Language-conditioned world model improves policy generalization by reading environmental descriptions

Anh (Joe) Nguyen
Oregon State University
nguyejoe@oregonstate.edu

Stefan Lee
Oregon State University
leestef@oregonstate.edu

Abstract

To interact effectively with humans in the real world, it is important for agents to understand language that describes the dynamics of the environment—that is, *how the environment behaves*—rather than just task instructions specifying *what to do*. For example, a cargo-handling robot might receive a statement like "the floor is slippery so pushing any object on the floor will make it slide faster than usual". Understanding this dynamics-descriptive language is important for human-agent interaction and agent behavior. Recent work [23, 43, 6] address this problem using a model-based approach: language is incorporated into a world model, which is then used to learn a behavior policy. However, these existing methods either do not demonstrate policy generalization to unseen language or rely on limiting assumptions. For instance, assuming that the latency induced by inference-time planning is tolerable for the target task or that expert demonstrations are available. Expanding on this line of research, we focus on improving policy generalization from a language-conditioned world model while dropping these assumptions. We propose a model-based *reinforcement learning* approach, where a language-conditioned world model is trained through interaction with the environment, and a policy is learned from this model—without planning or expert demonstrations. Our method proposes Language-aware Encoder for Dreamer World Model (LED-WM) built on top of DreamerV3 [14]. LED-WM features an observation encoder that uses an attention mechanism to explicitly ground language descriptions to entities in the observation. We show that policies trained with LED-WM generalize more effectively to unseen games described by *novel dynamics and language* compared to other baselines in several settings in two environments: MESSENGER and MESSENGER-WM. To highlight how the policy can leverage the trained world model before real-world deployment, we demonstrate the policy can be improved through fine-tuning on synthetic test trajectories generated by the world model.

1 Introduction

We envision a future where humans can seamlessly command AI agents through natural language to automate repetitive tasks in the real world. Traditionally, language has been used to specify task instructions, such as telling a navigation robot to "go to the door" [2, 20, 1]. However, language can also offer valuable information about environments. Such environmental description not only makes human interaction more natural, but also provides important contextual information about how the environment changes over time. It informs the agent about *how the environment behaves*—its dynamics, the current state of the world, and how various entities interact with each other and with the agent—not just *what to do*.

We illustrate dynamics-descriptive language by using a simple 2D grid-based game in Figure 1, instantiated by MESSENGER S2 [15]. This is the setting of our testbed environments and will be detailed in Section 3.1 Each game instance consists of several entities, an agent positioned in a

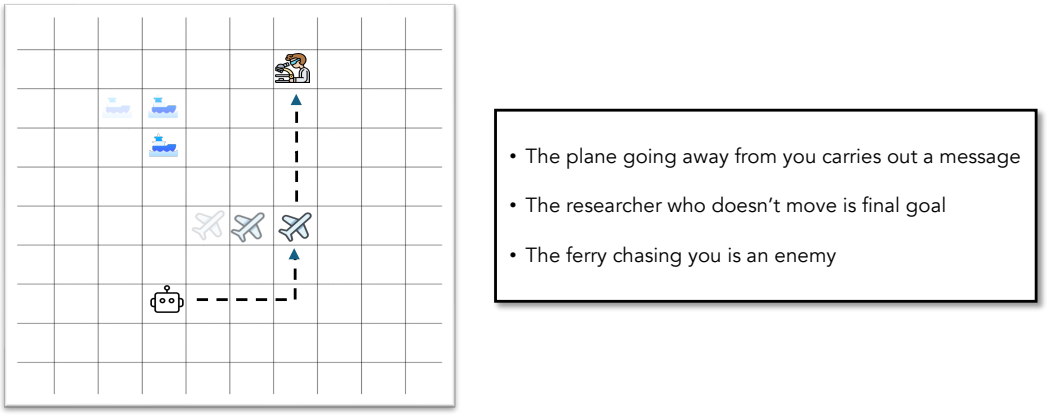


Figure 1: An example of dynamics-descriptive language in a game play. The observation includes a 10×10 grid-world with three entities represented by their associated symbols: (`ferry` -), (`plane` -), (`researcher` -) and one agent (depicted by). The observation also has a manual on the right, which describes the dynamics of the game. The agent can navigate the grid using five actions: `left`, `right`, `up`, `down`, and `stay`. The agent can only interact with entities when it is in the same grid cell as the entity. The agent's task is to identify roles of all entities from the manual, go to the `messenger`, then go to the `goal`, while avoiding the `enemy`. Shaded icons indicate one possible scenario of entity movement over time. By observing entity movement patterns and grounding language to entities based on their behaviors, the agent can infer the roles assigned to each entity: `(ferry-enemy)`, `(plane-messenger)`, and `(researcher-goal)`. The agent can then execute an appropriate plan to complete the task. The dashed line in the grid shows such a possible plan.

grid-world observation, and a language manual. Each entity has a role among `messenger`, `goal`, and `enemy`. The agent acts as a courier, tasked with picking up a message from the `messenger` and delivering it to the `goal` while avoiding the `enemy`. The manual provides descriptions of the entity attributes, helping the agent understand the environment's dynamics: what the roles of entities are and how the environment changes as the agent interacts with them. To succeed, the agent must interpret the language manual, identify the entities, and infer their respective roles based on observed behaviors.

Language is valuable because it allows for the description of novel games by recombining known concepts. For instance, consider Figure 1 as the training reference game and the following example manual: `The ship going away from you is the goal you need to go to. The stationary plane is an enemy. The scientist won't move and has an important message.` The example manual describes an *unseen dynamics* game with known concepts derived from the reference game. To succeed in this environment—where the dynamics have changed but the rules remains the same—the agent must adopt a different behavior than in the reference game. This manual also produces *novel surface-level language* through synonyms (e.g. "researcher" vs "scientist") and paraphrases ("won't move" vs "stationary"),

We want to study language grounding and how it affects agent generalizability. Therefore, we abstract away our observation to a discrete grid-world, thus simplifying perception complexity, similar to existing work [15, 29, 23, 43, 6]. Our goal is to develop an agent capable of understanding dynamics-descriptive language by grounding it to discrete entities. More importantly, we aim for the agent to generalize to *unseen games described by unseen dynamics and/or novel language*, allowing it to adapt agent behavior to new environmental changes.

In the current literature, there are two main approaches to building such an agent: model-free and model-based approach. Model-free methods [15, 29] directly map language to a policy. Language grounding is thus based entirely on policy learning signals, without modeling the environment dynamics. This might be challenging for agent to learn complex mapping from dynamics-descriptive language to action. Meanwhile, model-based methods like EMMA-LWM [43], Reader [6], and Dynalang [23] build a world model [12] simulating trajectories, which are then used to train a policy.

Dynamics-descriptive language is incorporated into the world model, enabling it to use language to predict environmental changes.

However, these existing works have some limitations. EMMA-LWM requires expert demonstrations—a constraint that may not be always feasible for real-life tasks. Reader assumes inference-time latency is tolerable for the target tasks. This is because Reader uses a Monte Carlo Tree Search (MCTS) to look ahead and generate a full plan. This approach may not be practical for applications that require quick policy responses. Last, we show that the policy learned from Dynalang fails to generalize over unseen games in Section 5.1. To address these limitations, we adopt a model-based reinforcement learning (MBRL) approach that builds a language-grounded world model from interaction with the environment, and then use this world model to train a policy. In contrast to previous methods, our approach does not require expert demonstrations, avoids expensive inference-time planning, and can generalize to unseen games.

We propose Language-aware Encoder for Dreamer World Model (LED-WM), building on a MBRL framework: DreamerV3 [14]. LED-WM introduces a new encoder for DreamerV3 that explicitly grounds entities to their language descriptions, using a simple yet effective attention mechanism. In this paper, we make the following contributions:

- We show that a language-conditioned MBRL without an explicit language grounding to entities, instantiated by Dynalang [23], fails to generalize over unseen games (see Section 5.1).
- By using an attention mechanism in LED-WM to do language grounding, we show that a policy trained from LED-WM can generalize over unseen games better than model-free and model-based baselines in several settings MESSENGER and MESSENGER-WM (see Section 4).
- We demonstrate that given a trained LED-WM, we can improve a trained policy by fine-tuning it in synthetic test trajectories generated by the world model (see Section 5.2).

2 Background

Problem formulation. We define our problem as a language-conditioned Markov Decision Process, represented by a tuple with common notations: $(\mathcal{S}, \mathcal{A}, r, T, \gamma, H)$. $\mathcal{S}$ represents the state space where each state has a 10×10 grid-world observation containing entity symbols and an agent (e.g.  ,  ,  ,  in Figure 1). Each state also has a language manual L, describing *environment dynamics*: transition function $T(s'|s, a)$ and reward function $r(s, a)$. L consists of N sentences associated with N entities, where each sentence describes the dynamics of each entity. An example of a state is shown in Figure 1. Action space $\mathcal{A} = \{\texttt{up}, \texttt{down}, \texttt{right}, \texttt{left}, \texttt{stay}\}$ is discrete. The agent must take a sequence of actions $a_t \in \mathcal{A}$ over a horizon H, where time step $t \in [1..H]$, resulting in a state-action trajectory $(s_1, a_1, \ldots, s_H, a_H)$. Our goal is to find a policy $\pi : \mathcal{S} \times L \to \mathcal{A}$ that maximizes the expected sum of discounted rewards: $\mathbb{E}_{\pi,L}\left[\sum_{t=1}^{H} \gamma^{t-1} r(s_t, a_t)\right]$.

