Communicating Plans, Not Percepts: Scalable Multi-Agent Coordination with Embodied World Models

Brennen A. Hill*
Department of Computer Science
University of Wisconsin-Madison
Madison, WI 53706
bahill4@wisc.edu

Mant Koh En Wei
Department of Computer Science
National University of Singapore
Singapore 119077
e0958776@u.nus.edu

Thangavel Jishnuanandh
Department of Computer Science
National University of Singapore
Singapore 119077
jishnuanandh@u.nus.edu

Abstract

The scaling of environmental complexity is a critical frontier for advancing multi-agent intelligence. As environments grow in size, dimensionality, and partial observability, agents require sophisticated coordination mechanisms to maintain performance. This paper investigates the role of communication in such scaled environments by comparing two distinct strategies in a cooperative multi-agent reinforcement learning (MARL) task: an emergent protocol and an engineered, intention-based protocol. For the emergent approach, we introduce Learned Direct Communication (LDC), where agents equipped with distinct neural network weights learn to generate and interpret messages end-to-end. For the engineered approach, we propose Intention Communication, a structured architecture featuring an Imagined Trajectory Generation Module (ITGM) and a Message Generation Network (MGN) that allows agents to explicitly formulate and share forward-looking plans. The ITGM uses an internal world model and the agent's own policy to generate and share *planned* future trajectories. We evaluate these strategies in a partially observable grid world, progressively scaling the environment's size. Our findings reveal that while emergent communication is viable in simpler settings, its performance degrades sharply as the environment scales. In contrast, the engineered Intention Communication approach demonstrates remarkable robustness, sample efficiency, and high performance, maintaining near-optimal success rates even in significantly larger and more challenging environments. This work underscores that for agents to succeed in increasingly complex, scaled-up interactive settings, structured and explicit coordination mechanisms may be fundamentally more scalable than purely emergent protocols.

1 Introduction

The development of general-purpose intelligent agents is deeply intertwined with the environments they inhabit. Environments are not merely passive backdrops for evaluation; they are the very data

*Corresponding author.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Scaling Environments for Agents (SEA).

from which agents learn adaptive behaviors, complex reasoning, and long-term planning [Sutton and Barto, 2018]. Echoing the successes seen in scaling models and datasets for Large Language Models (LLMs), the community now recognizes that scaling the structure, fidelity, and diversity of environments is a crucial vector for advancing agent capabilities [Cobbe et al., 2020]. This is particularly true in the context of multi-agent reinforcement learning (MARL), where the environment's dynamics are compounded by the simultaneous actions of multiple learning agents.

A central challenge in MARL is non-stationarity: from any single agent's perspective, the environment is constantly changing as its peers adapt their policies [Kefan et al., 2024]. This moving-target problem can destabilize learning and hinder effective coordination, especially under conditions of partial observability where agents have incomplete information about the global state [Oroojlooyjadid et al., 2023]. Communication offers a powerful mechanism to mitigate these challenges by enabling agents to share information, synchronize actions, and form coherent joint plans [Ming et al., 2024].

This raises a fundamental question: how should communication protocols be designed? One school of thought champions *emergent communication*, where protocols are learned end-to-end without human-specified structure, holding the promise of discovering novel and highly adaptive signaling systems [Foerster et al., 2016]. An alternative approach advocates for *engineered communication*, where inductive biases are embedded into agent architectures to facilitate the exchange of specific types of information, such as goals or intentions, promoting stability and sample efficiency [Qiu et al., 2024].

The choice between these paradigms becomes critical as we consider scaling environments for agents. An effective communication protocol must not only function in a static environment but also remain robust as the state-action space expands and the task difficulty increases. This paper directly addresses this issue by conducting a rigorous comparative study of learned versus engineered communication in the context of a scalable cooperative task.

1.1 Problem and Contributions

We focus on a cooperative navigation and task allocation problem where two agents in a grid environment must coordinate to occupy two distinct goal locations. We scale the environment along two primary axes: (1) **Observability**, transitioning from a fully observable to a partially observable setting, and (2) **Spatial Complexity**, by systematically increasing the size of the grid.

Our contributions are threefold:

1. We propose and implement two distinct communication architectures: **Learned Direct Communication (LDC)**, where a binary message is learned end-to-end as part of the policy output, and **Intention Communication**, a novel engineered architecture that uses an internal world model to generate and share imagined future trajectories.
2. We conduct a systematic evaluation comparing these two approaches against a non-communicating baseline, demonstrating how their performance changes as the environment scales in size and partial observability.
3. Our results provide strong evidence that while emergent communication can be effective in simple settings, the structured, forward-looking nature of engineered Intention Communication leads to vastly superior performance, scalability, and sample efficiency in more complex environments. This suggests that incorporating architectural priors for communication is a crucial design principle for developing agents capable of tackling large-scale coordination problems.

This work directly engages with environment infrastructure design, benchmarks for generalization, and policy learning through interaction. By analyzing how different agent architectures cope with environmental scaling, we provide insights into building more robust and capable multi-agent systems.

