3$, $R = 1/4$, and $R = 1/5$ respectively, with almost no loss in control performance. This is due to the stochastic nature of our target policy, as opposed to a deterministic one, allowing our method to adaptively learn a similar policy that maintains comparable rewards while being more favorable for channel coding. In summary, the experimental results demonstrate that our *Act2Comm* framework achieves satisfactory communication performance while maintaining the reward, as defined by a specific policy, at a certain level in scenarios such as those involving stochastic policies.