World model DreamerV3. We base our world model on DreamerV3 [14], which uses Recurrent State-Space Model (RSSM) [13] to build a recurrent world model. DreamerV3 receives a sequence of observations and predicts latent representations of future observations given actions. Specifically, at a time step t, DreamerV3 receives an observation x_t, an action a_t, and history information h_t. These inputs are compressed into a latent representation z_t and fed to RSSM with the action a_t to predict the next latent representation z_{t+1}. The world model has the following components:

$$\text{RSSM} \begin{cases} \text{Sequence model:} & h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\ \text{Encoder:} & z_t \sim q_\phi(z_t \mid h_t, x_t) \\ \text{Dynamics predictor:} & \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\ \text{Reward predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t) \\ \text{Continue predictor:} & \hat{c}_t \sim p_\phi(\hat{c}_t \mid h_t, z_t) \\ \text{Decoder:} & \hat{x}_t \sim p_\phi(\hat{x}_t \mid h_t, z_t) \end{cases} \tag{1}$$

In this work, we propose to change the encoder of DreamerV3 to better leverage language grounding to learn a more robust world model.

Table 1: Summary of generalization capabilities over unseen games in MESSENGER and MESSENGER-WM across stages. Examples with visualizations are provided in Appendix C.2.

	MESSENGER				MESSENGER-WM		
	S1	S2	S2-dev	S3	New Combo	New Attr	New All
Novel combinations of known entities	✓	✓	✓	✓	✓	✗	✓
Novel language (synonyms and paraphrase)	✓	✓	✓	✓	✓	✓	✓
Novel entity-role assignments	✓	✓	✓	✓	✗	✓	✓
Novel entity-movement-role assignments	✗	✓	✓	✓	✗	✓	✓
Novel game dynamics of known movement behaviors	✗	✓	✗	✗	✗	✓	✓
Novel game dynamics from one training dynamic	✗	✓	✗	✗	✗	✗	✗

3 Environment setup

We adopt MESSENGER [15] and MESSENGER-WM [43] as our test bed environments. Both environments have the same setup as the example game in Figure 1. To succeed in the game, the agent must understand the language manuals L and use reward and transitional signals to ground the roles, entity names and movement types to the entity symbols in the observation.

3.1 MESSENGER

Overview. As shown in Figure 1, MESSENGER [15] is a 10×10 grid-world environment. We refer the readers to Figure 1 for game rules and setup, and Appendix C.1.1 for environment dynamics and action. Each game includes a language manual and an observation containing entities and a single agent. For more details about language grounding to entities, we refer the readers to Appendix C.1.2.

Evaluation settings. MESSENGER offers four stages (stage S1, S2, S2-dev, S3) with different levels of generalization for test games. Each stage has its own training set, test set, and development set, all of which are described in detail in Appendix C.1.

3.2 MESSENGER-WM

Overview. While providing multiple stages to evaluate policy generalization over out-of-distribution dynamics, MESSENGER does not include a setting for compositional generalization dynamics. To bridge this gap, MESSENGER-WM [43], derived from MESSENGER S2, enables evaluation at compositional generalization for world model and policy. Together, these two environments offer a comprehensive framework for assessing generalization, under varying levels of unseen games.

Evaluation settings. MESSENGER-WM has three different evaluation settings with different levels of generalization: NewCombo, NewAttr, and NewAll. All settings share the same training set. More details about the evaluation settings are provided in Appendix C.3.2 and the original paper [43].

3.3 Evaluating generalization to unseen games

Together, MESSENGER and MESSENGER-WM offer different levels of generalization in test games. We summarize these in Table 1 and below:

- *Novel language*: the test manual uses synonyms and paraphrases to create novel language through surface structure.
- *Novel combinations of known entities*: the test game involve entities that appear in training set but never appear together in one training game.
- *Novel entity-role assignments*: at least one entity in the test game has a different role from its roles in training games.

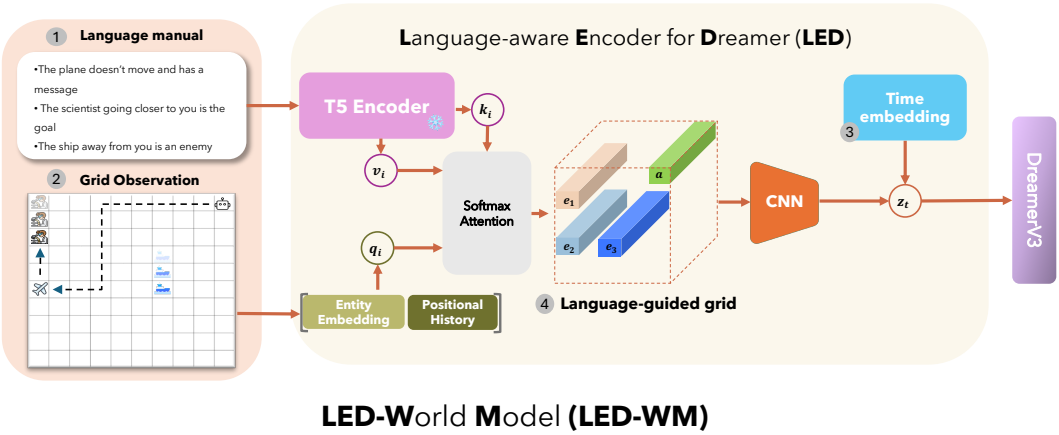


Figure 2: Overview of our proposed world model LED-WM. The world model input consists of: (1) a language manual L, (2) a grid-world observation representing entity and agent symbols, and (3) the current time step t. Entity, agent symbols, and time step are encoded using learned embeddings, while L is encoded via a frozen T5 encoder. To represent each entity, we employ a multi-layer perceptron (MLP) that processes the entity embedding and its temporal information, capturing its movement pattern relative to the agent, to produce a query vector. We apply an attention network between the query vectors and the sentence embeddings to align each entity with its corresponding sentence. The resulting vectors are then put into their respective entity positions. This produces (4) a language-grounded grid G_l, which is then processed by a CNN. The extracted feature vector is flattened and concatenated with the time embedding to form final observation representation x_t.

- *Novel entity-role-movement assignment*: at least one entity in the test game has a novel combination of entity-role-movement assignment.
- *Novel combinations of known movement behaviors (novel game dynamics)*: the test game has a novel movement combinations of entities, e.g. `(chaser-chaser-chaser)` for three entities.
- *Novel game dynamics from one training dynamic*: Game dynamic is defined by the combination of entity movements. In the training set, there is only one such combination `(chasing-fleeing-stationary)` across all training games. The test game meanwhile has a novel dynamic, e.g. `(chaser-chaser-chaser)`. This is also the difference between MESSENGER-WM and MESSENGER, which can be found more detailed in Appendix C.3.1

See Appendix C.2 for visualizations of these settings.

4 Method: Language-aware Encoder for Dreamer World Model (LED-WM)

To generalize policy across unseen games, we aim to develop a world model capable of doing language grounding to entities in a game. Inspired by EMMA [15], we propose Language-aware Encoder for Dreamer (LED), which uses cross-modal attention to align game entities with sentences. The resulting vectors are then placed back into their original entity locations, producing a language-aware grid observation. This grid is passed through a CNN encoder to extract observation features, which are used by the other components of DreamerV3. We call this overall model LED-World Model (LED-WM). We provide an overview of the encoder LED and the world model LED-WM in Figure 2 and describe each component in the following sections.

4.1 Observational inputs

The input to the world model consists of a natural language manual L, a grid observation o_t of size 10×10, containing symbolic entities, and the current time step t. The manual (1) comprises N sentences, each describing the dynamic of one of the N entities in the observation. Following Lin et al. [23], N sentences in L are encoded using a T5 encoder [33], resulting in N frozen sentence

embeddings, denoted by $s_1, s_2, ..., s_N$. In the grid (2), the N entities and the agent, represented by entity symbols, are encoded using a learned entity embedding vectors initialized with random weights. [1] This results in N symbol embeddings $sb_1, sb_2, ..., sb_N \in \mathbb{R}^{d_{sb}}$ and a single agent embedding $a \in \mathbb{R}^{d_{sb}}$. The current time step t (3) is encoded as $time_t$ using a learned time embedding, also initialized with random weights.

To build position history of each entity i, we capture temporal dynamics by constructing an array D_i temporally, with length corresponding to the maximum possible steps in the environment and initial values of -1. At time step t, let the 2D coordinate of the entity i be p_i^t and that of the agent be p_a^t. To determine the relative direction of the entity's movement with respect to the agent, we compute the dot product:

$$D_i^t = \frac{p_i^t - p_a^t}{\|p_i^t - p_a^t\|} \cdot \frac{p_i^t - p_i^{t-1}}{||p_i^t - p_i^{t-1}||}, \forall i \in [1..N], \forall t, \tag{2}$$

where the first term is a normalized vector from the agent to the entity i, and the second term is a normalized velocity vector of the entity i. This dot product quantifies the alignment between the entity's direction of motion and its position relative to the agent at each time step t.