2 Related Work

2.1 Communication in Multi-Agent Reinforcement Learning

Communication has long been identified as a key component for successful coordination in MARL. Early approaches enabled communication channels through which agents could pass information. Sukhbaatar et al. [2016]'s CommNet introduced a continuous communication vector, averaging all

agents' messages to provide a shared context for action selection. Foerster et al. [2016] proposed Differentiable Inter-Agent Learning (DIAL), allowing gradients to flow between agents through communication channels during training, treating communication as part of the end-to-end learning process. More recent methods like MADDPG [Lowe et al., 2017] often use centralized training with decentralized execution, where a centralized critic has access to all agents' actions and observations, implicitly easing the coordination problem without explicit communication at runtime. Our LDC model follows in the spirit of DIAL, focusing on a fully decentralized, end-to-end learned protocol.

2.2 Emergent versus Structured Communication

The study of emergent communication seeks to understand how meaningful signaling systems can arise from scratch among autonomous agents trying to solve a cooperative task [Mordatch and Abbeel, 2018]. While fascinating, research has shown that emergent protocols can be brittle, sensitive to initial conditions, and often lack the compositional structure of human language, making them difficult to interpret and generalize [Kottur et al., 2017].

In contrast, structured communication methods embed inductive biases about what information is useful to share. This often involves sharing specific, semantically meaningful information. For example, agents might share their intended goals [Liu et al., 2021], action plans, or value function estimates. Our Intention Communication model falls into this category. By designing the architecture to explicitly generate and communicate a representation of a future plan, we inject a strong prior that forward-looking information is key to coordination. This aligns with work on sharing latent representations of plans [Li et al., 2025] and using model-based rollouts to guide policy decisions [Liu et al., 2024].

2.3 Scaling Environments and Benchmarks

The MARL community has increasingly moved towards more complex and challenging environments to drive progress. Benchmarks like the StarCraft Multi-Agent Challenge (SMAC) [Samvelyan et al., 2019], Google Research Football [Kurach et al., 2020], and Overcooked-AI [Carroll et al., 2019] provide complex, high-dimensional, and partially observable settings that demand sophisticated coordination. Our work uses a conceptually simpler grid world, but does so with the explicit purpose of creating a controlled testbed. This allows us to precisely isolate and study the impact of specific scaling dimensions, namely spatial size and observability, on the efficacy of different communication strategies, providing a clear and interpretable case study relevant to the broader challenge of scaling environments.

3 Problem Formulation and Environment

We formalize our multi-agent task as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) [Bernstein et al., 2002], defined by the tuple $\langle \mathcal{I}, S, \{A_i\}_{i\in\mathcal{I}}, T, R, \{\Omega_i\}_{i\in\mathcal{I}}, O \rangle$.

- $\mathcal{I}$ is the set of $N = 2$ agents.
- S is the set of environment states, defining the coordinates of both agents and both goals.
- A_i is the action set for agent i, consisting of five discrete actions: {stay, up, down, left, right}. The joint action is $A = \times_{i\in\mathcal{I}} A_i$.
- $T : S \times A \to S$ is the deterministic state transition function.
- $R : S \times A \to \mathbb{R}$ is the shared reward function.
- Ω_i is the observation set for agent i.
- $O : S \times A \to \times_{i\in\mathcal{I}} \Omega_i$ is the observation function, which maps the true state to local observations for each agent.

The collective goal is to learn a set of policies $\{\pi_i(a_i|o_i)\}_{i\in\mathcal{I}}$ that maximizes the expected discounted cumulative reward, $J(\pi) = \mathbb{E}[\sum_{t=0}^{T_{max}} \gamma^t R(s_t, a_t)]$.

3.1 Environment Details

To isolate the effects of communication strategies, we use a deterministic grid world implemented using the PettingZoo library [Terry et al., 2021]. The environment consists of an $x \times x$ grid containing two agents and two goals. All experiments were conducted on Google Colab [Google, 2025],

imposing significant computational constraints. This constraint motivated the development of models that are not only effective but also highly sample-efficient.

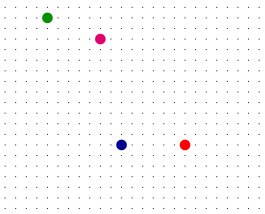

Figure 1: The experimental grid world setup, showing two agents (red and pink) and two goals (blue and green). Agents must coordinate to occupy distinct goals.

3.1.1 Observation and Action Spaces

- **Action Space:** As noted, each agent has a discrete action space of size 5.
- **Observation Space:** In the fully observable setting, each agent receives the coordinates of both goals. The observation for agent i at time t is a 4-tuple of relative distances $o_t^{(i)} = \left[x_{goal1},\, y_{goal1},\, x_{goal2},\, y_{goal2}\right] \in [0, x)^4$. In the partially observable setting, an agent observes a goal's location only if it is within a specified Euclidean distance, `vision_range`. If a goal is out of range, its coordinates are masked with a value of 0.

