LANGUAGE-GUIDED WORLD MODELS A MODEL-BASED APPROACH TO AI CONTROL

*Alex Zhang[◇], *Khanh Nguyen[♠], Jens Tuyls[◇], Albert Lin[♣], Karthik Narasimhan[◇]
[◇] Princeton University [♠]University of California, Berkeley [♣]University of Southern California

ABSTRACT

Installing probabilistic world models into artificial agents opens an efficient channel for humans to communicate with and control these agents. In addition to updating agent policies, humans can modify their internal world models in order to influence their decisions. The challenge, however, is that currently existing world models are difficult for humans to adapt because they lack a natural communication interface. Aimed at addressing this shortcoming, we develop *Language-Guided World Models* (LWMs), which can capture environment dynamics by reading language descriptions. These models enhance agent communication efficiency, allowing humans to simultaneously alter their behavior on multiple tasks with concise language feedback. They also enable agents to self-learn from texts originally written to instruct humans. To facilitate the development of LWMs, we design a challenging benchmark based on the game of MESSENGER (Hanjie et al., 2021), requiring compositional generalization to new language descriptions and environment dynamics. Our experiments reveal that the current state-of-the-art Transformer architecture performs poorly on this benchmark, motivating us to design a more robust architecture. To showcase the practicality of our proposed LWMs, we simulate a scenario where these models augment the interpretability and safety of an agent by enabling it to generate and discuss plans with a human before execution. By effectively incorporating language feedback on the plan, the models boost the agent performance in the real environment by up to three times without collecting any interactive experiences in this environment.

Project website: language-guided-world-model.github.io

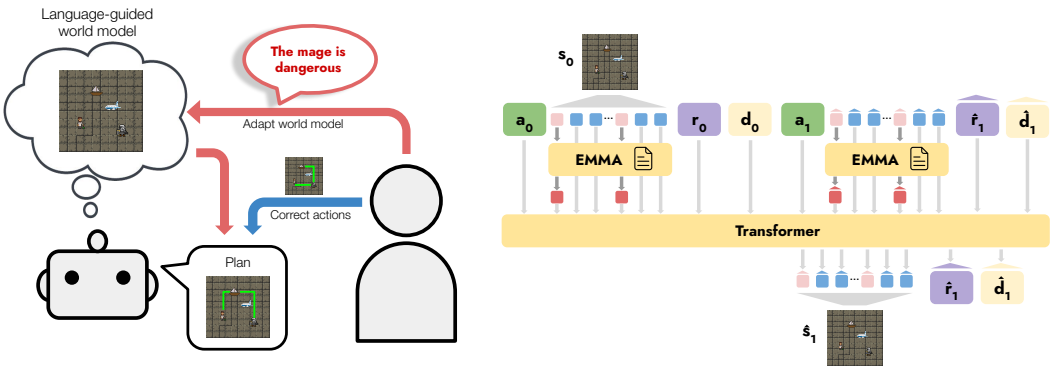
1 INTRODUCTION

Model-based agents are artificial agents equipped with probabilistic "world models" that are capable of foreseeing the future state of an environment (Deisenroth & Rasmussen, 2011; Schmidhuber, 2015). This endows the agents with the ability to plan and learn in imagination (i.e., internal simulation) and has led to exciting results in the field of reinforcement learning (Finn & Levine, 2017; Ha & Schmidhuber, 2018; Chua et al., 2018; Hafner et al., 2023). Most previous work has leveraged this approach to improve the autonomous performance of artificial agents.

In this paper, we endorse and enhance the model-based approach for a different purpose—strengthening the controllability of artificial agents. A model-based agent comprises a policy and a world model, where the policy is optimized with respect to the world model. This modular design can result in more effective control over "model-free" counterparts, allowing humans to alter the agent's behavior both *explicitly*, by directly updating its policies (e.g., through reinforcement or imitation learning), or *implicitly*, by adapting its world model (which subsequently also changes the policies). The latter mechanism has the potential to influence the agent performance across all tasks in the environment, saving substantial human communication effort. For example, telling a cleaning robot that "*the floor is slippery*" can effectively remind it to handle every object in a room with greater caution.

Current model-based agents are difficult for humans to adapt because they employ purely observational world models. These models can be modified only through observational data, which is not a

*First two authors contribute equally. Correspondence email: kxnguyen@berkeley.edu.



(a) (b)

Figure 1: Language-guided world models can significantly improve the interpretability, safety, and helpfulness of artificial agents. (a) These models enable an agent to compose intuitive plans and invite a human supervisor to validate and revise those plans. Moreover, they offer the supervisor various strategies for revising a plan: they can provide action-correcting feedback to update the agent's policy, or language feedback that describes the environment to modify its world model. (b) We design an effective architecture for language-guided world models which exhibits strong compositional generalizability. Our approach converts a trajectory into a long sequence of tokens and trains a Transformer to auto-regressively generate these tokens. It implements a specialized attention mechanism inspired by Hanjie et al. (2021) to incorporate textual information into the observation tokens.

suitable medium for humans to convey complex intentions (Sumers et al., 2023; Zheng et al., 2023). To overcome the limitations of observational world models, we develop ***L**anguage-Guided **W**orld **M**odels* (LWMs)—world models that can be effectively steered through human verbal communication. Agents equipped with LWMs inherit all the benefits of model-based agents while also being able to incorporate human language-based supervision. Language-based learning reduces human effort, as well as the amount of exploratory observations that these agents need to collect in an environment, mitigating the risk of them taking harmful actions due to insufficient knowledge. LWM-based agents can also self-improve by reading "free" texts composed for guiding humans (e.g., game manuals), lowering subsequent effort to fine-tune them through direct interaction.

Building these models poses a unique research challenge of how to accurately ground language to aspects of an environment's dynamics, which has been understudied in previous work. This problem is tremendously difficult because it requires a model to understand diverse linguistic concepts (e.g., motion, interaction, appearance, spatial/temporal relation, etc.) that can possibly be used to describe the world. To address this problem, we first construct a benchmark based on the MESSENGER work of Hanjie et al. (2021), which constructs environments whose dynamics are described by language descriptions depicting attributes of environment entities. We design settings of increasing difficulty, requiring a world model to generalize *compositionally* with respect to both the language input and the environment dynamics. Our learning approach starts by converting a training trajectory into a token-based representation that is convenient for the integration of the dynamics features extracted from the descriptions. We find that the prominent Transformer model architecture (Vaswani et al., 2017) is ineffective in learning to translate language descriptions into such a representation. Its attention mechanism lacks inductive biases to learn the right feature-extraction function and instead overfits to spurious correlations. We implement a new attention mechanism inspired by the Entity Mapper with Multi-Modal Attention (Hanjie et al., 2021), which mimics a two-step reasoning process. Our results confirm the effectiveness of this architecture, as it robustly generalizes even on the hardest evaluation setting, outperforming various Transformer-based baselines by substantial margins. It is even competitive with an oracle model with a perfect semantic-parsing capability.