4.2 LED: Building a language-aware encoder

We construct a language-grounded grid representation that aligns the language manual L, which consists of N sentence embeddings, with the observation o_t, which includes N entity embeddings and one agent embedding. To align the sentence embeddings with the entity embeddings, we use an attention network. The values are obtained through a linear transformation of the sentence embeddings s_i. Meanwhile, the queries are obtained through a multi-layer perception (MLP) applied to the entity embeddings sb_i and temporal array D_e. Likewise, the keys are obtained through an MLP applied to the sentence embeddings s_i :

$$q_i = \text{MLP}([sb_i, D_e]), \qquad k_i = \text{MLP}(s_i), \qquad v_i = W_v s_i, \tag{3}$$

$$q_i \in \mathbb{R}^d, \qquad k_i \in \mathbb{R}^d, \qquad v_i \in \mathbb{R}^{d_{\text{val}}}, \tag{4}$$

where d and d_{val} denote the dimensions of the query/key and value vectors, respectively. We then apply scaled dot-product attention [38]. Given $K \in R^{N \times d}$ as the key matrix where the row i of K is k_i^T, attention scores $\gamma_i \in R^N$ and resulting vector $e_i \in R^{d_a}$ for each entity i are calculated as:

$$\gamma_i = softmax\left(\frac{q_i \cdot K}{\sqrt{d}}\right), \quad e_i = \sum_{j=1}^{N} \gamma_{ij} v_j, \tag{5}$$

This attention aligns entity symbols in the observation with sentences in the manual based on attribute language descriptions such as movement (e.g., `chaser, moving away, stationary`) and entity name (e.g., `dog, wizard`). The resulting e_i from the attention is able to represent an associated role for entity i such as `enemy, messenger, goal`, which is vital information for world model and policy learning.

To retain the spatial information of entities, we place the resulting vectors e_i back into the original positions of their corresponding entities in the grid observation. This produces (4) a language-aware grid observation G_l of size $h \times w \times d_{val}$. We then use a CNN encoder to extract a feature map, which is subsequently flattened and concatenated with the time embedding time_t. The combined representation is processed through an MLP to obtain the final feature representation x_t for the observation o_t at time step t:

$$x_t = \text{MLP}(\text{Flatten}([\text{CNN}(G_l)), \text{time}_t]) \tag{6}$$

Denoting ϕ as the parameters of LED-WM, we can find stochastic variable z_t as the function of x_t:

$$z_t \sim q_\phi(z_t | h_t, x_t), \tag{7}$$

which now replaces the encoder in DreamerV3, as shown in Equation (1).

[1] This ensures the agent does not have prior knowledge about entity identities, requiring it to infer entities based on the language.

4.3 LED-WM: Combining LED with Dreamerv3

We replace DreamerV3's encoder with LED, resulting in our world model LED-WM. We adopt world model and policy learning from DreamerV3. However, we make the following changes to the original architecture to improve policy generalization and sample efficiency: we omit the reconstruction decoder (Decoder in Equation (1)) and adopt multi-step prediction for reward and continue prediction [16, 30]. For more details, we refer the readers to Appendix D for world model loss and Appendix E for training procedure.

5 Experiments

We want to answer the following questions: 1) Can a policy trained on our world model LED-WM generalize to unseen games? (see Section 5.1), and 2) Can the world model LED-WM generalize to unseen games? (see Section 5.2) To answer these, we use two environments: MESSENGER and MESSENGER-WM, which are detailed in Section 3. We detail the training settings in Appendix A.

5.1 Policy generalization trained from LED-WM

5.1.1 Policy baselines

As baselines, we adopt the following model-free (EMMA,CRL) and model-based (Dynalang, EMMA-LWM) methods:

- *EMMA* [15] uses attention between entities and sentences to generate language-conditioned observation to the policy. The policy is trained via curriculum learning where the agent is initialized with parameters learned from previous easier game settings. We report EMMA with curriculum learning from the original paper and EMMA without curriculum learning from [43].
- *CRL* [29] develops a specialized constraint for MESSENGER to overcome spurious correlations between entity identities and their roles in the training data. It has the state-of-the-art win rate performance in test environments of MESSENGER.
- *Dynalang* [23] use soft actor-critic for policy learning. Because the paper does not report policy generalization performance in MESSENGER, we first reproduce Dynalang using published code and train to convergence according to published hyperparameters and training steps. We then report its policy performance on test environment of MESSENGER in Table 2.
- *EMMA-LWM* [43] built a language-conditioned world model. A policy is trained with simulated trajectories from this world model through online imitation learning and filtered behavior cloning. Both methods require expert demonstrations. [2]

5.1.2 Evaluation metrics

- *Win Rates for MESSENGER*: To make our comparison consistent with reported results from EMMA [15] and CRL [29], we adopt win rate as the metric in MESSENGER. Win rate is calculated as the average number of games won by the agent over 1000 episodes.
- *Average Sum of Scores for MESSENGER-WM*: Likewise to be consistent with EMMA-LWM [43] studying MESSENGER-WM, we adopt average sum of scores as the metric. For each game configuration, we run the policy for 60 trials [3] and compute the average sum of scores. This process is repeated for 1000 games, and we report the average sum across all games.

5.1.3 Results

We report the win rate performance of our method and other baselines for MESSENGER in Table 2 and the average sum of scores for MESSENGER-WM in Table 3.

[2] Online imitation learning is where the expert supervises the optimal action to take in simulated states (from the world model). Meanwhile, in filtered behavior cloning, the expert uses only states from its own expert plan. The agent then only chooses plans that achieve the highest returns according to the world models to imitate.

[3] We find that 60 trials are enough to find a stable average sum of scores to evaluate a policy given a particular game configuration.

Table 2: Policy generalization in MESSENGER in terms of win rate. Note that other methods (Dynalang, CRL and LED-WM) do not use curriculum training. Results of Dynalang and LED-WM (*) are rounded to second decimal place, while results for CRL and EMMA are taken from their original papers. Results are recorded across five training seeds.

Method	MESSENGER			
	S1	**S2**	**S2-dev**	**S3**
Dynalang*	0.03 ± 0.02	0.04 ± 0.05	–	0.03 ± 0.05
CRL	88 ± 2.5	**76 ± 5**	–	32 ± 1.9
EMMA (w/o curriculum)	85 ± 1.4	45 ± 12	–	10 ± 0.8
EMMA(w/ curriculum)	88 ± 2.3	95 ± 0.4	–	22 ± 3.8
LED-WM (Ours)*	**100 ± 0**	51.6 ± 2.7	**96.6 ± 1.0**	**34.97 ± 1.73**

Table 3: Policy generalization in MESSENGER-WM in terms of average sum of scores. EMMA-LWM results are taken from its original paper [43]. Results are recorded across five training seeds.

Method	MESSENGER-WM		
	NewCombo	**NewAttr**	**NewAll**
EMMA-LWM			
Online IL	1.01 ± 0.12	0.96 ± 0.17	0.62 ± 0.21
Filtered BC (near-optimal)	1.18 ± 0.10	0.75 ± 0.20	0.44 ± 0.18
Filtered BC (suboptimal)	0.98 ± 0.13	0.29 ± 0.25	0.13 ± 0.19
LED-WM (Ours)	**1.31 ± 0.05**	**1.15 ± 0.08**	**1.16 ± 0.02**

In MESSENGER-WM, LED-WM outperforms EMMA-LWM in all settings without using any expert demonstrations. In MESSENGER, Dynalang fails to generalize to unseen games. We hypothesize that this is because Dynalang lacks an explicit mechanism to ground language to each entity. Meanwhile, LED-WM is better than other baselines in S1 and comparable to CRL in S3.

However, LED-WM underperforms CRL in S2, where the agent is trained on only one movement combination `chasing-fleeing-stationary` but is evaluated over different unseen movement combinations (unseen dynamics - see Table 1). In contrast, LED-WM performs well on S2-dev, where its setting is similar to S2, but its test dynamics are the same as the training games. We hypothesize that this occurs because CRL incorporates an explicit mechanism to mitigate the data bias in S2 that there is only one movement combination in the training data and spurious correlations between entity identities and their roles. For instance, the assumption "a dog is always a goal". Therefore, this mechanism might enhance generalization in test scenarios where the dog is either a friend or an enemy. Incorporating such a mechanism in LED-WM might be a promising direction for future work.

5.2 World model generalization

To evaluate the generalization of a world model, one pragmatic metric is to measure how its generated rollouts on unseen dynamics benefit policy learning. If the world model can generalize to unseen dynamics in test games, which effectively simulates these dynamics, a policy finetuned on these rollouts should improve in new games.

Finetuning procedure. Given a trained LED-WM, a trained policy from LED-WM, and a test game, LED-WM takes the initial observation and the manual as input to generate 60 synthetic trajectories. These trajectories are then used to determine whether the policy should be finetuned on this game. We estimate the value of the trained policy from the world model and finetune the policy if the estimated value is smaller than a pre-defined threshold. For each gradient update on the policy finetune, we generate 60 synthetic trajectories. We repeat this process in 2000 optimization steps. We illustrate the finetuning procedure in Appendix F with a Python-like format.

Table 4: World model generalization over S2-dev and S3 (MESSENGER) through finetune procedure. Finetune results are recorded in average sum of scores across five seeds.

Method	S1	S2	S2-dev	S3
LED-WM (Ours)	1.500 ± 0	-	1.4478 ± 0.01	-0.11 ± 0.05
After finetune	-	-	1.4513 ± 0.01	-0.01 ± 0.12

Evaluation metrics and results. We adopt the average sum of scores due to its robustness to the stochasticity of the environment. We show policy finetune results in Table 4 for MESSENGER. In MESSENGER, we show that the finetuning procedure improves the trained policy in S2-dev, [4] and in S3, demonstrating that the world model is generalizable to test trajectories. However, the absolute policy improvement is still limited in our experiments.