To provide agents with short-term memory and a sense of motion, we stack observations from the last F frames, yielding an input vector of dimension $4 \times F$ for each agent's policy network. To prevent agents from learning trivial static policies (e.g., agent 1 always goes to goal 1), the order of goals in the observation vector is randomized for one of the agents in each episode.

3.1.2 Initialization, Termination, and Rewards

At the start of each episode, agents and goals are placed uniformly at random on distinct cells. An episode terminates successfully if both agents occupy different goals. It terminates with failure if a maximum of T_{max} steps is reached. The shared reward signal is designed to encourage cooperation:

- $+\mathbf{1.0}$ for the success condition (agents on two distinct goals).
- $-\mathbf{0.10}$ for collision (agents on the same goal).
- $-\mathbf{0.01}$ per-step penalty to encourage efficient paths.

3.2 Training Framework

We use an on-policy, actor-critic algorithm based on Advantage Actor-Critic (A2C) with a learned baseline [Sutton and Barto, 2018]. For all models, the agent architecture consists of a shared feature extraction body, implemented as an MLP with N_{hidden} hidden layers of D_{hidden} units and ReLU activations. After this shared body, the network splits into two separate heads: a policy (actor) head and a value (critic) head. These networks are updated at the end of each episode.

The agent's total policy π is composed of a physical action policy π_{act} and a message policy π_{msg}. The actor loss L_{actor} applies the policy gradient to both sampled outputs, weighted by the advantage $A_t = R_t + \gamma V(s_{t+1}) - V(s_t)$. The message m_t is a composite vector $\langle m_t^{(1)}, \ldots, m_t^{(N_{comp})} \rangle$ of N_{comp} discrete symbols.

$$L_{\text{actor}} = -A_t \left(\log \pi_{act}(a_t|\cdot) + \beta \sum_{k=1}^{N_{comp}} \log \pi_{msg}^{(k)}(m_t^{(k)}|\cdot) \right)$$

The total loss is a combined actor-critic loss: $L = L_{\text{actor}} + \alpha_c \times L_{\text{critic}}$, where β is a hyperparameter balancing the importance of the message policy.

To balance exploration and exploitation, we employ a linear learning rate decay schedule:

$$\text{lr}_{\text{ep}} = \text{lr}_0 \left(1 \;-\; \tfrac{\text{episode}}{N_{\text{total}}}\right)$$

This schedule, clipped at a minimum value (lr_f), permits larger policy updates early in training and smaller, fine-tuning updates later, which helps prevent catastrophic forgetting of successful strategies. Key hyperparameters are listed in Table 6 in the appendix.

4 Approach 1: Learned Direct Communication (LDC)

Our first approach investigates whether an effective communication protocol can emerge without explicit structural guidance. In the Learned Direct Communication (LDC) model, each agent's policy network outputs both a distribution over actions and a distribution over messages. The message is then broadcast and becomes part of the other agent's observation in the subsequent timestep.

4.1 Message Generation and Integration

The policy network of agent i is parameterized as $\pi_{\theta_i}(a_t^{(i)}, m_t^{(i)} | o_t^{(i)}, m_{t-1}^{(j)})$, where $j \neq i$. The message space is kept minimal to facilitate learning: a single binary value, $m_t^{(i)} \in \{0, 1\}$. The network outputs logits for the action (π_{act}) and message (π_{msg}), which are converted to probabilities via softmax and sigmoid functions, respectively. During training, actions and messages are sampled from these distributions. During evaluation, the most probable message is chosen deterministically. The entire system is trained end-to-end using the actor-critic loss (with $N_{comp} = 2$), meaning the communication protocol is shaped purely by its ability to contribute to the shared reward.

4.2 Experimental Validation

We first tested LDC in a fully observable 6×6 grid to verify if a meaningful protocol could be learned under ideal conditions.

4.2.1 Conditional Probability Analysis

After training, we analyzed the correlation between received messages and subsequent actions to understand the semantics of the emergent protocol. As shown in Tables 1 and 2, the action distributions are clearly conditioned on the received message. This reactive behavior demonstrates that each agent's learned world model implicitly contains a model of its teammate's policy; it anticipates the other's likely behavior based on the message received and adjusts its own strategy accordingly. For example, Agent 1 is significantly less likely to move 'Up' after receiving message 1 (4.53%) compared to message 0 (13.54%). This emergent behavioral convention, grounded in each agent modeling the other, is what allows for successful coordination.

Table 1: Agent 1's Action probabilities Conditioned on Agent 2's Message (Fully Observable)

	Action				
Message	Stay	Left	Right	Up	Down
0	31.47%	16.98%	19.21%	13.54%	18.80%
1	33.95%	22.86%	14.76%	4.53%	23.90%

Table 2: Agent 2's Action Probabilities Conditioned on Agent 1's Message (Fully Observable)

	Action				
Message	Stay	Left	Right	Up	Down
0	13.04%	21.61%	38.08%	22.83%	4.44%
1	13.81%	24.55%	29.18%	24.75%	7.71%

4.2.2 Ablation Study

To confirm a causal link, we performed an ablation study by replacing messages with a constant value (0) during evaluation. Table 3 shows that removing communication slightly degrades performance, increasing the average steps and decreasing the success rate, confirming the protocol's utility.