Besides conducting intrinsic evaluation, we illustrate a promising application of LWMs. Concretely, we simulate a cautious agent that, instead of performing a task right away, uses its LWM to generate an execution plan and asks a human to review it. This form of pre-execution communication can potentially improve the agent's interpretability and safety. Moreover, the performance of the agent can also be enhanced by incorporating human feedback on the plan. We train all model-based agents

with action-correcting feedback via imitation learning. Our LWM-based agent has the advantage of also being able to incorporate *language feedback* describing the environment dynamics. We demonstrate that the language understanding capabilities of our LWM are sufficient to improve the agent's capabilities dramatically. In our hardest evaluation setting, without gathering additional interactions in the real environment, this agent achieves an average cumulative reward that is up to three times that of an agent trained without language supervision.

We hope that our work will serve as a catalyst for exploring novel approaches that enable humans to tap into the world models of artificial agents. More generally, we call for the design of modular agents whose components are parameterized by natural language. We hypothesize that such agents would allow for effective incorporation of diverse linguistic signals from humans, potentially surpassing the currently prevalent model-free approaches that use language that aims at only controlling a policy (i.e. instructions) (Bisk et al., 2016; Misra et al., 2018; Anderson et al., 2018) or feed language conveying diverse intentions directly to a black-box policy (e.g., Narasimhan et al. (2018); Hanjie et al. (2021); Zhong et al. (2021) and work on large language models like Ouyang et al. (2022)).

2 BACKGROUND: WORLD MODELS

We consider a Markov Decision Process (MDP) environment E with state space $\mathcal{S}$, action space $\mathcal{A}$, and transition function $M(s_{t+1}, r_{t+1}, d_{t+1} \mid s_t, a_t) : \mathcal{S} \times \mathcal{A} \to \Delta(\mathcal{S} \times \mathbb{R} \times \{0, 1\})$, where Δ denotes the set of all probability distributions over a set. An agent implementing a policy $\pi(a \mid s) : \mathcal{S} \to \Delta(\mathcal{A})$ interacts with the environment by choosing actions using its policy. Taking an action $a_t \sim \pi(s_t)$ in state s_t transitions the agent to a new state s_{t+1}, and incurs a reward r_{t+1} and a termination signal d_{t+1}, where $s_{t+1}, r_{t+1}, d_{t+1} \sim M(s_t, a_t)$.

A (one-step) *world model* (Robine et al., 2023; Micheli et al., 2023; Hafner et al., 2023) approximates $M(s_{t+1}, r_{t+1}, d_{t+1} \mid s_t, a_t)$. A *model-based agent* learns this model in addition to its policy and leverages it to plan or optimize the policy. To improve this type of agent, being able to adapt only the policy is insufficient. Learning an optimal policy with respect to an erroneous world model still results in suboptimal behavior in the real world. Therefore, it is essential to be able to also alter the world model.

The dominant approach to world modeling learns a function $M_\theta(s_{t+1}, r_{t+1}, d_{t+1} \mid h_t)$ parameterized by a neural network θ and conditioned on a history $h_t = (s_1, r_1, d_1, a_1, \ldots, s_t, r_t, d_t, a_t)$. We refer to this class of models as *observational world models* because they can be adapted with only observational data, through either in-weight learning (updating the model parameters to fit a dataset of observations), or in-context learning (plugging in a history of observations).

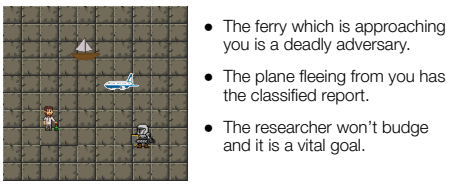


Figure 2: MESSENGER environment.

Relying on observation-based adaptation leads to two major drawbacks of observational world models. First, humans cannot effectively control them, because observations are inadequate for conveying complex, abstract human intentions. Second, collecting observations requires taking real actions in the environment, which can be expensive, time-consuming, and risky. We elucidate these drawbacks using the game of MESSENGER (Hanjie et al., 2021). In an example game shown in Figure 2, the ferry plays the role of an enemy who will immediately purge the player and end the game once it collides with them. An observational world model cannot determine the role of this entity without witnessing it killing the player, which is an undesirable event to trigger. Past experiences could be misleading: for example, the model might have seen the ferry playing only other roles in the past, and therefore infer that it is unlikely to be an enemy. The game manual reveals the identity of the enemy, but the model is unable to utilize such information. The only way for a human to communicate the enemy role to this model is to draw observations illustrating the player's death after colliding with the entity, which requires a lot of manual effort.

3 LANGUAGE-GUIDED WORLD MODELS (LWMS)

LWMs can interpret language descriptions to adapt to new environment dynamics. This class of models addresses the two aforementioned drawbacks of observational world models. They allow humans to easily adapt their behavior through natural language. Consequently, humans can effectively assist these models by providing language-based information, significantly reducing the amount of interactive experiences needed in real environments. They can also leverage texts written for humans to quickly learn new environment dynamics.

3.1 FORMULATION

Our formal setup augments the MDP setting by assuming that the environment is accompanied by a *language manual* $\ell = \{l_1, \cdots, l_N\}$ consisting of text descriptions l_i that disclose information about the transition function. Our goal is to learn a model $M_\theta(s_{t+1}, r_{t+1}, d_{t+1} \mid h_t, \ell)$ that can interpret the manual ℓ to better predict s_{t+1}, r_{t+1}, and d_{t+1}. We first present the general training and evaluation procedures in this section and describe the specific architecture that we implemented in the next section.

Training. Let $P^{\text{train}}_{\mathcal{S},\mathcal{A}}(E)$ be a distribution over manual-augmented environments with state space $\mathcal{S}$ and action space $\mathcal{A}$. The training data for our world model is a set of tuples $\mathcal{B}_{\text{train}} = \{(\tau^i, \ell^i)\}_{i=1}^{|\mathcal{B}_{\text{train}}|}$ where τ^i is a trajectory generated in an environment $E^i \sim P^{\text{train}}_{\mathcal{S},\mathcal{A}}$ and ℓ^i is the manual accompanying that environment. Each trajectory $\tau = (s_1, r_1, d_1, a_1, \ldots, s_T, r_T, d_T)$ is a sequence of states, actions, rewards, and termination signals. These trajectories are generated using a behavior policy π, which can be a rule-based or learned policy, or a human. We train an LWM as an auto-regressive sequence generator to minimize the cross-entropy loss on the training data

$$\min_\theta J(\theta; \mathcal{B}_{\text{train}}) \triangleq \min_\theta -\frac{1}{|\mathcal{B}_{\text{train}}|} \sum_{(\tau,\ell)\in\mathcal{B}_{\text{train}}} \frac{1}{T_\tau - 1} \sum_{t=1}^{T_\tau - 1} \log M_\theta(s^\tau_{t+1}, r^\tau_{t+1}, d^\tau_{t+1} \mid h^\tau_t, \ell) \quad (1)$$

where T_τ is number of time steps in τ and h^τ_t is a (learned) representation of τ up to time step t.

