Learning to Model the World With Language

Jessy Lin [1] **Yuqing Du** [1] **Olivia Watkins** [1] **Danijar Hafner** [1] **Pieter Abbeel** [1] **Dan Klein** [1] **Anca Dragan** [1]

Abstract

To interact with humans and act in the world, agents need to understand the range of language that people use and relate it to the visual world. While current agents can learn to execute simple language instructions, we aim to build agents that leverage diverse language—language like "this button turns on the TV" or "I put the bowls away"—that conveys general knowledge, describes the state of the world, provides interactive feedback, and more. Our key idea is that *agents should interpret such diverse language as a signal that helps them predict the future*: what they will observe, how the world will behave, and which situations will be rewarded. This perspective unifies language understanding with future prediction as a powerful self-supervised learning objective. We instantiate this in Dynalang, an agent that learns a multimodal world model to predict future text and image representations, and learns to act from imagined model rollouts. While current methods that learn language-conditioned policies degrade in performance with more diverse types of language, we show that Dynalang learns to leverage environment descriptions, game rules, and instructions to excel on tasks ranging from game-playing to navigating photorealistic home scans. Finally, we show that our method enables additional capabilities due to learning a generative model: Dynalang can be pretrained on text-only data, enabling learning from offline datasets, and generate language grounded in an environment.

1 Introduction

A long-standing goal of artificial intelligence is to develop agents that can use language to interact naturally with people in the physical world (Winograd, 1972). Current embodied agents can follow basic instructions like "bring me the apple" (Driess et al., 2023). However, the full potential of language affords much richer communication beyond task specification. Consider a household robot—beyond delegating tasks to it one after another ("clean the dishes...pick up the toy..."), one would imagine that a truly natural interaction would allow us to communicate beyond the "here and now" (Hockett & Hockett, 1960): sharing knowledge such as "the top left button turns off the TV," providing situational information such as "we're out of milk," and coordinating by saying "I already vacuumed the living room." Language communicates what we know about the state of the world or how things work, not just what to do.

It is unclear how to best integrate diverse kinds of language with vision and action in a single agent. In many ways, current methods are specialized for learning from language instructions, e.g. by embedding a task description ("pick up the blue block") at the beginning of an episode and outputting a sequence of motor controls (Brohan et al., 2023; Nair et al., 2021). To freely collaborate with humans, agents ideally would continuously integrate language inputs while acting in the environment. This natural interaction requires us to move beyond language "prompting" towards methods that input and output language continuously, along with action and video. More crucially, we hypothesize that directly mapping diverse language to optimal actions is a difficult learning problem, because language and optimal actions may only be weakly correlated if their dependency is complex. Consider the example "I put the bowls away": if the task at hand is cleaning up, the agent should respond by moving on to the next cleaning step, whereas if it is serving dinner, the agent should retrieve the bowls. The key question is thus what the right *learning signal* is for understanding the full range of language while acting in the world.

In this work, we propose that agents can ground diverse kinds of language by using it to *predict the future*. We instantiate this idea with Dynalang, an agent that learns a joint generative model of language and vision (i.e., a multimodal *world model*; Ha & Schmidhuber (2018)), and uses the model to plan and act. We build on the DreamerV3 algorithm (Hafner et al., 2023), training the world model to predict future latent representations of all observation modalities, using experience collected from acting in the environment. In contrast to directly predicting what to *do* with a language-conditioned policy, Dynalang decouples learning to model the world with language (supervised learning

[1]UC Berkeley. Website: dynalang.github.io
Correspondence to: Jessy Lin <jessy_lin@berkeley.edu>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024.