Table 3: LDC Performance With and Without Communication in a 6×6 Grid

Observability	Condition	Success Rate	Average Steps
Fully	With Messages	89.4%	4.39
	Messages Ablated (Set to 0)	88.6%	4.43
Partially (`vision_range=3`)	With Messages	31.89%	
	Messages Ablated (Set to 0)	30.26%	

The effect was more pronounced in a partially observable setting, where communication is more critical for sharing information about unseen goals. As shown in the bottom half of Table 3, communication provides a clear, albeit modest, benefit.

4.3 Limitations of Emergent Communication

While LDC demonstrates that communication can emerge, our supplementary experiments (see Appendix A) revealed its fragility. Increasing the message capacity (e.g., using more bits or a larger integer range) consistently prevented the policy from converging. We hypothesize that a larger message space dramatically expands the joint action-observation space, destabilizing the learning signal. This suggests that without strong architectural priors, learning a complex communication protocol from sparse rewards is exceptionally difficult, hinting at its poor scalability.

5 Approach 2: Intention Communication

To address the limitations of emergent protocols, we designed Intention Communication, an architecture with strong inductive biases for forward-looking coordination. The core idea is for agents to share explicit information about their future plans, allowing their teammates to anticipate their behavior and avoid conflicts [Liu et al., 2021]. This is operationalized through a two-stage process: an agent first generates an internal mental simulation of its future trajectory, and then compresses this plan into a message for its teammate.

5.1 Imagined Trajectory Generation Module (ITGM)

The ITGM serves as a compact, learned module to generate a plan. It combines a learned latent world model with the agent's own policy to generate a short-term forecast. Its objective is to simulate a future trajectory that would result from following its current policy. This policy-conditioned simulation provides the raw material for forming an intention. The process is as follows.

Initial State Encoding: The module first takes the agent's current local observation $o_t^{(i)}$ (a $(4 \times F)$-dimensional vector) and the received message from its teammate in the previous step, $m_{t-1}^{(j)}$. (This message $m_{t-1}^{(j)}$ is first embedded and flattened into a fixed-length vector). These are concatenated and passed through a linear encoder to produce an initial D_{hidden}-dimensional latent state vector, $z_t^{(i)}$: $z_t^{(i)} = \text{ReLU}(\mathbf{W}_{enc}[o_t^{(i)} \oplus \text{embed}(m_{t-1}^{(j)})] + \mathbf{b}_{enc})$.

where $\oplus$ denotes vector concatenation. This initial state $z_t^{(i)}$ represents a compressed summary of the agent's current knowledge and serves as the starting point for the plan.

Latent Rollout: The ITGM generates the plan by unrolling a trajectory for a fixed horizon H. This rollout actively uses the agent's own policy head (π_{act}) and a learned latent transition model (f_{trans}). This transition model is a network (e.g., an MLP) that predicts the next latent state given the current latent state and a *planned action*.

For each step k in the horizon ($k = 0, \ldots, H-1$), a planned action is sampled from the agent's policy and used by the latent transition model to predict the next state. This iterative process is implemented as follows, where $\text{embed}(\cdot)$ is an action embedding layer and π_{act} is the same policy

head used to select the agent's real action:

$$\tilde{a}_{t+k} \sim \pi_{act}(\cdot|z^{(i)}_{t+k})$$

$$z^{(i)}_{t+k+1} = \text{ReLU}(\mathbf{W}_{trans}[z^{(i)}_{t+k} \oplus \text{embed}(\tilde{a}_{t+k})] + \mathbf{b}_{trans})$$

This iterative process generates a sequence of predicted future latent states, which forms the plan (the policy-conditioned trajectory): $\tau^{(i)} = \langle z^{(i)}_{t+1}, z^{(i)}_{t+2}, \ldots, z^{(i)}_{t+H} \rangle$.

This feed-forward design was chosen for its computational efficiency, which is critical given our resource constraints. Both the encoder, the transition model, and the forward model are feed-forward networks with D_{hidden} hidden units and ReLU activation functions.

5.2 Message Generation Network (MGN)

The role of the MGN is to distill the high-dimensional plan (the latent trajectory $\tau^{(i)}$) into a compact, fixed-length message vector $m^{(i)}_t$ that effectively summarizes the agent's intention. A raw sequence of latent states is too unwieldy to use as a direct communication signal. Instead, we use a self-attention mechanism to identify and aggregate the most salient features of the planned trajectory.