Evaluation. Previous work mostly evaluates world models on downstream control tasks (Ha & Schmidhuber, 2018; Finn & Levine, 2017; Hafner et al., 2023). While it serves a practical end goal, this extrinsic evaluation scheme does not truly measure the quality of a world model since the result also depends on the strength of the search or RL algorithm used. Since our focus is to learn accurate world models that can be employed for different purposes (including control tasks), we also conduct intrinsic evaluation of the models. We estimate the probabilistic divergence with respect to the true model by computing the approximate cross entropy $J(\theta; \mathcal{B}_{\text{eval}})$, where $\mathcal{B}_{\text{eval}}$ contains trajectories generated in previously unseen environments.[1] We also use the learned model to generate imaginary trajectories and compute the precision of the next state, reward, and termination-signal predictions.

3.2 MODEL

Learning LWMs poses a challenging problem that involves the retrieval and incorporation of information expressed in different modalities. Due to the complexity of the general problem, we consider a simplified setting in which the manual conveys information about attributes of entities in an environment, and the environment state follows an entity-based representation. Despite this simplification, our setting remains highly general and is by no means easy to tackle. As we will show, the state-of-the-art Transformer architecture struggles to generalize compositionally in this setting. In this section, we present a more effective modeling approach. Our proposed architecture is illustrated in Figure 1b.

Entity-based state representation. We focus on 2D grid-world environments featuring a set of C entities. Each entity c has an *identity* (e.g., a mage) and K *attributes* (e.g., movement pattern, role). Each identity is represented by an index $i \in [I]$, where I is the number of identities. Similarly, each attribute value is an index $v_k \in [V_k]$ where V_k is the number of distinct values of attribute k. The

[1] See Appendix E for a precise mathematical interpretation of this metric.

identity and attributes of an entity vary among environment instances. The entities can travel in an environment and interact with one another, and the outcomes of their interactions are decided on the basis of their attributes.

A state s in these worlds is represented by an $H \times W$ grid with C channels (an $H \times W \times C$ tensor), where each channel corresponds to a single entity. In each channel c, there is a single non-zero cell $s(h, w, c) \in [I]$ that represents the identity and position of entity c.

World modeling as sequence generation. Our model is an encoder-decoder Transformer which encodes a manual ℓ and decodes a trajectory τ. We transform the trajectory into a long sequence of tokens and train the model as a sequence generator. Prior work (Robine et al., 2023; Micheli et al., 2023; Hafner et al., 2023) employs a latent-variable model to discretize observations into tokens. However, for this problem, to effectively incorporate information extracted from the manual, we require an observation representation that disentangles the entities (i.e., the states and attributes of the entities must be captured by disjoint sets of tokens). Such a representation is easily derived for entity-based observations without having to implement a latent-variable model. We defer the problem of learning entity-disentangled representation for pixel-based observations to future work.

Our model processes a data point (τ, ℓ) as follows. For the manual $\ell = \{l_i\}_{i=1}^{N}$, we first use a pre-trained BERT model to convert each description l_i into a sequence of hidden vectors. We feed each sequence to a Transformer encoder, which outputs a tensor $\boldsymbol{m}^{\text{enc}}$ of size $N \times L \times D$, where N is the number of descriptions, L is the maximum number of words in a description, and D is the hidden size.

For the trajectory, we convert each tuple (a_{t-1}, s_t, r_t, d_t) into a token block B_t. The first action a_0 is set to be a special <s> token. Each state s_t is mapped to $3C$ tokens $(i_t^1, h_t^1, w_t^1, \cdots, i_t^C, h_t^C, w_t^C)$, which represents each of the C entities by its identity i followed by its coordinates (h, w) in the grid. The real-valued reward r_t is discretized into an integer label, and the termination signal d_t is translated into a binary label. In the end, B_t consists of $3C+3$ tokens $(a_{t-1}, i_t^1, h_t^1, w_t^1, \cdots, i_t^C, h_t^C, w_t^C, r_t, d_t)$. Finally, we concatenate all T_τ blocks in the trajectory into a sequence of $T_\tau \times (3C + 3)$ tokens, embed them into a $T_\tau \times (3C + 3) \times D$ tensor, and add positional embeddings. We will use bold notation (e.g. $\boldsymbol{a}$, $\boldsymbol{i}$) to refer to the resultant embeddings of the tokens.

Entity mapper with multi-modal attention. We implement a variant of EMMA (Hanjie et al. (2021)) that first identifies the manual description that mentions each entity and extracts from it words corresponding to the attributes of the entity. Formally, from the tensor $\boldsymbol{m}_n^{\text{enc}}$ computed by the encoder, we generate a key tensor $\boldsymbol{m}^{\text{key}}$ and a value tensor $\boldsymbol{m}^{\text{val}}$, both of which are of size $N \times L \times D$, where

$$\boldsymbol{m}_n^{\text{key}} = \texttt{Softmax}(\texttt{Linear}_{\text{key}}^{D \to 1}(\boldsymbol{m}_n^{\text{enc}})^\top)\boldsymbol{m}_n^{\text{enc}} \tag{2}$$

$$\boldsymbol{m}_n^{\text{val}} = \texttt{Softmax}(\texttt{Linear}_{\text{val}}^{D \to 1}(\boldsymbol{m}_n^{\text{enc}})^\top)\boldsymbol{m}_n^{\text{enc}} \tag{3}$$

for $1 \leq n \leq N$. Here, $\texttt{Linear}_{\text{key}}^{D \to 1}$ and $\texttt{Linear}_{\text{val}}^{D \to 1}$ are linear layers that transform the input's last dimension from D to 1, and $\texttt{Softmax}(\cdot)$ applies the softmax function to the last dimension. Intuitively, we want each $\boldsymbol{m}_n^{\text{key}}$ to retain words that signal the identity of the entity mentioned in the n-th description (e.g., ferry, plane, researcher), and $\boldsymbol{m}_n^{\text{val}}$ to capture words that convey the attributes of the entity (e.g., approaching, deadly, fleeing).

Let $\boldsymbol{i}_t^c$ be the embedding of the identity of entity c. We perform a dot-product attention with $\boldsymbol{i}_t^c$ as the query, $\boldsymbol{m}^{\text{key}}$ as the set of keys, and $\boldsymbol{m}^{\text{val}}$ as the set of values to compute the attribute features $\boldsymbol{z}_t^c$ of entity c

$$\boldsymbol{z}_t^c = \texttt{DotAttend}(\boldsymbol{i}_t^c, \boldsymbol{m}^{\text{key}}, \boldsymbol{m}^{\text{val}}) \tag{4}$$

The features are incorporated to the input of the decoder Transformer at each time step t as follows:

$$(\boldsymbol{a}_{t-1}, \boldsymbol{i}_t^1 + \boldsymbol{z}_t^1, \boldsymbol{h}_t^1, \boldsymbol{w}_t^1, \cdots, \boldsymbol{i}_t^C + \boldsymbol{z}_t^C, \boldsymbol{h}_t^C, \boldsymbol{w}_t^C, \boldsymbol{r}_t, \boldsymbol{d}_t) \tag{5}$$

Unlike a standard encoder-decoder Transformer model, our model does not perform cross-attention between the encoder and the decoder because information from the encoder has already been incorporated into the decoder through EMMA.