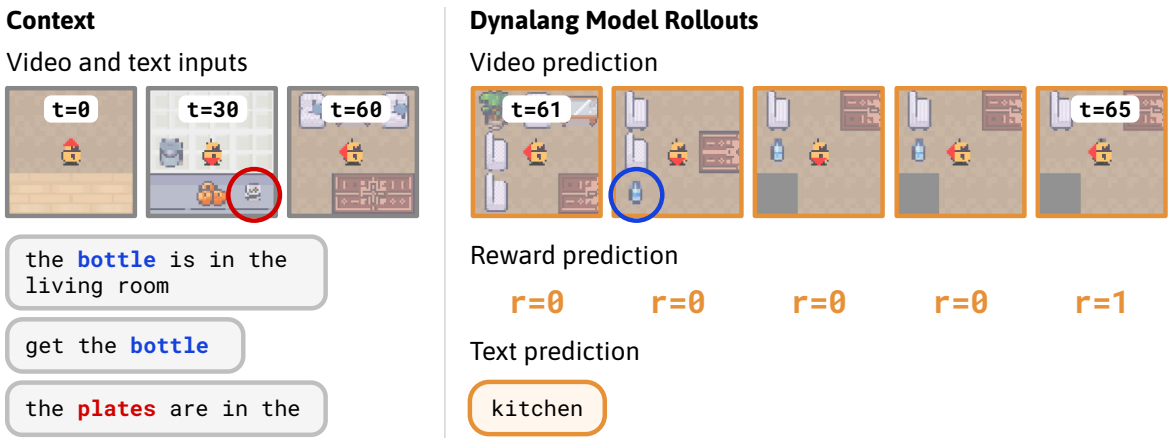


Figure 1. Dynalang learns to use language to make predictions about future (text + image) observations and rewards. Here, we show real model predictions in the HomeGrid environment. From the past text "the bottle is in the living room", the agent predicts at timesteps 61-65 that it will see the bottle in the final corner of the living room. From the text "get the bottle" describing the task, the agent predicts that it will be rewarded for picking up the bottle. The agent can also predict future text observations: given the prefix "the plates are in the" and the plates it observed on the counter at timestep 30, the model predicts the most likely next token is "kitchen."

with prediction objectives) from learning to act given that model (reinforcement learning with task rewards). Future prediction provides a rich grounding signal for learning what language utterances mean, which in turn equip the agent with a richer understanding of the world to solve complex tasks. Intuitively, the utterance "I put the bowls away" helps agents make better predictions about future observations (i.e., that if it opens the cabinet, it will observe the bowls there). Prior knowledge such as "wrenches can be used to tighten nuts" helps agents predict environment dynamics. We can also formulate instruction following predictively—instructions help agents predict how they will be rewarded.

World models are well-studied in model-based RL (Ha & Schmidhuber, 2018; Hafner et al., 2023)—our goal is not to introduce a new world modeling architecture or to demonstrate that world models can be augmented with language, but to investigate *whether learning language-conditioned world models enable agents to scale to more diverse language use*, compared to language-conditioned policies. First, we explore the design space for augmenting model-based agents with language, demonstrating that many ways of fusing language and vision in a world model underperform the simple and effective architecture in Dynalang. Then, we investigate whether Dynalang can indeed learn from diverse kinds of language, using language beyond instructions to better solve tasks, while maintaining strong performance on instruction following benchmarks. Our evaluation focuses on scaling up *language complexity* in a controlled way (as opposed to task or visual complexity) across a wide range of simulated environments. We build a home cleanup environment focused on testing agents' ability to understand diverse language types, HomeGrid, where Dynalang learns to use language hints about future observations, environment dynamics, and corrections, outperforming strong model-free language-conditioned policies whose performance *degrades* with more diverse language. On the Messenger benchmark (Hanjie et al., 2021), we show that Dynalang can read game manuals to fit the most challenging stage of the game, outperforming task-specific architectures. In vision-language navigation (Krantz et al., 2020), we show that Dynalang can also follow instructions in a visually and linguistically complex domain.

Additionally, we explore whether Dynalang enables additional scalability benefits and capabilities due to learning a generative model. While our previous experiments train agents from scratch, pretraining on offline text data is known to be an important ingredient for scaling models to open-domain language. We validate that we can pretrain Dynalang on single-modality text-only data without actions or rewards at a small scale, demonstrating that both in-domain text and a general-domain text dataset of ∼500M tokens improves downstream RL performance. Finally, we explore whether we can leverage the generative language-vision world model to generate language, enabling the agent to speak. In our framework, what the agent sees can inform future token predictions; by augmenting the action space with language and regularizing it towards the generative model, we demonstrate that Dynalang can gather information to answer questions about a simple visual environment.

Our contributions:

- We propose that predicting the future in a multimodal world model allows embodied agents to ground diverse types of language to visual experience. We implement Dynalang to instantiate this idea.
- We explore the design space for fusing language and vision in a world model and find a simple and effective architecture.

Figure 2. We consider a range of environments that feature visual inputs and diverse types of language. We introduce HomeGrid, a challenging visual gridworld with instructions and diverse hints. Messenger is a benchmark with symbolic inputs and hundreds of human-written game manuals that require multi-hop reasoning. Habitat simulates photorealistic 3D homes for vision-language navigation in hundreds of scenes and crowdsourced language. LangRoom is a simple visual grid world with partial observability, where the agent needs to produce both motor actions and language.

- We find that the performance of language-conditioned policies degrades when faced with more diverse language, while Dynalang learns to utilize many kinds of language to excel on a broad range of tasks.
- We show that learning a generative model of language and vision enables additional capabilities, such as embodied language generation and text-only pretraining without actions or rewards.