We employ a multi-head self-attention mechanism [Vaswani et al., 2017] with N_{heads} head(s) and a D_{hidden}-dimensional embedding to compute a weighted summary of the trajectory sequence. The sequence $\tau^{(i)}$ serves as the query, key, and value inputs:$\mathbf{Q} = \tau^{(i)}\mathbf{W}_Q, \quad \mathbf{K} = \tau^{(i)}\mathbf{W}_K, \quad \mathbf{V} = \tau^{(i)}\mathbf{W}_V$. The attention mechanism produces an output sequence where each element is a weighted sum of the values, with weights determined by the query-key similarities. This process allows the network to learn which future steps in its imagined plan are most important for coordination (e.g., the final destination, a potential point of conflict).

This attention output is aggregated via mean pooling to produce a single vector v_{pooled}. This vector is then passed through two hidden layers, $h_1 = \text{ReLU}(\mathbf{W}_1 v_{pooled} + b_1)$ and $h_2 = \text{ReLU}(\mathbf{W}_2 h_1 + b_2)$.

To produce the message policy, h_2 is fed into N_{comp} separate linear heads. Each head $k \in [1, N_{comp}]$ outputs a vector of N_{sym} logits: $\text{logits}^{(k)} = \mathbf{W}^{(k)}_{head} h_2 + b^{(k)}_{head}$.

This defines N_{comp} independent categorical distributions, $\pi^{(k)}_{msg} = \text{Softmax}(\text{logits}^{(k)})$, one for each component of the message. This structured, multi-part discrete message is far more expressive than the single bit used in LDC.

5.3 Integration and End-to-End Training

The ITGM and MGN modules are seamlessly integrated into the agent's decision-making process. The full forward pass for a single agent i at timestep t explicitly separates the action and message generation to respect the constraint:

1. The agent receives its observation $o^{(i)}_t$ and the previous message from its teammate, $m^{(j)}_{t-1}$.
2. These inputs are encoded to produce the initial latent state $z^{(i)}_t$.
3. (Action Branch) The agent's policy head computes the real action distribution using this initial state: $\pi_{act}(a^{(i)}_t|z^{(i)}_t)$. An action $a^{(i)}_t$ is sampled.
4. (Plan/Message Branch) In parallel, the ITGM uses $z^{(i)}_t$ as its starting point to generate the plan $\tau^{(i)} = \langle z^{(i)}_{t+1}, \ldots, z^{(i)}_{t+H} \rangle$. This rollout process re-uses the same policy head π_{act} to sample hypothetical future actions $\tilde{a}_{t+k}$ at each step of the simulation.
5. The MGN processes the resulting plan $\tau^{(i)}$ to output the $N_{comp} \times N_{sym}$ logits for the N_{comp} message distributions $\pi^{(k)}_{msg}$.
6. A message $m^{(i)}_t = \langle m^{(1)}_t, \ldots, m^{(N_{comp})}_t \rangle$ is sampled by taking one symbol from each of the N_{comp} categorical distributions. This composite message vector is broadcast to its teammate.
7. The entire model is trained end-to-end using the A2C loss function. Gradients from the L_{actor} term (specifically the $A_t \sum_k \log \pi^{(k)}_{msg}(m^{(k)}_t)$ component) flow back to update the weights of the MGN, its hidden layers, the ITGM's transition model ($\mathbf{W}_{trans}$), and the

initial encoder ($\mathbf{W}_{enc}$). The gradients from both the action ($A_t \log \pi_{act}(a_t)$) and message terms also update the shared policy head π_{act}.

The entire model is trained end-to-end using the A2C loss function, allowing gradients to flow back through all modules.

6 Results: Scaling the Environment

We conducted our main experiments by comparing three models, Baseline (no communication), LDC, and Intention Communication, in partially observable environments of increasing scale. For these experiments, the `vision_range` was fixed to 2, making communication essential for locating and coordinating on goals, especially in larger grids.

6.1 Performance on Scaled Grids

The primary metric for evaluation is the success rate, defined as the percentage of episodes where agents successfully navigated to two distinct goals within the 200-step limit.

Table 4: Success Rates in Partially Observable Environments of Increasing Size (`vision_range=2`)

Environment Size	Model	Success Rate
10×10	Baseline (No Communication)	0.0%
	Learned Direct Communication	30.8%
	Intention Communication	**99.9%**
15×15	Baseline (No Communication)	0.0%
	Learned Direct Communication	12.2%
	Intention Communication	**96.5%**

The results are summarized in Table 4. The baseline model completely fails to solve the task, confirming that coordination is challenging without communication. LDC achieves a modest success rate in the 10×10 environment, but its performance collapses dramatically when the grid size is increased to 15×15. The sparse, unstructured binary signal is insufficient to handle the increased state space and the longer-horizon planning required.

In sharp contrast, Intention Communication demonstrates exceptional performance and robustness. It achieves a near-perfect success rate in the 10×10 grid and maintains an outstanding 96.5% success rate in the much larger 15×15 environment. The structured, forward-looking messages provide a rich and stable coordination signal that scales effectively with environmental complexity.