6 Related work

Due to space limit, we provide a detailed related work in Appendix B. In this section, we briefly review related work on **language-conditioned dynamics using model-based approach**. Recent efforts focus on integrating dynamics-descriptive language into world models, resulting in language-conditioned world models. Dynalang [23] shows that such world model improves policy's sample efficiency compared to model-free approaches. However, it does not demonstrate policy generalization in unseen games. Reader [6] shows that a MCTS planner can generalize to unseen games using a language-conditioned world model. Despite this, its environment (RTFM [46]) does not require language grounding to entities. Zhang et al. [43] introduce MESSENGER-WM, a compositional benchmark based on MESSENGER, and EMMA-LWM—a policy can generalize over unseen games from a language-based world model and expert demonstrations. Though sharing this same goal of policy generalization with our work, these studies rely on limiting assumptions. Planning with an MCTS tree in Reader, involves incurring computational cost to generate plans in inference time. This approach may not be practical for applications that require quick policy responses. On the other hand, EMMA-LWM requires expert demonstrations to use imitation learning and behavior cloning. This assumption may not always be feasible for every application. In contrast, our work lifts these assumptions and demonstrates policy generalization over unseen games in two environments that require language grounding: MESSENGER and MESSENGER-WM.

7 Conclusion

We develop an agent that can understand dynamics-descriptive language in interactive tasks. We adopt a model-based reinforcement learning (MBRL) approach, where a language-conditioned world model is trained through interactions with the environment, and a policy is learned from this world model. Unlike existing works, we do not require expert demonstrations or expensive planning during inference. Our method proposes Language-aware Encoder for Dreamer World Model (LED-WM). LED-WM adopts an attention mechanism to explicitly align language description to entities in the observation. We show that policies trained with LED-WM can generalize better to unseen games than existing baselines. We can also further improve the trained policy through fine-tuning on synthetic test trajectories generated by the world model.

8 Acknowledgment

We thank everyone from VIRL lab (Oregon State University), especially Skand and Akhil for their valuable feedback and discussions. The first author was personally supported by Amanda Putiza, Nguyen Thi Ngoc Anh, Ngo Thi Bich Lan, Tran Thanh Nhu, Bui Thuy Tien, and Nguyen Hoang Kieu Anh. This work is supported by NSF CAREER Award 2339676. We also thank the anonymous reviewers for their valuable feedback and suggestions.

[4] In S2-dev, we use Wilcoxon signed-rank [40] and hierarchical bootstrap sampling [8] corresponding to two levels of hierarchies in our experiments (episodes and run trials): at a 95% confidence level, hierarchical sampling indicates an improvement between 0.014 and 0.019.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *arXiv [cs.CV]*, November 2017. [Cited on pages 1, 13, and 14]

[2] Yonatan Bisk, Deniz Yuret, and Daniel Marcu. Natural language communication with robots. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 751–761. Association for Computational Linguistics, June 2016. [Cited on pages 1 and 14]

[3] Tianshi Cao, Jingkang Wang, Yining Zhang, and Sivabalan Manivasagam. Zero-shot compositional policy learning via language grounding. *arXiv*, April 2023. [Cited on page 14]

[4] Ching-An Cheng, Andrey Kolobov, Dipendra Misra, Allen Nie, and Adith Swaminathan. LLF-bench: Benchmark for interactive learning from language feedback. *arXiv*, December 2023. [Cited on page 13]

[5] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A platform to study the sample efficiency of grounded language learning. *arXiv*, December 2019. [Cited on page 13]

[6] Nicola Dainese, Pekka Marttinen, and Alexander Ilin. Reader: Model-based language-instructed reinforcement learning. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, page 16583–16599, Singapore, December 2023. Association for Computational Linguistics. [Cited on pages 1, 2, and 9]

[7] Nicola Dainese, Matteo Merler, Minttu Alakuijala, and Pekka Marttinen. Generating code world models with large language models guided by monte carlo tree search. *arXiv*, October 2024. [Cited on page 14]

[8] A C Davison and D V Hinkley. *Cambridge series in statistical and probabilistic mathematics: Bootstrap methods and their application series number 1*. Cambridge University Press, Cambridge, England, June 2013. [Cited on pages 9 and 25]

[9] Yilun Du, Mengjiao Yang, Bo Dai, Hanjun Dai, Ofir Nachum, Joshua B Tenenbaum, Dale Schuurmans, and Pieter Abbeel. Learning universal policies via text-guided video generation. *arXiv*, November 2023. [Cited on page 14]

[10] B Efron. Bootstrap methods: Another look at the jackknife. *Ann. Stat.*, 7(1):1–26, January 1979. [Cited on page 25]

[11] Prasoon Goyal, Scott Niekum, and Raymond J Mooney. Using natural language for reward shaping in reinforcement learning. *arXiv*, May 2019. [Cited on page 13]

[12] David Ha and Jürgen Schmidhuber. World models. March 2018. [Cited on page 2]

[13] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. *arXiv [cs.LG]*, November 2018. [Cited on page 3]

[14] Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models - new 2024. *arXiv*, April 2024. [Cited on pages 1, 3, 13, 14, and 20]

[15] Austin W Hanjie, Victor Zhong, and Karthik Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. *arXiv*, June 2021. [Cited on pages 1, 2, 4, 5, 7, 14, and 15]

[16] Nicklas Hansen, Hao Su, and Xiaolong Wang. TD-MPC2: Scalable, robust world models for continuous control. *arXiv*, October 2023. [Cited on pages 7 and 20]

[17] Haibo He and E A Garcia. Learning from imbalanced data. *IEEE Trans. Knowl. Data Eng.*, 21 (9):1263–1284, September 2009. [Cited on page 21]

[18] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. *arXiv*, February 2020. [Cited on page 14]

[19] Isaac Kauvar, Chris Doyle, Linqi Zhou, and Nick Haber. Curious replay for model-based adaptation. *arXiv*, June 2023. [Cited on page 21]

[20] Jacob Krantz and Stefan Lee. Sim-2-sim transfer for vision-and-language navigation in continuous environments. *arXiv*, April 2022. [Cited on pages 1 and 14]

[21] Jacob Krantz, Shurjo Banerjee, Wang Zhu, Jason Corso, Peter Anderson, Stefan Lee, and Jesse Thomason. Iterative vision-and-language navigation. *arXiv [cs.CV]*, October 2022. [Cited on page 13]

[22] Jacob Krantz, Shurjo Banerjee, Wang Zhu, Jason Corso, Peter Anderson, Stefan Lee, and Jesse Thomason. Iterative vision-and-language navigation. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14921–14930, Vancouver, BC, Canada, June 2023. IEEE. [Cited on page 13]

[23] Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. *arXiv [cs.CL]*, July 2023. [Cited on pages 1, 2, 3, 5, 7, 9, and 13]

[24] Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. *arXiv*, May 2023. [Cited on page 13]

[25] Sabrina McCallum, Max Taylor-Davies, Stefano V Albrecht, and Alessandro Suglia. Is feedback all you need? leveraging natural language feedback in goal-conditioned reinforcement learning. *arXiv*, December 2023. [Cited on page 13]

[26] Nikhil Mehta, Milagro Teruel, Patricio Figueroa Sanz, Xin Deng, Ahmed Hassan Awadallah, and Julia Kiseleva. Improving grounded language understanding in a collaborative environment by interacting with agents through help feedback. *arXiv*, February 2024. [Cited on page 13]

[27] Khanh Nguyen, Yonatan Bisk, and Hal Daumé, III. A framework for learning to request rich and contextually useful information from humans. *arXiv*, June 2022. [Cited on page 13]

[28] Meenal Parakh, Alisha Fong, Anthony Simeonov, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Lifelong robot learning with human assisted language planners. *arXiv*, October 2023. [Cited on page 13]

[29] Shaohui Peng, Xing Hu, Rui Zhang, Jiaming Guo, Qi Yi, Ruizhi Chen, Zidong Du, Ling Li, Qi Guo, and Yunji Chen. Conceptual reinforcement learning for language-conditioned tasks. *arXiv*, March 2023. [Cited on pages 2, 7, and 22]

[30] Skand Peri, Iain Lee, Chanho Kim, Li Fuxin, Tucker Hermans, and Stefan Lee. Point cloud models improve visual robustness in robotic learners. *arXiv*, April 2024. [Cited on pages 7, 14, and 20]

[31] Wasu Top Piriyakulkij, Yichao Liang, Hao Tang, Adrian Weller, Marta Kryven, and Kevin Ellis. PoE-world: Compositional world modeling with products of programmatic experts. *arXiv*, May 2025. [Cited on page 14]

[32] Rudra P K Poudel, Harit Pandya, Chao Zhang, and Roberto Cipolla. LanGWM: Language grounded world model. *arXiv*, November 2023. [Cited on page 14]

[33] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv*, September 2023. [Cited on page 5]

[34] Yi Ren, Samuel Lavoie, Mikhail Galkin, Danica J Sutherland, and Aaron Courville. Improving compositional generalization using iterated learning and simplicial embeddings. *arXiv*, October 2023. [Cited on page 13]

[35] Pratyusha Sharma, Balakumar Sundaralingam, Valts Blukis, Chris Paxton, Tucker Hermans, Antonio Torralba, Jacob Andreas, and Dieter Fox. Correcting robot plans with natural language feedback. *arXiv*, April 2022. [Cited on page 13]