ChatGPT-based hard attention. We also attempt an alternative approach that leverages the language-understanding capabilities of ChatGPT. Through few-shot prompting, we instruct this model to determine the identity of the entity mentioned in each manual description. In this approach, we generate only the set of values $\boldsymbol{m}^{\text{val}}$ as in Eq 3. Instead of learning soft attention, we directly route the values to the identity embeddings. Concretely, the feature vector added to $\boldsymbol{i}_t^c$ in Eq 5 is $\boldsymbol{z}_t^c = \boldsymbol{m}_{\hat{i}_c}^{\text{val}}$ where $\hat{i}_c \in [N]$ is the index of the description that mentions entity c according to ChatGPT.

Model training. We train the model to minimize cross-entropy loss with respect to ground-truth (tokenized) trajectories in the training set. The label at each output position is the next token in the ground-truth sequence. Particularly, we do not compute the losses at the positions of the action tokens and the first block's tokens, because those tokens will be set during inference.

4 THE MESSENGER-WM BENCHMARK: EVALUATING THE COMPOSITIONAL GENERALIZABILITY OF WORLD MODELS

We suppose that the dynamics of an environment is governed by independent latent factors. A robust world model must adapt robustly to changes in the values of these factors, generalizing to novel compositions of these values. We construct a benchmark to evaluate this capability of world models.

Environment. Our benchmark is built on top of the MESSENGER environment and dataset (Hanjie et al., 2021). MESSENGER is a role-playing game that takes place in a 10×10 grid world. A player interacts with entities having one of three *roles*: message, goal, or enemy. The objective of the player is to acquire the message and deliver it to the goal while avoiding the enemy. In addition to the role, each entity is assigned an *identity* among twelve possibilities (mage, airplane, orb, etc.) and a *movement pattern* (chasing the agent, fleeing from the agent, immobile).

Each game features a combination $\{(i^c, v^c_{\text{move}}, v^c_{\text{role}})\}_{c=1}^3$ where i^c denotes the identity of the entity c, v^c_{move} its movement pattern, and v^c_{role} its role. There is exactly one entity for each role. The movement patterns and roles of the entities are the latent factors that dictate the environment dynamics; the identity is irrelevant, but observable.

Game manual. A game's manual consists of three descriptions. The authors of MESSENGER collected a data set of 5,316 language descriptions, each of which describes a combination of identity, role, and movement. Manual descriptions employ various linguistic expressions for each identity, role, or movement pattern (e.g., an airplane can be mentioned as a "plane", "jet", or "airliner"), making it nontrivial to parse the referred identities and attributes. Evaluation descriptions are completely unseen during training.

Evaluating compositional generalization. We create three levels of compositional generalization:

- **NewCombo (easy).** Each game features a combination of three identities that were never seen together in a training game. However, the attributes (role and movement pattern) of each identity are the same as during training.
- **NewAttr (medium).** The three identities were seen together in a training game, but each identity is assigned at least a new attribute (role, or movement pattern, or both).
- **NewAll (hard).** This setting combines the difficulties of the previous two. The identity triplet is novel, and each identity is assigned at least a new attribute.

To generate trajectories, we implement rule-based behavior policies that execute various intentions: act randomly, avoid the enemy, suicide (go to the enemy), obtain the message, and win the game (obtain the message and deliver it to the goal). We generate a total of 100K trajectories for training, each of which is generated by rolling out a uniformly randomly chosen rule-based policy. More details of the data are given in Appendix A.

Challenging world models. Modeling the dynamics of a new environment without interactions is daunting, especially when the training data deliberately present strong spurious correlations to deceive models. When a world model is deployed for planning or learning, it must make decisions based on self-generated histories. Every error made in one time step propagates to all future time

Table 1: Cross entropy losses (↓) of different models on test ground-truth trajectories. Note that the minimal losses are non-zero because MESSENGER is stochastic. We run each model with five different random seeds, selecting the final checkpoint for each seed based on the loss in the development NewAll split. We report the mean losses with 95% t-value confidence intervals. The bold number in each column indicates the best non-oracle mean. `EMMA-LWM` demonstrates robustness in the more difficult splits (NewAttr and NewAll), approaching the performance of an oracle model with a perfect semantic-parsing capability.

World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
`Observational`	0.120 ± 0.037	0.182 ± 0.015	0.187 ± 0.012
`Standard`	0.101 ± 0.036	0.145 ± 0.042	0.157 ± 0.034
`GPTHard`	0.096 ± 0.021	0.150 ± 0.007	0.155 ± 0.004
`EMMA-LWM`	**0.083** ± 0.005	**0.104** ± 0.016	**0.128** ± 0.014
`OracleParse`	0.084 ± 0.011	0.093 ± 0.024	0.117 ± 0.055

steps. Our hardest evaluation setting ensures that it is highly unlikely that a model can accurately predict the first few time steps without understanding the manual. In general, our evaluation is more comprehensive than the stage-two evaluation of the original MESSENGER paper, which does not construct different levels of compositional generalization, and is more difficult than that of Lin et al. (2023), which does not test for generalization.

5 EXPERIMENTS

5.1 BASELINES

We compare our proposed model, which we call `EMMA-LWM`, with the following models:

(a) **`Observational`** world model learns from only observations. It is identical to `EMMA-LWM` except that the manual representation $\boldsymbol{m}^{\text{enc}}$ is a zero matrix rather than a representation of the text manual;
(b) **`Standard`** is the encoder-decoder Transformer model introduced by Vaswani et al. (2017) with multi-headed cross-attention between the decoder and the encoder. Similarly to `EMMA-LWM`, the model uses BERT to initially encode the manual into hidden vectors. The encoder applies self-attention to the hidden vectors of each description separately, instead of joining all vectors into a sequence and applying self-attention on it;
(c) **`GPTHard`** implements the ChatGPT-based hard attention approach described in §3.2;
(d) **`OracleParse`** assumes oracle semantic parsing. It is similar to the `GPTHard` approach except that the manual is parsed into canonical identity and attribute names. For example, a description *"the crucial target is held by the wizard and the wizard is fleeing from you"* is converted into *"mage fleeing goal"* for the model.

We train all models using the AdamW optimizer (Loshchilov & Hutter, 2017) for 10^5 iterations with a learning rate of 10^{-4} and a batch size of 32. For further details, please refer to Appendix C.

5.2 INTRINSIC EVALUATION

Evaluation with ground-truth trajectories. Table 1 shows the cross-entropy losses of all models on ground-truth trajectories sampled from the true environment dynamics. In the more difficult NewAttr and NewAll splits, our `EMMA-LWM` model consistently outperforms all baselines, nearing the performance of the `OracleParse` model. As expected, the `Observational` model is easily fooled by spurious correlations between identity and attributes, and among attributes. A specific example is illustrated in Figure 3. There, the `Observational` model incorrectly captures the movement of the whale and the queen. It also mistakenly portrays the whale as an enemy, whereas, in fact, the entity holds the message. In contrast, `EMMA-LWM` is capable of interpreting the previously unseen manual and accurately simulates the dynamics of the environment.