6.2 Sample Efficiency and Learning Dynamics

In both environmental settings, Intention Communication not only reaches a much higher asymptotic performance but also learns significantly faster. The architectural priors in the Intention Communication model provide a stronger and more targeted learning signal, allowing it to discover an effective coordination strategy with far greater sample efficiency. This is a critical advantage, especially under the computational constraints we operated under, and it is a key attribute for agents intended for large-scale or real-world deployment.

6.3 Analysis of Scaling Effects

The performance gap between LDC and Intention Communication widens as the environment scales. We attribute this to two key factors:

1. **Information Content:** The LDC's single-bit message acts as a severe information bottleneck. While it can learn simple conventions (e.g., "bit 0 means I'm going for the goal on the left"), this emergent semantics is not expressive enough to convey the nuanced plans needed in a large, partially observabåle world. Intention Communication, by contrast, transmits a compressed representation of a multi-step plan, a much richer information source.

2. **Structural Priors:** The Intention Communication architecture embeds the strong prior that planning ahead is useful. By forcing the agent to generate and communicate imagined trajectories, we guide the learning process toward discovering forward-looking, model-based coordination strategies. LDC has no such guidance and must discover this principle from scratch via trial and error, a task that becomes intractable as the state space grows.

7 Discussion and Future Work

Our findings have significant implications for the design of agents intended to operate in large, complex, and interactive environments.

On Scaling Environments for Agents: This work serves as a case study demonstrating that as we scale environmental complexity, we must also scale the sophistication of agent architectures. Naive, end-to-end learning approaches may not be sufficient. The success of Intention Communication suggests that building agents with explicit capacities for planning, prediction, and structured communication is a promising direction for achieving robust performance in scaled-up settings. The simple, configurable grid world we use can serve as a valuable benchmark for probing the scalability of different MARL coordination mechanisms along controlled axes of difficulty.

Limitations: Our study is conducted in a relatively simple, deterministic grid world with only two agents. Generalizing these findings to environments with stochastic dynamics, continuous state-action spaces, or a much larger number of agents presents a significant research challenge. With many agents, the broadcast communication channel used here could become a bottleneck, suggesting a need for more targeted or hierarchical communication schemes. Furthermore, our constrained computational budget prevented exploration of more powerful, transformer-based architectures for LDC, which might have yielded a more robust emergent protocol.

Future Work: An exciting avenue for future research is the development of hybrid communication models. Such models could combine the stability of engineered protocols with the flexibility of learned communication, perhaps by allowing agents to learn *what*, *how*, and *when* to communicate within a structured framework. Applying the Intention Communication architecture to more complex benchmarks, such as cooperative robotics or software navigation tasks, would be a critical next step in validating its effectiveness. We have begun using RoboCup as an environment[Kitano et al., 1997]. Finally, exploring co-evolutionary approaches where both the agent policies and the environmental complexity scale together could lead to a curriculum learning paradigm that fosters the development of more generally intelligent agents.

8 Conclusion

In this paper, we compared the scalability of emergent and engineered communication strategies for a cooperative multi-agent task. We demonstrated that while an end-to-end learned protocol (LDC) can facilitate coordination in simple environments, it fails to scale as the environment becomes larger and more challenging. In contrast, our proposed Intention Communication architecture, which equips agents with an internal planning module and a mechanism for sharing compressed future trajectories, proved to be highly robust, sample-efficient, and scalable.

This result highlights a critical principle for the development of autonomous agents: for coordination to be effective in complex, scaled environments, it may be necessary to move beyond purely emergent systems and endow agents with structured architectural priors that facilitate explicit and forward-looking communication. As we continue to push the boundaries of agent capabilities, building in such foundational capacities for planning and intention sharing will be a key enabler of success.

Acknowledgments and Disclosure of Funding

We thank Denon Chong Cheng Zong and Jeff Lee for their contributions to research on MARL communication. We thank Dr. Matthew Berland for their mentorship and feedback.

Contributions

Brennen Hill: Project concept, project leadership, environment development, baseline model development, Learned Direct Communication development, Intention Communication development, scaling, optimization, MARL communication research, agent training.

Mant Koh En Wei: Intention Communication development, baseline model development, optimization, scaling, agent training.

Thangavel Jishnuanandh: Baseline model development, MARL communication research, optimization, scaling, agent training.

References

Daniel S. Bernstein, Shlomo Zilberstein, and Neil Immerman. The complexity of decentralized control of markov decision processes. *Journal of Artificial Intelligence Research*, 17:379–428, 2002. doi: 10.1613/jair.999.

Micah Carroll, Rohin Shah, Mark K. Ho, Thomas L. Griffiths, Sanjit A. Seshia, Pieter Abbeel, and Anca D. Dragan. On the Utility of Learning About Humans for Human-AI Coordination. In *Advances in Neural Information Processing Systems*, volume 32, pages 568–579, 2019.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pages 2048–2056. PMLR, 2020.