[36] Allison C Tam, Neil C Rabinowitz, Andrew K Lampinen, Nicholas A Roy, Stephanie C Y Chan, D J Strouse, Jane X Wang, Andrea Banino, and Felix Hill. Semantic exploration from language abstractions and pretrained representations. *arXiv*, April 2023. [Cited on page 13]

[37] Hao Tang, Darren Key, and Kevin Ellis. WorldCoder, a model-based LLM agent: Building world models by writing code and interacting with the environment. *arXiv*, May 2024. [Cited on page 14]

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv*, August 2023. [Cited on page 6]

[39] Zhengyong Wang, Liquan Shen, Mei Yu, Kun Wang, Yufei Lin, and Mai Xu. Domain adaptation for underwater image enhancement. *arXiv*, August 2021. [Cited on page 14]

[40] Frank Wilcoxon. Individual comparisons by ranking methods. *Biom. Bull.*, 1(6):80–83, 1945. [Cited on pages 9 and 25]

[41] Tianbao Xie, Siheng Zhao, Chen Henry Wu, Yitao Liu, Qian Luo, Victor Zhong, Yanchao Yang, and Tao Yu. Text2Reward: Automated dense reward function generation for reinforcement learning. *arXiv*, September 2023. [Cited on page 13]

[42] Zhongwei Yu, Jingqing Ruan, and Dengpeng Xing. Explainable reinforcement learning via a causal world model. *arXiv*, May 2023. [Cited on page 13]

[43] Alex Zhang, Khanh Nguyen, Jens Tuyls, Albert Lin, and Karthik Narasimhan. Language-guided world models: A model-based approach to AI control. *arXiv*, July 2024. [Cited on pages 1, 2, 4, 7, 8, 9, and 14]

[44] Yang Zhang, Shixin Yang, Chenjia Bai, Fei Wu, Xiu Li, Zhen Wang, and Xuelong Li. Towards efficient LLM grounding for embodied multi-agent collaboration. *arXiv*, May 2024. [Cited on page 14]

[45] Zhaoheng Zheng, Haidong Zhu, and Ram Nevatia. CAILA: Concept-aware intra-layer adapters for compositional zero-shot learning. *arXiv*, May 2023. [Cited on page 13]

[46] Victor Zhong, Austin W Hanjie, Sida Wang, Karthik Narasimhan, and Luke Zettlemoyer. SILG: The multi-domain symbolic interactive language grounding benchmark. In *Advances in Neural Information Processing Systems*, volume 34, page 21505–21519. Curran Associates, Inc., 2021. [Cited on pages 9 and 14]

[47] Victor Zhong, Austin W Hanjie, Sida I Wang, Karthik Narasimhan, and Luke Zettlemoyer. SILG: The multi-environment symbolic interactive language grounding benchmark. *arXiv*, January 2022. [Cited on page 14]

[48] Victor Zhong, Jesse Mu, Luke Zettlemoyer, Edward Grefenstette, and Tim Rocktäschel. Improving policy learning via language dynamics distillation. *arXiv*, September 2022. [Cited on page 13]

[49] Siyu Zhou, Tianyi Zhou, Yijun Yang, Guodong Long, Deheng Ye, Jing Jiang, and Chengqi Zhang. WALL-E 2.0: World alignment by NeuroSymbolic learning improves world model-based LLM agents. *arXiv*, April 2025. [Cited on page 14]

[50] Siyuan Zhou, Yilun Du, Jiaben Chen, Yandong Li, Dit-Yan Yeung, and Chuang Gan. RoboDreamer: Learning compositional world models for robot imagination. *arXiv*, April 2024. [Cited on page 14]

Appendix A Training details

Hyperparameter	Value
Batch size	30
Batch length	300
Optimizer	Adam
World model learning rate	3e-4
Max. world model gradient norm	30
Actor learning rate	2e-4
Max. actor gradient norm	100
Critic learning rate	1e-4
Max. critic gradient norm	100

Table 5: Training hyperparameters.

Hyperparameter	Symbol	Value
Dynamics loss scale	β_{dyn}	1
Representation loss scale	β_{rep}	0.1
Latent unimix	—	1%
Free nats	—	1
Sentence embedding dim	d_s	32
Symbol/Agent embedding dim	d_{sb}	32
MLP layers	—	3
MLP hidden units	—	512
Query/key dim	d	128
Value dim	d_{val}	128
RSSM deterministic dim	—	512

Table 6: World model hyperparameters. Other hyperparameters are the same as in DreamerV3 [14].

Hyperparameter	Symbol	S1	S2	S2-dev	S3	MESSENGER-WM
Number of entities	N	3	3	3	5	3
Episode horizon	H	4	32	32	32	32
Finetune threshold	$thres$	-	1.2	1.2	1.4	-
Training environment steps	-	1M	10M	10M	20M	10M
Training GPU hours	-	6	24	24	72	24

Table 7: Environment hyperparameters. Training GPU hours are estimated based on 1 NVIDIA H100 GPU.

Appendix B Detailed related work

How language is used in RL tasks? Language is often employed as step-by-step instructions or goal specification in domains such as 1) visual language navigation (VLN) [21] [22] [1] 2) grid-world games like BabyAI [5] and SILG benchmark [48], and 3) manipulation tasks [34] [45] [28]. Another research direction in language for RL explores how language can accelerate policy learning by providing richer feedback rather than just numerical rewards: language for plan correction [35] [24] [25] [4] [27] [26], providing more descriptions of the current state or current goal [27] [23], generating dense rewards [11] [42] [41], clarifying information [26], and speeding up exploration [36]. This study investigates an alternative use of language in RL problems: describing the dynamics of environments.

Language-conditioned dynamics environments. In language-conditioned environments, while language can be used as step-by-step instructions [20] [2] [1], language can also be used to describe environment dynamnics—that is, *how environments change over time*. Formally, language describes the transition function $T(s'|s, a)$ and reward function $r(s, a)$ of a MDP system defined in Section 2. Several environments have been proposed to provide language-conditioned dynamics [3] [46] but their settings do not require understanding entity interaction or grounding language to entities. To fill this gap, as discussed in Section 3 about the environment setup, Hanjie et al. [15] present MESSENGER, a more challenging game requiring language grounding to entities based on environment dynamics. Zhang et al. [43] later propose MESSENGER-WM built out of MESSENGER to test compositional generalization world model and policy. In this study, we focus on MESSENGER and MESSENGER-WM due to their requirements for language grounding to entities and a history of previous works on these datasets.

How to solve language-conditioned dynamics environments? Generally speaking, there are two ways to understand language-conditioned dynamics: model-free and model-based learning. First, in model-free approach, language is used to build language-conditioned observation, which is then fed directly to policy in a model-free manner. Second, in model-based learning, language is used to build language-conditioned world model, which is then used to plan or learn policy. Our work focuses on the second category: we use dynamics-related language explicitly for dynamics learning, in model-based RL fashion, thus improving policy performance over model-free approaches.

Language-conditioned dynamics in RL: model-free approach. In model-free approach, language is used to build language-conditioned observation, which is then fed directly to policy. Hanjie et al. [15] utilize an attention mechanism to ground language to individual entities, forming a language-aware representation for the policy. Zhong et al. [47] develop a model of environmental dynamics learned from language-conditioned and state-only (without action) demonstrations. This dynamics model is then used to initialize and distill to the representation of a policy learner, which helps sample-efficiency and generalization across language RL tasks. Wang et al. [39] proposes compact and invariant concept-like representations through extracting similarities across observations, which is then proved to be useful for policy learning. Wang et al. [39] proposes two-agent system where the manager agent reads the instructions and manuals to devise the plan with sub-goals and the worker agent fulfills the sub-goals in the plan one-by-one. The model, however, assumes access to the sub-goal text instructions to train the manager. Those works use language directly for policy learning while our work uses language for world model learning.

Language-condition instruction-based world model (LWM). World models in model-based reinforcement learning (RL) involve learning the dynamics of the environment. In visual-understanding interaction domains like robotics and video games, world models have been widely studied and are empirically proven to be sample-efficient for policy learning [18] [14] [30]. However, in language-understanding interaction tasks, most language-conditioned world models have been developed to process task instructions or action descriptions, rather than to capture environmental dynamics. Poudel et al. [32] integrates human language into the world model, however language primarily describes observations rather than environment's dynamics. For example, a description like "there is an apple on the left, 2 meters from here" describes the future observation of environment without addressing the transition function. Recent works [50] [9] [44] develop LWM with compositional generalizability. While these works use more visually realistic input and require more reasoning to solve their tasks, the language they study is task instruction that involves straightforward mapping from language to objects such as colors and object names, e.g. "Move A Red Block to A Brown Box."

To bridge this gap, we focus on language-conditioned world models that incorporate dynamics-descriptive language—language that explains how entities interact and how the environment changes—rather than just providing direct task instructions. In contrast, the language used in our testbed environments MESSENGER and MESSENGER-WM describes environmental dynamics, which is the focus of our study.

Large language model (LLM) for world model. Recent work [7] [37] [31] [49] use LLMs to build a world model for policy learning. However, their environments does not require language understanding or language grounding. In other words, they do not build *a language-conditioned*

world model like our work. Further, they do not study the generalization behavior of LLM-based world model in a out-of-distribution (OOD) set up. In contrast, our work focuses on building a world model that requires language grounding to entities. We also study the generalization behavior of a language-conditioned world model and a policy learned from this world model. We achieve this by running experiments in a controlled OOD set up from MESSENGER and MESSENGER-WM.