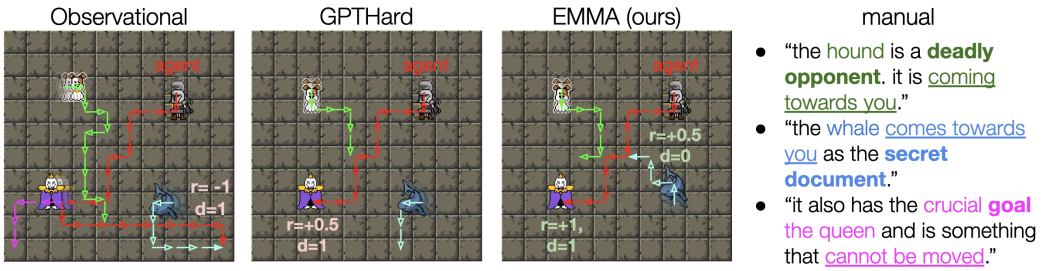


Figure 3: A qualitative example taken from the NewAll split. The `Observational` model mistakenly captures the movement patterns of the immobile queen goal and the chasing whale message. It also misrecognizes the whale as an enemy, predicting a wrong reward r and incorrectly predicting a termination state d after the player collides with this entity. The `GPTHard` model incorrectly identifies the queen as the message and predicts the whale to be fleeing. Meanwhile, our model `EMMA-LWM` accurately captures all of those roles and movements.

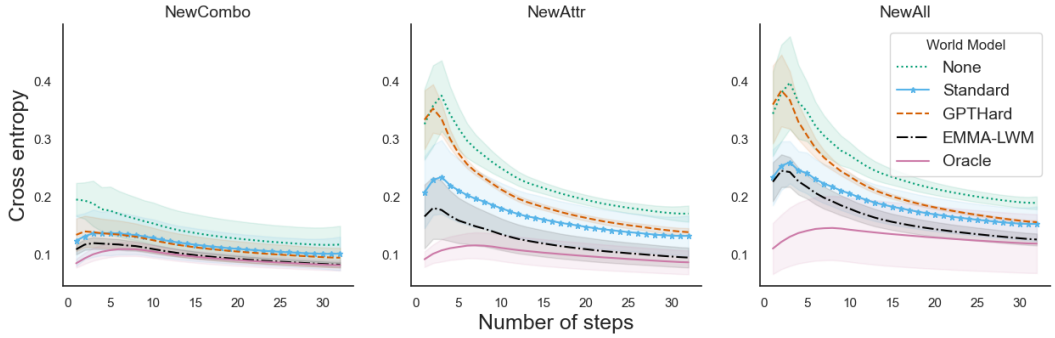


Figure 4: The cross entropy losses of the models when conditioned on ground-truth trajectory prefixes up to a certain length. We plot the means with 95% t-value confidence intervals. The losses generally decrease as the prefix length increases. `EMMA-LWM` outperforms baselines given any prefix length.

The performance of the `Standard` model is sensitive to initialization; in some runs, it performs as well as `EMMA-LWM`, but in others it performs as badly as `Observational`. A plausible explanation is that the model's attention mechanism lacks sufficiently strong inductive biases to consistently find generalizable solutions. Our results agree with previous work on the lack of compositional generalizability of Transformers, which is often remedied by adding numerous forms of inductive bias (Keysers et al., 2020; Jiang & Bansal, 2021; Chaabouni et al., 2021; Dziri et al., 2023).

Another interesting finding is that the `GPTHard` model does not perform as well as we expected. As a reminder, this model relies on ChatGPT to parse identities from descriptions and only needs to learn to extract attributes. Its underperformance compared to `EMMA-LWM` can be attributed to (i) the imperfection of ChatGPT in identifying identities in descriptions (its accuracy is around 90%; see Appendix A) and (ii) the fact that `EMMA-LWM` jointly learns to extract both identity and attribute words, which may be more effective than learning to extract only attribute words.

Figure 4 studies the performance of the models when conditioned on prefixes of the ground-truth trajectories. The losses of all models decrease as the prefix length increases, but the baselines cannot close the gaps with `EMMA-LWM`. Across all splits, `EMMA-LWM` conditioned on a one-step history outperforms `Observational` conditioned on one third of a ground-truth trajectory, demonstrating that our model has effectively leveraged the textual information.

Evaluation with imaginary trajectories. In this evaluation, for each world model and test trajectory, we reset the model to the initial state of the trajectory and sequentially feed actions in the trajectory to the model until it predicts the end of the episode. This process generates an imaginary trajectory. We refer to the evaluation trajectory as the real trajectory. We compute precisions of

Table 2: Results on imaginary trajectory generation. Δ_{dist} measures the similarity between the distances from the player to an entity in a real trajectory and the corresponding imaginary trajectory. The bold number in each column represents the best non-oracle result. `EMMA-LWM` outperforms all baselines in all metrics.

	Δ_{dist} (↓)			Non-zero reward precision (↑)			Termination precision (↑)		
World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)	NewCombo (easy)	NewAttr (medium)	NewAll (hard)	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
`Observational`	2.04	2.91	3.00	0.39	0.20	0.15	0.51	0.33	0.28
`Standard`	0.82	1.48	1.68	0.68	0.43	0.50	0.75	0.55	0.62
`GPTHard`	0.89	2.74	2.89	0.75	0.34	0.25	0.79	0.45	0.45
`EMMA-LWM`	**0.57**	**1.14**	**1.29**	**0.88**	**0.69**	**0.70**	**0.88**	**0.75**	**0.71**
`OracleParse`	0.49	0.77	0.92	0.93	0.81	0.77	0.89	0.84	0.79

Table 3: Average returns (↑) in real environments of policies trained with imaginary imitation learning using world models. For each world model type, we randomly choose one out of the five checkpoints mentioned in Table 1. Experiments are conducted in 30 environments for each test split. For each environment and learned policy, we compute the average return over 48 runs. We report the means of the average returns in the 30 environments with 95% t-value confidence intervals. Bold numbers indicate the best non-oracle means in the corresponding settings. Results of all models are available in Appendix F.

Setting	World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
Online IL *(near-optimal expert)*	`Observational`	0.75 ± 0.16	-0.41 ± 0.21	-0.21 ± 0.21
	`EMMA-LWM` (ours)	**1.01** ± 0.12	**0.96** ± 0.17	**0.62** ± 0.21
	`OracleParse`	1.04 ± 0.13	0.85 ± 0.20	0.91 ± 0.18
Filtered BC *(near-optimal expert)*	`Observational`	0.77 ± 0.14	-0.42 ± 0.15	-0.30 ± 0.16
	`EMMA-LWM` (ours)	**1.18** ± 0.10	**0.75** ± 0.20	**0.44** ± 0.18
	`OracleParse`	1.17 ± 0.11	0.84 ± 0.19	0.80 ± 0.18
Filtered BC *(suboptimal expert)*	`Observational`	0.71 ± 0.15	-0.35 ± 0.18	-0.33 ± 0.17
	`EMMA-LWM` (ours)	**0.98** ± 0.13	**0.29** ± 0.25	**0.13** ± 0.19
	`OracleParse`	1.09 ± 0.13	0.50 ± 0.24	0.49 ± 0.18