Jakob N. Foerster, Yannis M. Assael, Nando de Freitas, and Shimon Whiteson. Learning to communicate with deep multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 29:2137–2145, 2016. doi: 10.5555/3157096.3157290. URL https://papers.nips.cc/paper/6570-learning-to-communicate-with-deep-multi-agent-reinforcement-learning.pdf.

Google. Colaboratory. https://colab.research.google.com/, 2025. Accessed: 2025-08-18.

Su Kefan, Zhou Siyan, Jiang Jiechuan, Gan Chuang, Wang Xiangjun, and Lu Zongqing. Multi-agent alternate q-learning. *IFAAMAS*, 2024.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2014.

Hiroaki Kitano, Minoru Asada, Yasuo Kuniyoshi, Itsuki Noda, Eiichi Osawa, and Hitoshi Matsubara. Robocup: A challenge problem for ai. *AI magazine*, 18(1):73–85, 1997.

Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Natural Language Does Not Emerge 'Naturally' in Existing Dialog Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2645–2654. Association for Computational Linguistics, 2017.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajkac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, and Sylvain Gelly. Google Research Football: A Novel Reinforcement Learning Environment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 4501–4508, 2020.

Xuesi Li, Shuai Xue, Ziming He, and Haobin Shi. Tmac: a transformer-based partially observable multi-agent communication method. *PeerJ Computer Science*, 2025. doi: 10.7717/peerj-cs.2758. URL https://peerj.com/articles/cs-2758.pdf.

Zeyang Liu, Lipeng Wan, Xue Sui, Kewu Sun, and Xuguang Lan. Multi-agent intention sharing via leader-follower forest. *arXiv preprint arXiv:2112.01078*, 2021. doi: 10.48550/arXiv.2112.01078. URL https://arxiv.org/abs/2112.01078.

Zeyang Liu, Lipeng Wan, Xinrui Yang, Zhuoran Chen, Xingyu Chen, and Xuguang Lan. Imagine, initialize, and explore: An effective exploration method in multi-agent reinforcement learning. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence (AAAI-24)*. AAAI Press, 2024.

Ryan Lowe, Yi I. Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In *Advances in Neural Information Processing Systems*, volume 30, pages 6379–6390, 2017.

Yang Ming, Zhao Kaiyan, Wang Yiming, Dong Renzhi, Du Yali, Liu Furui, Zhou Mingliang, and Hou Leong. Team-wise effective communication in multi-agent reinforcement learning. *Springer US*, 2024.

Igor Mordatch and Pieter Abbeel. Emergence of Grounded Compositional Language in Multi-Agent Populations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, pages 871–879, 2018.

Abolfazl Oroojlooyjadid, Dongsheng He, and Mihaela van der Schaar. A Review of Cooperative Multi-Agent Deep Reinforcement Learning. *IEEE Transactions on Neural Networks and Learning Systems*, 34(10):6352–6371, 2023. doi: 10.1109/TNNLS.2022.3165387.

Xihe Qiu, Haoyu Wang, Xiaoyu Tan, Chao Qu, Yujie Xiong, Yuan Cheng, Yinghui Xu, Wei Chu, and Yuan Qi. Towards collaborative intelligence: Propagating intentions and reasoning for multi-agent coordination with large language models. *arXiv preprint arXiv:2407.12532*, 2024. URL https://arxiv.org/abs/2407.12532.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Phelps, and Shimon Whiteson. The StarCraft Multi-Agent Challenge, 2019.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob Fergus. Learning multiagent communication with back-propagation. *Advances in Neural Information Processing Systems*, 2016. URL https://papers.nips.cc/paper/6398-learning-multiagent-communication-with-backpropagation.pdf.

Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 2nd edition, 2018. URL http://incompleteideas.net/book/the-book-2nd.html.

J Terry, Benjamin Black, Nathaniel Grammel, Mario Jayakumar, Ananth Hari, Ryan Sullivan, Luis S Santos, Clemens Dieffendahl, Caroline Horsch, Rodrigo Perez-Vicente, et al. Pettingzoo: Gym for multi-agent reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 15032–15043, 2021.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention Is All You Need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008, 2017.

A Technical Appendices and Supplementary Material

A.1 Exploration of LDC Limitations

We conducted several additional experiments to better understand the limitations and fragility of the Learned Direct Communication (LDC) approach, which justify our conclusion that it does not scale well with environmental complexity.

A.1.1 Varying Message Capacity

A natural hypothesis for LDC's failure in larger environments is its limited one-bit message space. We experimented with increasing the message capacity by widening the range of values (e.g., integers from 0-4 or 0-99, treated as a categorical distribution) and by allowing agents to send multiple message values per timestep.