Appendix C Environment details

C.1 MESSENGER

C.1.1 Environment Dynamics and Action

Reward and Game Ending. The agent loses the game and incurs -1 reward [5] if either of two events occurs - the agent is in the same cell as the enemy or reaches the goal without first getting the message. Reaching the messenger gives the agent a reward of 0.5, and then reaching the goal provides a reward of 1.

Observation change. The agent can be with or without a message, represented by two different symbols in the observation. The observation changes when the agent interacts with entities according to their roles, specifically:

- When the agent without a message picks up `messenger`, the messenger disappears from the observation. The agent now has a message and is represented by a different symbol from when it was without a message.
- When the agent loses the game, either by reaching the goal without first getting the message or by touching the enemy, the agent disappears from the observation.
- When the agent wins the game by reaching the goal with the message, the goal disappears from the observation.

Action. The agent can navigate the grid using five actions: `left`, `right`, `up`, `down`, and `stay`. The agent can only interact with entities when it is in the same grid cell as the entity. [6]

C.1.2 Entities and Language Manuals

Each game includes a language manual and an observation containing entities and a single agent. There are twelve different entities (e.g., `airplane`, `researcher`, etc.) denoted by a fixed set of corresponding symbols that are used consistently across game instances. For instance, symbol for entity `plane` shown in Figure 1. Note that the observation does not have entity names (e.g. `airplane`) and the agent must observe the entity's symbol and ground the entity's name to its corresponding symbols from the manual.

There are also three movement types for entities: `moving`, `fleeing`, and `stationary`, which describe movement trends relative to the agent's position. For example, a manual "heading closer and closer to where you are" describes the movement type `moving`.

For each game, the game engine assigns different roles (`enemy, goal, messenger`) and movement types (`moving, fleeing, stationary`) to a set of entities, along with the associated language manuals containing this information. For example, "the plane fleeing from you has the classified report". As a result, two games with the same set of entities and identical grid-world observations can have different language manuals and, consequently, different reward and transition functions.

[5] In the $S3$ setting of the game, there is an inconsistency in the environment implementation with the description provided in [15]: when the agent collides with two enemy entities, the environment returns a reward of -2 instead of -1. We observe that this rarely happens and thus has no significant impact on expected policy's behavior.

[6] We observed an inconsistency in the implementation with the environment description outlined in [15]. The agent can collide with entities even when they are not in the same grid cell. This is however deemed acceptable to the policy as the agent is still able to try to either go back or stay away from the other entity. More details can be found in this discussion: https://github.com/ahjwang/messenger-emma/issues/6

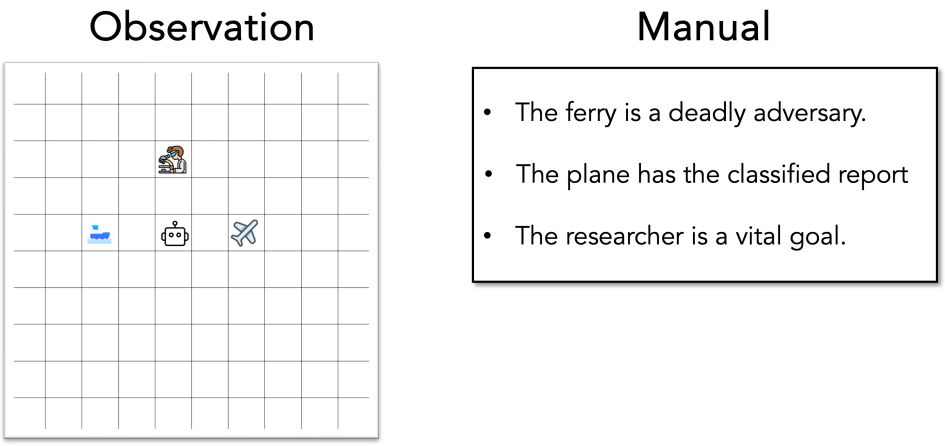


Figure 3: An example game of MESSENGER S1. In this game, the entity does not have message at the beginning of the game. Therefore, it goes to the messenger to retrieve the message and ends the game. All entities except the agent are stationary, thus the manual only describes roles associated with entity names.

C.1.3 Evaluation settings

MESSENGER provides three stages with different levels of language generalization assessment:

Stage 1 (S1). This stage tests the agent's ability to ground entity names in the manual to entity symbols in the observation. Test games offer two different levels of generalization evaluation. First, new languages describing the same entity name using synonyms, e.g. `researcher-scholar`. Second, new languages describing new combinations of known entities in a game, i.e. the agent has played with entities `ferry`, `plane`, `researcher` in train but not in the same game, and the agent is tasked to play with all of them in a test game.

As shown in Figure 3, this stage includes three entities, each with one of the three roles: `enemy`, `messenger`, and `goal`, along with their corresponding descriptions. All entities are stationary and placed two steps away from the agent, which starts in the center of the grid. The language descriptions only specify the entities and their roles, with no mention of movement. The agent starts the game either with or without the `message`.

Stage 2 (S2) and S2-dev. As shown in Figure 4 [7], S2 uses the same set of entities as S1 but introduces movement dynamics: entities can now exhibit one of three movement types: `moving`, `fleeing`, or `stationary`. The agent always starts without the `message`. During training games, only *one* movement combination is used: one `moving` (chasing), one `fleeing`, and one `stationary` entity, all of which describe how entities are moving compared to the agent. In test games, the agent must handle scenarios where a movement type can appear multiple times, e.g. `moving-moving-fleeing`. To examine the impact of this single-movement constraint, MESSENGER provides a different stage S2-dev, a variation of S2 that also features unseen dynamics but maintains the same movement constraint observed during training for all test games.

In addition to the capabilities demonstrated in S1, the objective of the agent in S2 is to generalize across new language featuring novel environmental dynamics. Specifically, the agent must understand the movement descriptions to make optimal actions, but does not need to ground movement descriptions to the entities based on their observed behaviors. This is because the agent can ground the sentences to the entities based on their names in the manual and their associated symbols in the observation. For example, given the game in Figure 1, the agent can ground the sentence "The plane fleeing from you has the classified report" to entity symbol  based on the entity name `the plane`-to-symbol  mapping. The agent must understand the entity's behavior to move closer to it,

[7] This figure is the duplication of Figure 1 and is put here for the sake of reading flow.

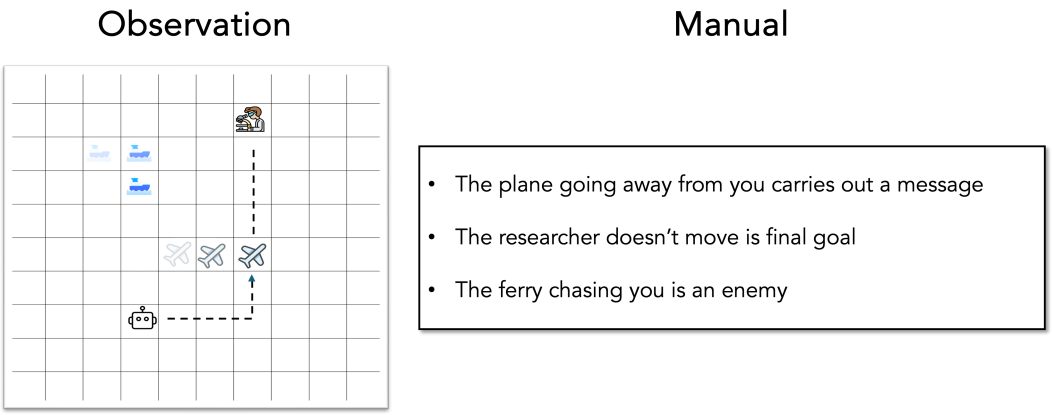


Figure 4: An example of a game play within a 10 × 10 grid-world from MESSENGER S2. The observation on the left includes three entities represented by their associated symbols: (`ferry` -), (`plane` -), (`researcher` -) and one agent (depicted by). The game involves three roles: `messenger`, `goal`, and `enemy`. The agent's task is to identify roles of all entities, locate the `messenger`, deliver it to the `goal`, and avoid the `enemy`. To achieve this objective, the agent must use the manual to infer entity roles based on their described dynamics and observed behavior. In the observation in the example, shaded icons indicate one possible scenario of entity locations over time. By observing entity movement patterns and grounding language to entities based on their according behaviors, the agent can infer the roles are assigned: `(ferry-enemy)`, `(plane-messenger)`, and `(researcher-goal)`. After inferring all entity roles, the agent can execute an appropriate plan to complete the task. The dashed line in the observation shows such a possible plan.

and it can achieve this based on the description "The plane fleeing from you", even without directly observing the behavior.

Stage 3 (S3). In addition to the capabilities demonstrated in Stage 1, the objective of the agent in Stage 3 is to generalize over new language featuring new combinations of known entity movement dynamics. Unlike in Stage 2, the agent in S3 must ground the sentences using both entity name-to-symbol mappings and observed entity behavior-to-movement description mappings.