predicting non-zero rewards ($r \neq 0$) and terminations ($d = 1$). To evaluate movement prediction, we compare the distances from the player to an entity in the real and imaginary trajectories. Concretely, let $\delta^{\text{real}}_{c,t}$ and $\delta^{\text{imag}}_{c,t}$ be the Hamming distances from the player to entity c at the t-th time step in a real trajectory τ_{real} and an imaginary trajectory τ_{imag}, respectively. We calculate the average difference in a specific time step: $\Delta_{\text{dist}} = \frac{1}{|\mathcal{D}_{\text{eval}}|} \sum_{\tau_{\text{real}} \in \mathcal{D}_{\text{eval}}} \frac{1}{T_{\text{min}}} \sum_{t=1}^{T_{\text{min}}} |\delta^{\text{real}}_{c,t} - \delta^{\text{imag}}_{c,t}|$ where $\mathcal{D}_{\text{eval}}$ is an evaluation split, $T_{\text{min}} = \min(|\tau_{\text{real}}|, |\tau_{\text{imag}}|)$, and τ_{imag} is generated from τ_{real} . For example, for a chasing entity, $\delta^{\text{real}}_{c,t}$ decreases as t increases. If a model mistakenly predicts the entity to be immobile, $\delta^{\text{imag}}_{c,t}$ remains a constant as t progresses. In this case, Δ_{dist} is non-negligible, indicating an error. All evaluation metrics are given in Table 2. The ordering of the models is similar to that in the evaluation with ground-truth trajectories. `EMMA-LWM` is still superior to all baselines in all metrics.

5.3 Application: Agents That Discuss Plans With Humans

In this section, we showcase the effectiveness of our LWM in facilitating plan discussion between an agent and its human supervisor, which consequently improves the interpretability, safety, and performance of the agent.

We imagine an agent ordered to perform a task in a previously unseen environment (Figure 1a). Letting the agent perform the task immediately would be extremely risky because of its imperfect knowledge of the environment. Implementing a world model enables the agent to *imagine* a solution

trajectory and present it to a human as a plan for review. Conveying plans as trajectories helps the human envision the future behavior of the agent in the real world. Furthermore, the human can improve this behavior by providing feedback for the agent to revise its plan.

The human can provide feedback that rectifies the agent's actions. However, enhancing the agent's plan requires correcting not only its actions, but also its world model. Although the agent can update its world model on its own by collecting extra experiences in the real environment, doing so would defy the very purpose of having a plan discussion, which is to prevent the agent from acting recklessly in the real world. An LWM resolves this problem by **enabling the world model to learn from language feedback given by the human**. An observational world model would require much more effort from the human to communicate feedback; they have to be able to generate observations in the same format as those in the agent's plans (e.g., they have to draw grids). Furthermore, abstract concepts (e.g., "slippery") may not be precisely conveyed through a few images.

Simulating this scenario, we implement a randomly initialized policy (the agent) learning to solve a MESSENGER environment with an expert policy (the human supervisor). The learning policy is forbidden to interact with the real environment until test time. Learning is entirely imaginary: all trajectories are generated using a world model. Importantly, the world model was never trained on any data collected in the real environment, simulating the fact that the environment is previously unseen. After training, we run the learned policy in the real environment and record its average return. We consider two types of action-correcting feedback: in the *online imitation learning* setting (Ross et al., 2011), the expert corrects actions in agent-generated trajectories; in the *filtered behavior cloning* setting, the expert presents demonstrative trajectories. In the latter setting, before each learning episode, we add a set of demonstrative trajectories to a buffer and retain only those that achieve the highest returns in the buffer for the policy to imitate. We experiment with a near-optimal expert and a suboptimal expert.

We present the results in Table 3. Learning with the `Observational` world model amounts to the case where the human provides only action-correcting feedback and cannot adapt the world model. Meanwhile, learning with `EMMA-LWM` represents the case where the human can use language feedback to improve the world model. In all evaluation settings, we observe significant improvements in the average return of policies that adopt our `EMMA-LWM`. There are still considerable gaps compared to using the `OracleParse` model, indicating that our model still has room for improvement.

6 RELATED WORK

World models. World models have a rich history dating back to the 1980s (Werbos, 1987). The base architectures of these models have evolved over time, from feed-forward neural networks (Werbos, 1987), to recurrent neural networks (Schmidhuber, 1990b;a; 1991), and most recently, Transformers (Robine et al., 2023; Micheli et al., 2023). In RL settings, world models serve as the key component of model-based approaches, which use these models for training policies to reduce the amount of interactions with real environments. Model-based RL has been successfully applied to a variety of robotic tasks (Finn & Levine, 2017) and video games (Hafner et al., 2019; 2020; 2023). However, the incorporation of language information into world models has been underexplored. Cowen-Rivers & Naradowsky (2020) propose to language-conditioned world models but focus on emerging language rather than human language. Poudel et al. (2023) ground natural language to learn representation for world models. The closest to our work is that of Lin et al. (2023) but the authors do not demonstrate the ability to generalize beyond the training environments.

Adapting models through language. Language information has been incorporated into various aspects of learning. In instruction following (Bisk et al., 2016; Misra et al., 2018; Anderson et al., 2018; Nguyen & Daumé III, 2019), agents are given descriptions of the desired behaviors and learn to interpret them to perform tasks. Language-based learning (Nguyen et al., 2021; Scheurer et al., 2023) employs verbal descriptions and evaluations of agent behaviors to offer richer learning signals than traditional feedback such as numerical rewards or reference actions. Closest to our work is the line of work that exploits language descriptions of environment dynamics to improve policy learning (Narasimhan et al., 2018; Branavan, 2012; Hanjie et al., 2021; Wu et al., 2023a; Nottingham et al., 2022; Zhong et al., 2020). Rather than using texts to directly improve a policy, our work leverages them to enhance a model of an environment. Recently, several papers propose agents that can read

text manuals to play games (Wu et al., 2023a;b). Our work differs from these papers in that we aim to build models that capture exactly the transition function of an environment.

7 Conclusion

In this work, we introduce *Language-Guided World Models*, which can be adapted through natural language. These models have numerous advantages over traditional observational world models. We demonstrate the challenges of building these models even in a grid-world environment and present an approach that outperforms strong baselines. Our model is still lacking in performance and the grid-world environments we experiment with severely underrepresent the real world. Nevertheless, we hope that this work helps envision the potential of LWMs in enhancing the safety and utility of artificial agents for humans, and inspires future efforts to further develop these models.

Acknowledgements

We thank Ameet Deshpande, Vishvak Murahari, and Howard Chen from the Princeton NLP group for valuable feedback, comments, and discussions. We thank Kurtland Chua for helpful feedback. This material is based upon work supported by the National Science Foundation under Grant No. 2239363. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

References

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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3674–3683, 2018.

Yonatan Bisk, Deniz Yuret, and Daniel Marcu. Natural language communication with robots. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 751–761, 2016.

SRK Branavan. Learning to win by reading manuals in a monte-carlo framework. *Journal of Artificial Intelligence Research*, 43:661–704, 2012.

Rahma Chaabouni, Roberto Dessì, and Eugene Kharitonov. Can transformers jump around right in natural language? assessing performance transfer from scan. In *BlackboxNLP workshop (EMNLP)*, 2021.

Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

Alexander I Cowen-Rivers and Jason Naradowsky. Emergent communication with world models. *arXiv e-prints*, pp. arXiv–2002, 2020.

Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *Proceedings of the 28th International Conference on machine learning (ICML-11)*, pp. 465–472, 2011.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jian, Bill Yuchen Lin, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D Hwang, et al. Faith and fate: Limits of transformers on compositionality. In *Proceedings of Advances in Neural Information Processing Systems*, 2023.

Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2786–2793. IEEE, 2017.

David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Austin W Hanjie, Victor Y Zhong, and Karthik Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. In *International Conference on Machine Learning*, pp. 4051–4062. PMLR, 2021.

Yichen Jiang and Mohit Bansal. Inducing transformer's compositional generalization ability via auxiliary sequence prediction tasks. In *Proceedings of Empirical Methods in Natural Language Processing*, 2021.

Daniel Keysers, Nathanael Schärli, Nathan Scales, Hylke Buisman, Daniel Furrer, Sergii Kashubin, Nikola Momchev, Danila Sinopalnikov, Lukasz Stafiniak, Tibor Tihon, et al. Measuring compositional generalization: A comprehensive method on realistic data. In *Proceedings of the International Conference on Learning Representations*, 2020.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, Pieter Abbeel, Dan Klein, and Anca Dragan. Learning to model the world with language. *arXiv preprint arXiv:2308.01399*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *Proceedings of the International Conference on Learning Representations*, 2023.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*, 2018.

Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Grounding language for transfer in deep reinforcement learning. *Journal of Artificial Intelligence Research*, 63:849–874, 2018.

Khanh Nguyen and Hal Daumé III. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. *arXiv preprint arXiv:1909.01871*, 2019.

Khanh X Nguyen, Dipendra Misra, Robert Schapire, Miroslav Dudík, and Patrick Shafto. Interactive learning from activity description. In *International Conference on Machine Learning*, pp. 8096–8108. PMLR, 2021.

Kolby Nottingham, Alekhya Pyla, Sameer Singh, and Roy Fox. Learning to query internet text for informing reinforcement learning agents. *arXiv preprint arXiv:2205.13079*, 2022.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35: 27730–27744, 2022.

Rudra PK Poudel, Harit Pandya, Chao Zhang, and Roberto Cipolla. Langwm: Language grounded world model. *arXiv preprint arXiv:2311.17593*, 2023.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *Proceedings of the International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=TdBaDGCpjly.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Jérémy Scheurer, Jon Ander Campos, Tomasz Korbak, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. Training language models with language feedback at scale. *arXiv preprint arXiv:2303.16755*, 2023.

Jürgen Schmidhuber. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *1990 IJCNN international joint conference on neural networks*, pp. 253–258. IEEE, 1990a.

Jürgen Schmidhuber. *Making the world differentiable: on using self supervised fully recurrent neural networks for dynamic reinforcement learning and planning in non-stationary environments*, volume 126. Inst. für Informatik, 1990b.

Jürgen Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991.

Jürgen Schmidhuber. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *arXiv preprint arXiv:1511.09249*, 2015.

Theodore R Sumers, Mark K Ho, Robert D Hawkins, and Thomas L Griffiths. Show or tell? exploring when (and why) teaching with language outperforms demonstration. *Cognition*, 232:105326, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Paul J Werbos. Learning how the world works: Specifications for predictive networks in robots and brains. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics, NY*, 1987.

Yue Wu, Yewen Fan, Paul Pu Liang, Amos Azaria, Yuanzhi Li, and Tom Mitchell. Read and reap the rewards: Learning to play atari with the help of instruction manuals. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023a.

Yue Wu, So Yeon Min, Shrimai Prabhumoye, Yonatan Bisk, Ruslan Salakhutdinov, Amos Azaria, Tom Mitchell, and Yuanzhi Li. Spring: Studying papers and reasoning to play games. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Ruijie Zheng, Khanh Nguyen, Hal Daumé III, Furong Huang, and Karthik Narasimhan. Progressively efficient learning. *arXiv preprint arXiv:2310.13004*, 2023.

Victor Zhong, Tim Rocktäschel, and Edward Grefenstette. Rtfm: Generalising to new environment dynamics via reading. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SJgob6NKvH.

Victor Zhong, Austin W. Hanjie, Sida I. Wang, Karthik Narasimhan, and Luke Zettlemoyer. Silg: The multi-environment symbolic interactive language grounding benchmark. In *Neural Information Processing Systems (NeurIPS)*, 2021.

A DATASET

Split		Unique games	Unique descriptions	Trajectories	ChatGPT identity-parsing accuracy (%)
Train		1,536	986	101,376	92
Dev	NewCombo	896	598	4,480	89
	NewAttr	204	319	1,020	88
	NewAll	856	1,028	4,280	86
Test	NewCombo	896	587	4,480	90
	NewAttr	204	306	1,020	93
	NewAll	856	1,016	4,280	88

Table 4: MESSENGER data statistics. The last column shows the fraction of games in each split in which ChatGPT correctly identifies all three identities in a game.

Statistics of our dataset are provided in Table 4. The maximum trajectory length is 32. We implement five rule-based behavior policies: `survive` (avoid the enemy and goal), `win the game`, `suicide` (go to the enemy), `obtain the message`, and `act randomly`. The `survive` policy acts randomly when the distances to the enemy and the goal are greater than or equal to 6. Otherwise, it takes the action that makes its distance to those entities at least 3. If that is impossible, it chooses the action that maximizes the minimum distance to one of the two entities. The `win the game` policy is not optimal: it simply aims to obtain the message and then run to the goal, without having a strategy to avoid the enemy. We run a breadth-first search to find the next best action to get to an entity.

For the training split, we generate 66 trajectories per game. The behavior policy for each trajectory is chosen uniformly randomly among the five rule-based policies. For each evaluation split, we generate 5 trajectories per game, using every rule-based policy to generate trajectories.

B CHATGPT PARSING PROMPT

The following is the prompt that we compose for parsing descriptions. We use the "May 3, 2023" release of ChatGPT. We feed to the model one description at a time instead of a whole manual of three descriptions. We ask it to also extract the role and movement pattern, but use only the parsed identity in the `GPTHard` model. The "ChatGPT identity-parsing" column in Table 4 shows the fraction of games in each split in which ChatGPT correctly identifies all three identities in a game.

```
You are playing a role-playing video game where you will need to read
    textual descriptions to figure out the attributes of a character.

This is a list of characters and their corresponding IDs:
airplane: 2
mage: 3
dog: 4
bird: 5
fish: 6
scientist: 7
thief: 8
ship: 9
ball: 10
robot: 11
queen: 12
sword: 13

This is a list of movement types and their corresponding IDs:
chasing: 0
fleeing: 1
stationary: 2

This is a list of role types and their corresponding IDs:
dangerous enemy: 0
```

```
secret message: 1
essential objective: 2

Now, read a description and tell me which character is being mentioned
    and what are its movement type and role type. Your answer should
    follow this format:
```
Answer: Character ID, movement type ID, role type ID
```

Here are a few examples:

Description: the plane that's flying near where you are is the critical
    objective.
Answer: 2, 0, 2

Description: the escaping humanoid is an important goal.
Answer: 11, 1, 2

Description: the mage is inching near you is a lethal opponent.
Answer: 3, 0, 0

Description: the classified document is the hound coming your way.
Answer: 4, 0, 1

Description: the important goal is the orb which is creeping close to you
    .
Answer: 10, 0, 2

Now provide the answer for the following description. Follow the format
    of the previous answers:

Description: [PLACEHOLDER]
```

C TRAINING DETAILS

Hyperparameter	Value
Hidden size	256
Number of encoder layers	4
Number of decoder layers	4
Number of decoder token blocks	33 (max. trajectory length + 1)
Dropout rate	0.1
Batch size	32
Number of training batches	100K
Evaluation every	500 batches
Optimizer	AdamW
Learning rate	1e-4
Max. gradient norm	10

Table 5: Training hyperparameters.