Table 5: Convergence Results for LDC with Varying Message Capacity

Message Range	Message Count Per Step	Converged to >1% Success Rate
0-1	1	**Yes**
0-1	2	No
0-1	4	No
0-1	10	No
0-2	1	Yes
0-2	2	No
0-2	4	No
0-2	10	No
0-4	1	No
0-4	2	No
0-4	4	No
0-9	1	No
0-9	2	No
0-9	4	No
0-99	1	No
0-99	2	No
0-99	4	No

As shown in Table 5, the only policy that converged to a non-trivial success rate was the one with a single binary message. We hypothesize that a larger message space significantly increases the dimensionality of the agent's observation and action spaces. The policy must not only learn to interpret incoming messages but also to generate meaningful outgoing ones. With a sparse, global reward signal, this credit assignment problem becomes intractable, introducing significant noise into the learning process and hindering convergence.

A.1.2 Rewarding Meaningful Messages

To guide the learning of the communication protocol, we attempted to introduce an explicit reward for sending meaningful messages. This was based on the concept of influence, where the influence of a message is measured by how much it improves the recipient's value function estimate. However, this form of intrinsic reward shaping proved exceptionally difficult to tune. High reward coefficients for communication would often override the primary task reward, leading to policies that were effective at signaling but poor at navigation. Conversely, low coefficients had no discernible effect on performance, failing to accelerate the learning of a useful protocol.

A.1.3 Forcing Communication Necessity

We designed an environment variant where communication was structurally indispensable. In this setting, each agent could only see the goal intended for its teammate, making information exchange the only possible path to a solution. Even under these conditions, the LDC model failed to converge. The agents could not bootstrap a meaningful protocol from random initial messages and actions, indicating a fundamental difficulty in learning semantics from scratch in challenging scenarios.

A.1.4 Varying Network Architectures

We explored various network architectures for the LDC agents. Deviations from our standard architecture (two hidden layers with 64 ReLU-activated nodes each), such as making the networks deeper or wider, generally resulted in longer training times or a complete failure to converge. Smaller networks lacked the capacity to solve the task, while larger networks seemed to exacerbate the training instability.

A.2 Conclusions from Additional Experiments

These supplementary experiments collectively underscore the difficulty of fostering stable and scalable emergent communication under significant computational constraints and without strong architectural biases. The success of the Intention Communication approach highlights the value of engineered strategies that structure the communication problem around semantically meaningful concepts like plans and intentions, leading to more robust, scalable, and sample-efficient learning.

A.3 Key Training and Architectural Hyperparameters

We share the hyperparameters used in this experiment for greater reproducability in table 6.

Table 6: Key Training and Architectural Hyperparameters

Parameter	Value
Training Hyperparameters	
Initial LR (lr_0)	1×10^{-3}
Final LR (lr_f)	1×10^{-4}
Discount Factor (γ)	0.99
Optimizer	Adam [Kingma and Ba, 2014]
Critic Loss Coefficient (α_c)	0.5
Max Episode Length (T_{max})	200
Architectural Hyperparameters (all)	
Frame Stack (F)	4
Hidden Dimension (D_{hidden})	64
Hidden Layers (N_{hidden})	2
Message Loss Coefficient (β)	0.1
Architectural Hyperparameters (Intention Communication)	
ITGM Rollout Horizon (H)	3
Attention Heads (N_{heads})	1
Message Components (N_{comp})	8
Symbol Vocabulary Size (N_{sym})	8

A.4 LDC architecture

The architecture for Learned Direct Communication is depicted in figure 2.

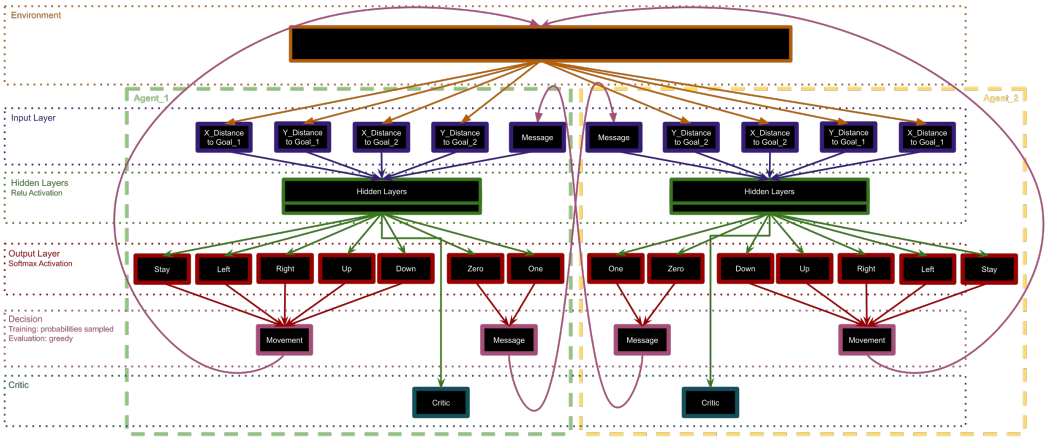

Figure 2: The LDC architecture. Each agent's policy network takes its local observation and the incoming message from the previous timestep to produce an action and an outgoing message for the next timestep. The critic network estimates the state value.