As shown in Figure 5, this stage includes five entities, three retain the roles of `enemy`, `messenger`, and `goal`. The manual has six sentences featuring these five entities and one extraneous entity, which is not available in the observation. Specifically, its referred sentence has the same name as the `enemy` entity, but is different in movement and is described as either `goal` or `messenger`. Two additional entities are duplicates that share the same entity symbols and names of the `messenger` and `goal` accordingly, but they are assigned the role of `enemy`. To differentiate these entities, their movement dynamics must be used. For example, descriptions like "the fleeing enemy is the dog" versus "the chasing goal is the dog" help the agent identify the correct entities based on their behaviors. In this case, the dog that is consistently going towards the agent can be inferred as the goal.

C.2 Examples of different level of generalization evaluation in MESSENGER and MESSENGER-WM environment

We illustrate different levels of generalizations, described in Section 3.3, by examples in Figure 6. For each type of generalization, for the sake of simplicity, we consider a hypothetical dataset where a training set consists of only two samples, and one test sample. The best view is in colors.

C.3 MESSENGER-WM details

C.3.1 Differences between MESSENGER and MESSENGER-WM

Similar to MESSENGER S2, in MESSENGER-WM, there are three entities with three roles: `messenger`, `enemy`, and `goal`. However, unlike S2 where training games only have one movement combination `chasing-fleeing-stationary`, training games in MESSENGER-WM can

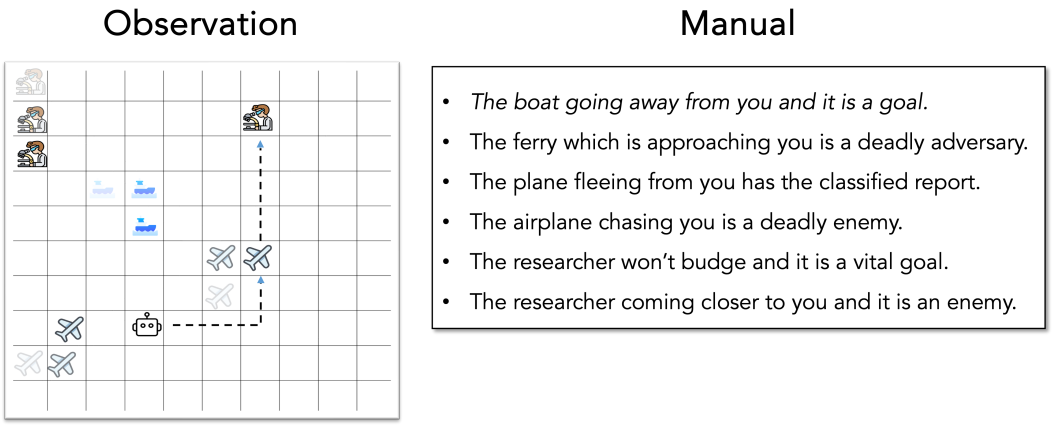


Figure 5: An example game of MESSENGER S3. To win the game, the agent must infer the roles of entities given the manual. Specifically, the same entity names (e.g. `airplane, plane` with different roles (e.g. `enemy, messenger`) must be disambiguated by their movement dynamics (e.g. `chasing, fleeing`). Note that we have a *italicized* sentence describing an extraneous entity that is not available in the game observation. We also have synonyms for entity names and roles, e.g., `airplane, plane; adversary, enemy`. The shaded entities show possible entity locations over time and the dashed line shows a possible path for the agent to win the game.

have more than one entity having the same movement pattern, e.g. `chasing-chasing-stationary`. This makes generalization in S2 is more challenging than MESSENGER-WM because the agent must overcome this data bias to generalize over unseen combinations of movement patterns.

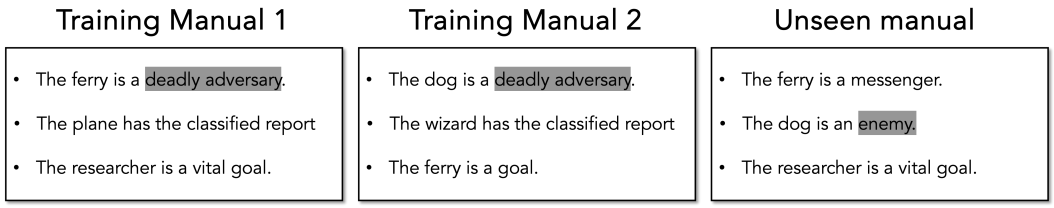
C.3.2 Evaluation settings

To help ground our descriptions of these settings, consider the following manual for a hypothetical test game:

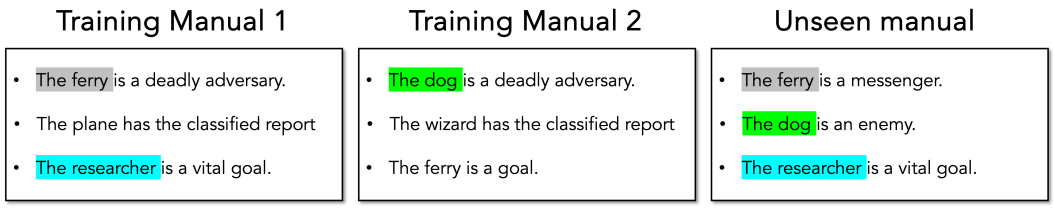
```
The hound is a deadly opponent.  It is towards you.
The whale comes towards you as the secret document.
It also has the crucial goal, the queen, and is something that
cannot be moved.
```

This manual describes the entity combination: `hound, whale, queen`, with the following feature assignments: `(hound-chasing-enemy)`, `(whale-chasing-messenger)`, `(queen-stationary-goal)`. Descriptions of each setting are as follows, based on whether the test game falls into each:

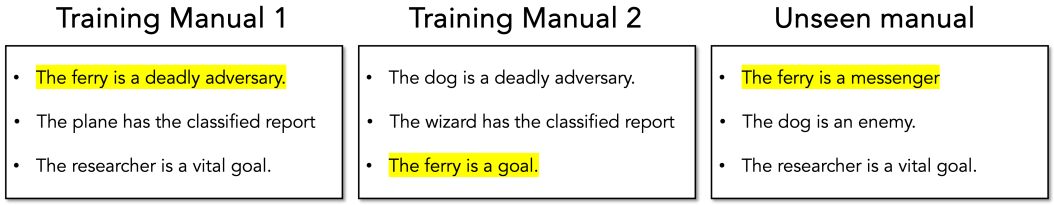
1. **NewCombo:** Each game represents an unseen combination of entities. However, any entity-role-movement combination in this set also presents in the training games. In this example, the agent never sees entity combination `(hound, whale, queen)` in the same game during training, although it can see each entity individually across different games.
2. **NewAttr:** Each game features seen combinations of entities, but at least one attribute (role, movement type, or both) for each entity is novel. In is example, the agent has seen entity combination `(hound, whale, queen)` during training but each entity-role-movement assignment is new: i.e. the assignment `(hound-chasing-enemy)` is unseen but `(hound-chasing-goal)` or `(hound-fleeing-messenger)` are seen during training.
3. **NewAll:** This setting combines the challenges of the first two. The combination of entities is novel, and each entity is assigned at least one new attribute. In this example, entity combinations `hound, whale, queen` and all entity-role-movement (i.e. `(hound-chasing-enemy)`, `(whale-chasing-messenger)`, `(queen-stationary-goal)`) are unseen.



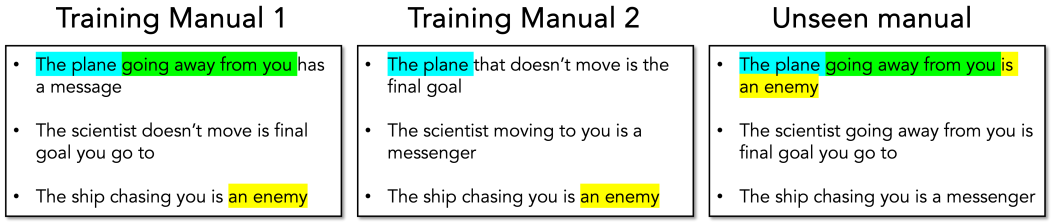
(a) Novel language through synonyms and paraphrase



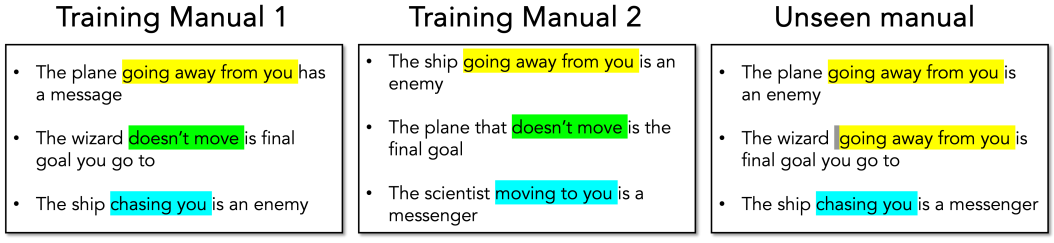
(b) Novel combinations of known entities. In test games, entities never appear in the same game during training.



(c) Novel entity-role combinations. In test games, at least one entity-role combination is unseen during training.



(d) Novel entity-role-movement combinations. In test games, at least one entity-role-movement combination is unseen during training.



(e) Novel combinations of known entity movements or novel game dynamics. In training games, there is only one movement combination `chasing`-`fleeing`-`stationary`. In test games, there are more than one movement combination.

Figure 6: Examples of different levels of generalization evaluation in MESSENGER and MESSENGER-WM environments.