Our implementation of Transformer is largely based on the IRIS codebase(Micheli et al., 2023).[2] We implement cross-attention for the `Standard` baseline, and EMMA for our model.

[2] https://github.com/eloialonso/iris

Initialization. We find that the default PyTorch initialization scheme does not suffice for our model to generalize compositionally. We adopt the following initialization scheme from the IRIS codebase:

```
def init_weights(module):
    if isinstance(module, (nn.Linear, nn.Embedding)):
        module.weight.data.normal_(mean=0.0, std=0.02)
        if isinstance(module, nn.Linear) and module.bias is not None:
            module.bias.data.zero_()
    elif isinstance(module, nn.LayerNorm):
        module.bias.data.zero_()
        module.weight.data.fill_(1.0)
```

which is evoked by calling `self.apply(init_weights)` in the model's constructor. We initialize all models with this scheme, but only `EMMA-LWM` and `OracleParse` perform well consistently on various random seeds.

D IMITATION LEARNING EXPERIMENTS

The learning policy follows the EMMA-based policy architecture of Hanjie et al. (2021), which at each time step processes a stack of 3 most recent observations with a convolution-then-MLP encoder. We train the policy with 2,000 batches using the same optimizer hyperparameters as those of the world models.

For the online IL setting, we use the `win the game` rule-based policy (Appendix A) as the expert. For the filtered BC setting, we train an EMMA policy to overfit the test environment. We then use a fully converged checkpoint of the policy as the near-optimal expert, and a not fully converged checkpoint as the suboptimal expert. The former is trained for 10,000 iterations and the latter is trained for 2,000 iterations.

The test environments are randomly chosen from the test splits. We select 10 environments per split. We evaluate each policy for 48 episodes in the real environment. These episodes cover all 24 initial configurations of a stage-two MESSENGER game.

E MATHEMATICAL INTERPRETATION OF THE CROSS ENTROPY LOSS

Let $P_\pi(h \mid E)$ be the distribution over histories in environment E induced by executing a behavioral policy π. We evaluate an LWM on environments drawn from a distribution $P^{\text{eval}}_{\mathcal{S},\mathcal{A}}(E)$, which may not necessarily be the same as the training distribution $P^{\text{train}}_{\mathcal{S},\mathcal{A}}(E)$. Let $\mathcal{B}_{\text{eval}}$ be a set of trajectories sampled using $P^{\text{eval}}_{\mathcal{S},\mathcal{A}}(E)$ and π. We report $J(\theta; \mathcal{B}_{\text{eval}})$ as an estimate of the expected KL divergence between the evaluated and true models under the joint distribution $P(h, E \mid \pi) \triangleq P_\pi(h \mid E) P^{\text{eval}}_{\mathcal{S},\mathcal{A}}(E)$:

$$
\begin{aligned}
J(\theta; \mathcal{B}_{\text{eval}}) &\approx \underbrace{\mathbb{E}_{E \sim P^{\text{eval}}_{\mathcal{S},\mathcal{A}}(\cdot), h \sim P_\pi(\cdot \mid E)}[\mathcal{H}(M, M_\theta \mid h, \ell_E)]}_{\text{expected conditional cross entropy between } M \text{ and } M_\theta} \\
&= \underbrace{\mathbb{E}_{E \sim P^{\text{eval}}_{\mathcal{S},\mathcal{A}}(\cdot), h \sim P_\pi(\cdot \mid E)}[\text{KL}(M \mid\mid M_\theta; h, \ell_E)]}_{\text{expected conditional KL divergence between } M \text{ and } M_\theta} + \underbrace{\mathbb{E}_{E \sim P^{\text{eval}}_{\mathcal{S},\mathcal{A}}(\cdot), h \sim P_\pi(\cdot \mid E)}[\mathcal{H}(M \mid h, \ell_E)]}_{\text{expected conditional entropy of } M \text{ (constant w.r.t. } \theta)}
\end{aligned}
$$

Note that the minimum value of this metric is *not* zero when the environment is stochastic.

F EXTENDED IMAGINARY IMITATION LEARNING RESULTS (TABLE 6)

Table 6: Average returns (↑) in real environments of policies trained with imaginary imitation learning using world models. For each world model type, we use the best checkpoint of a run chosen randomly among the five runs mentioned in Table 1. Experiments are conducted in 90 environments randomly chosen from the test splits (30 from each split). For each environment and learned policy, we compute the average return over 48 runs. For each split, we report the means of the average returns in the 30 environments with 95% t-value confidence intervals. Bold numbers indicate the best non-oracle means in the corresponding settings. `EMMA-LWM` outperforms all baselines in all settings.

Setting	World model	NewCombo (easy)	NewAttr (medium)	NewAll (hard)
Online IL *(near-optimal expert)*	`Observational`	0.75 ± 0.16	-0.41 ± 0.21	-0.21 ± 0.21
	`Standard`	0.93 ± 0.13	0.04 ± 0.26	0.30 ± 0.22
	`GPTHard`	0.82 ± 0.15	-0.20 ± 0.20	-0.06 ± 0.21
	`EMMA-LWM` (ours)	**1.01** ± **0.12**	**0.96** ± **0.17**	**0.62** ± **0.21**
	`OracleParse`	1.04 ± 0.13	0.85 ± 0.20	0.91 ± 0.18
Filtered BC *(near-optimal expert)*	`Observational`	0.77 ± 0.14	-0.42 ± 0.15	-0.30 ± 0.16
	`Standard`	1.05 ± 0.14	0.20 ± 0.27	0.17 ± 0.20
	`GPTHard`	0.79 ± 0.15	-0.10 ± 0.20	-0.07 ± 0.20
	`EMMA-LWM` (ours)	**1.18** ± **0.10**	**0.75** ± **0.20**	**0.44** ± **0.18**
	`OracleParse`	1.17 ± 0.11	0.84 ± 0.19	0.80 ± 0.18
Filtered BC *(suboptimal expert)*	`Observational`	0.71 ± 0.15	-0.35 ± 0.18	-0.33 ± 0.17
	`Standard`	0.68 ± 0.15	-0.15 ± 0.21	-0.10 ± 0.17
	`GPTHard`	0.75 ± 0.22	0.05 ± 0.25	0.06 ± 0.17
	`EMMA-LWM` (ours)	**0.98** ± **0.13**	**0.29** ± **0.25**	**0.13** ± **0.19**
	`OracleParse`	1.09 ± 0.13	0.50 ± 0.24	0.49 ± 0.18