Appendix D LED-WM details

Our world model LED-WM is built on the Recurrent State Space Model (RSSM) in [14]. However, we make the following modifications. First, we find that reconstruction decoder in DreamerV3 negatively impacts policy generalization. Therefore, we omit the decoder from DreamerV3 architecture. Second, to improve sample efficiency, we adopt multi-step prediction for reward and continue prediction [16] [30]. Specifically, we rollout the latent states in the future for H steps to supervise reward and continue prediction. We replace the encoder of DreamerV3 with our LED encoder, and keep the rest of the architecture unchanged, resulting in the following components:

$$\text{RSSM} \begin{cases} \text{Sequence model:} & h_t = f_\phi(h_{t-1}, z_{t-1}, a_{t-1}) \\ \textbf{LED} \text{ (Section 4.2):} & z_t \sim q_\phi(z_t \mid h_t, x_t) \\ \text{Dynamics predictor:} & \hat{z}_t \sim p_\phi(\hat{z}_t \mid h_t) \\ \text{Reward predictor:} & \hat{r}_t \sim p_\phi(\hat{r}_t \mid h_t, z_t) \\ \text{Continue predictor:} & \hat{c}_t \sim p_\phi(\hat{c}_t \mid h_t, z_t) \end{cases} \tag{8}$$

World model loss. Given a sequence of observations $o_{1:T}$, actions $a_{1:T}$, rewards $r_{1:T}$ and continuation flags $c_{1:T}$ where T is the horizon of a training episode, we optimize the world model parameters ϕ to minimize the following loss:

$$\mathcal{L}(\phi) \doteq \mathbb{E}_{q_\phi}\left[\sum_{t=1}^{T}\left(\beta_{\text{pred}}\mathcal{L}_{\text{pred}}(\phi) + \beta_{\text{dyn}}\mathcal{L}_{\text{dyn}}(\phi) + \beta_{\text{rep}}\mathcal{L}_{\text{rep}}(\phi)\right)\right], \tag{9}$$

where $\mathcal{L}_{pred}$, $\mathcal{L}_{dyn}$, and $\mathcal{L}_{rep}$ are prediction loss, dynamics loss, and representation loss, along with their corresponding weights β_{pred}, β_{dyn} and β_{rep}:

1. Prediction loss ($\mathcal{L}_{pred}$): trains the reward predictor via symlog loss and the continue predictor via binary classification loss. To address the issue of slower training caused by removing the decoder and observational reconstruction loss, we adopt multi-step prediction for reward and continue [16] [30], to improve sample efficiency. We rollout the latent states in the future for H steps to supervise reward and continue prediction. Specifically, given a state-action trajectory over $H+1$ step $(x_t, a_t, x_{t+1}, \ldots, x_{t+H})$ associated with a sequence of rewards $r_{t:t+H}$ and continue flags $c_{t:t+H}$, we first compute z_t as the posterior state from x_t. We then rollout this z_t over H steps to get prior states $\hat{z}_{t+1\ldots t+H-1}$ and deterministic states $h_{t+1\ldots t+H}$ to predict rewards $r_{t+1:t+H}$ and continue flags $c_{t:t+H}$:

$$\begin{aligned} \mathcal{L}_{pred} = & \underbrace{-\ln p_\theta(r_t|z_t,h_t)}_{\text{reward loss}} \underbrace{-\ln p_\theta(c_t|z_t,h_t)}_{\text{continue loss}} \\ & \underbrace{-\sum_{k=t+1}^{H} \lambda^{k-t-1}\left[\ln p_\theta(r_k \mid \hat{z}_k, h_k) + \ln p_\theta(c_k \mid \hat{z}_k, h_k)\right]}_{\text{multi-step reward and continue loss}}, \end{aligned} \tag{10}$$

$$\tag{11}$$

   where $\lambda = 0.9$ is a discount factor when the environment is stochastic and $\lambda = 1$ when the environment is deterministic. Recall that $\hat{z}_k$ denotes the prior stochastic state generated by the world model, without access to observation at time step k.

2. Dynamics and representation loss: We adopt the dynamics and representation loss unchanged from Dreamerv3.

Appendix E Training procedure

During world model training, we observe two challenges. First, at the beginning of training, successful episodes in which the agent wins the game and receives positive rewards are rare. As a result, the

world model takes longer to learn from these rare instances, leading to reduced sample-efficiency in policy training. Second, as the policy converges and produces mostly successful episodes, the replay buffer becomes dominated by these episodes. This causes the world model to rarely encounter failed episodes where the agent loses the game and receives negative rewards, potentially harming its generalization performance. We therefore adapt the following strategies to improve sample efficiency:

Prioritized Replay Buffer. We adopt the Prioritized Replay Buffer from [19], where the authors propose the following prioritization strategies:

- Count-based replay: Biases sampling towards recent experiences in the replay buffer.
- Adversarial replay: Prioritizes experiences where the world model makes incorrect predictions.

Balanced weights. We adopt a balanced weighting technique for handling class imbalance, inspired by methods used in classification tasks [17], and apply it to world model training. This weighting ensures that underrepresented classes contribute proportionally more to the training loss, improving sample-efficiency of the policy.

In our setting, for a given episode e with T is the episode horizon and the state-action trajectory: $\tau_e = (o_1, a_1, \ldots, o_T, a_T)$, we define the "class" c_e as the accumulated sum of rewards for the episode:

$$c_e = \sum_{t=1}^{T} r_t(o_t, a_t), \tag{12}$$

representing different gameplay scenarios. For example, in the MESSENGER environment, episodes fall into three classes: 1.5, -0.5, and -1.

To address the imbalance between training instances of negative classes and positive class in the replay buffer, we scale the world model loss $\mathcal{L}_{\text{pred}}$ in each class proportional to the inverse square root of its frequency in the replay buffer. The scaled loss is computed as:

$$\mathcal{L}_{\text{pred}} = \mathcal{L}_{\text{pred}} \times \sqrt{\frac{|RB|}{\text{count}(c)}}, \tag{13}$$

where $|RB|$ is the total number of episodes in the replay buffer RB, and count(c) is the number of instances of class c in the replay buffer.

Increase throughput for replay buffer. In the original Dreamerv3 implementation, one trajectory with L time steps of observations in the replay buffer is duplicated L times, making the training data throughput inefficient. We therefore remove this duplication to speed up the training throughput in the replay buffer, resulting in more sample-efficient.

Appendix F Finetune a trained policy using a trained world model

Algorithm 1: Policy Finetune with LED-WM

Input: The trained LED-WM, the trained policy π, a test game G with the first observation obs_o and a language manual L.
Output: A finetuned policy $\hat{\pi}$ if needed

Function `EstimateReturn` *(π, LED-WM, obs_o, L)*:
    returns = []
    for _ in range(60): `// Generate synthetic test trajectories`
        traj = LED-WM.GenerateTrajectory(obs_o, L)
        returns.append(sum_rewards(traj))
    $\hat{V}_\pi$ = mean(returns)
    return $\hat{V}_\pi$

Function `FineTune` *(π, LED-WM, obs_o, L)*:
    for gradient_step in range(2000):
        trajectories = []
        for _ in range(60):
            trajectories.append(LED-WM.GenerateTrajectory(obs_o, L))
        π.train(trajectories)
    return π

`// Main function: Finetune the policy` π `using LED-WM`
Function *PolicyFinetune(LED-WM, π, obs_0, L)*:
    $\hat{V}_\pi$ = `EstimateReturn(`*LED-WM*, π, obs_0, L`)`
    if $\hat{V}_\pi >= thres$:
        $\hat{\pi} = \pi$
    else:
        $\hat{\pi}$ = `FineTune(`π, *LED-WM*, G, obs_0, L`)`
    return $\hat{\pi}$

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: Our claim about policy generalization and world model generalization in Section 1 are reflected in our experiment results in Section 5.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss our limitations in policy generalization where CRL [29] outperforms LED-WM in S2 in Section 5.

Guidelines:

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

3. **Theory assumptions and proofs**

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

   Answer: [Yes]

   Justification: In Section 1, we stated that our assumption is that our observation is symbolic and confied to a discrete grid-world.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: The paper provides the code and hyperparameters for training the world model and the policy in Section 5 and Appendix A.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

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

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

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

   Answer: [Yes]

   Justification: We provide the code and hyperparameters for training the world model and the policy in Section 5 and Appendix A.

   Guidelines:

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

- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We adopt MESSENGER and MESSENGER-WM environments which follow their standard train/dev/test split. We provide our code and necessary hyperparameters in Appendix A.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We use Wilcoxon signed-rank [40], bootstrap sampling [10], and hierarchical bootstrap sampling [8] to do statistical tests for policy finetune results in Table 4 in S2-dev.

   Guidelines:

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

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provide training time and our used GPU information in Table 7.

   Guidelines:

   - The answer NA means that the paper does not include experiments.

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

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and the paper conforms to the code of ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [NA]

    Justification: We do not have any potential positive or negative societal impacts.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [NA]

Justification: We do not have any data or models that have a high risk for misuse.

Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We cited the original papers for the environments and other baselines.

Guidelines:

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

13. **New assets**

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

Answer: [Yes]

Justification: We will release the code and instructionsfor our experiments.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

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

Answer: [NA]

Justification: We do not involve crowdsourcing nor research with human subjects.

Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

Answer: [NA]

Justification: We do not involve research with human subjects.

Guidelines:

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

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: we only use LLM for editing purposes.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.