2 Related Work

Much work has focused on teaching reinforcement learning agents to utilize language to solve tasks by directly conditioning policies on language (Lynch & Sermanet, 2021; Shridhar et al., 2022; Abramson et al., 2020). More similar to our work, recent work proposes text-conditioning a video model trained on expert demonstrations and using the model for planning (Du et al., 2023b; Yang et al., 2023). However, language in these settings has thus far been limited to short instructions, and only a few works investigate how RL agents can learn from other kinds of language like descriptions of how the world works, in simple settings (Zhong et al., 2020; Hanjie et al., 2021). In reality, human language is far richer than imperative commands. Other work looks to diversify the utterances that agents can understand with supervised learning on human-annotated or expert datasets in embodied environments (Das et al., 2018; Thomason et al., 2019; Bara et al., 2021) or pretrained large language models (LLMs) (Huang et al., 2022a; Li et al., 2022; Ahn et al., 2022; Driess et al., 2023). These works approach the realism of natural language in the diversity of utterances and *roles* of language in the world they consider. However, supervised approaches rely on expensive human data (often with aligned language annotations and expert demonstrations), and both these approaches have limited ability to improve their behaviors and language understanding online.

In contrast to previous RL agents, Dynalang takes a step towards more diverse language understanding by making predictions in a world model, providing a learning signal that is both rich (unlike rewards), self-supervised (unlike demonstrations), and compatible with offline pretraining. We refer to Appendix C for a more detailed discussion of related work.

3 Dynalang

Dynalang utilizes diverse types of language in visual environments by encoding multiple modalities into learned representations and then predicting the sequence of future representations given actions. For our algorithm, we build on the model-based algorithm DreamerV3 (Hafner et al., 2023) and extend it to process and optionally produce language. The world model is trained from a replay buffer of past experience while the agent is interacting with the environment. It can additionally be pretrained from text-only data. To select actions, we train an actor-critic model from sequences of representations imagined by the world model. The algorithm is summarized in Algorithm 1.

Problem setting To perform interactive tasks, an agent chooses actions a_t to interact with an environment that responds with rewards r_t, a flag for whether the episode continues c_t, and observations o_t. In this work, we consider

Algorithm 1: Dynalang

define rewards r_t, episode continue flag c_t, images x_t, language tokens l_t, actions a_t, model state (h_t, z_t).
while *acting* **do**
  Step environment $r_t, c_t, x_t, l_t \leftarrow \text{env}(a_{t-1})$.
  Encode observations $z_t \sim \text{enc}(x_t, l_t, h_t)$.
  Execute action $a_t \sim \pi(a_t \mid h_t, z_t)$.
  Add transition $(r_t, c_t, x_t, l_t, a_t)$ to replay buffer.
while *training* **do**
  Draw batch $\{(r_t, c_t, x_t, l_t, a_t)\}$ from replay buffer.
  Use world model to compute multimodal representations z_t, future predictions $\hat{z}_{t+1}$, and decode $\hat{x}_t$, $\hat{l}_t$, $\hat{r}_t$, $\hat{c}_t$.
  Update world model to minimize $\mathcal{L}_{\text{pred}} + \mathcal{L}_{\text{repr}}$.
  Imagine rollouts from all z_t using π.
  Update actor to minimize $\mathcal{L}_\pi$.
  Update critic to minimize $\mathcal{L}_V$.
while *text pretraining* **do**
  Sample text batch $\{l_t\}$ from dataset.
  Create zero images x_t and actions a_t.
  Use world model to compute representations z_t, future predictions $\hat{z}_{t+1}$, and decode $\hat{l}_t$.
  Update world model to minimize $\mathcal{L}_{\text{pred}} + \mathcal{L}_l$.

environments with multimodal observations $o_t = (x_t, l_t)$, containing both visual x_t and textual input l_t at each time step. The agent's goal is to choose actions that maximize the expected discounted sum of rewards $\mathrm{E}\left[\sum_{t=1}^{T} \gamma^t r_t\right]$, where $\gamma < 1$ is a discount factor, T is the episode length, and $c_T = 0$ signals the episode end. In most of our experiments, the actions a_t are integers in a categorical action space. However, we also consider factorized action spaces where the agent outputs both a discrete movement command and a language token at each time step.

Multimodal alignment While previous work assumes that language such as instructions arrive at the beginning of an episode, we consider a diverse range of environments, summarized in Figure 2, where agents receive a continuous stream of video and text observations. For humans, reading, listening, and speaking extends over time, during which we receive new visual inputs and can perform motor actions. Analogously, at each time step we provide Dynalang with one video frame x_t and one language token l_t as input (with zero or padding if no video frame or token is available at that time step), and the agent outputs one motor action (and in applicable environments, a language token). While a natural question is whether the language token l_t needs to be semantically aligned with the video frame x_t, Dynalang does not assume or require that modalities are aligned. Intuitively, the prediction problem at each time step is to predict the future given past inputs from all modalities, and it does not matter how modalities were aligned in the past as long as information from each modality is represented in the model state. We can thus use a simple architecture that does not require more complex temporal segmentation, while outperforming other ways of fusing modalities (Section 4.1) and enabling language model-style pretraining (Section 4.6).

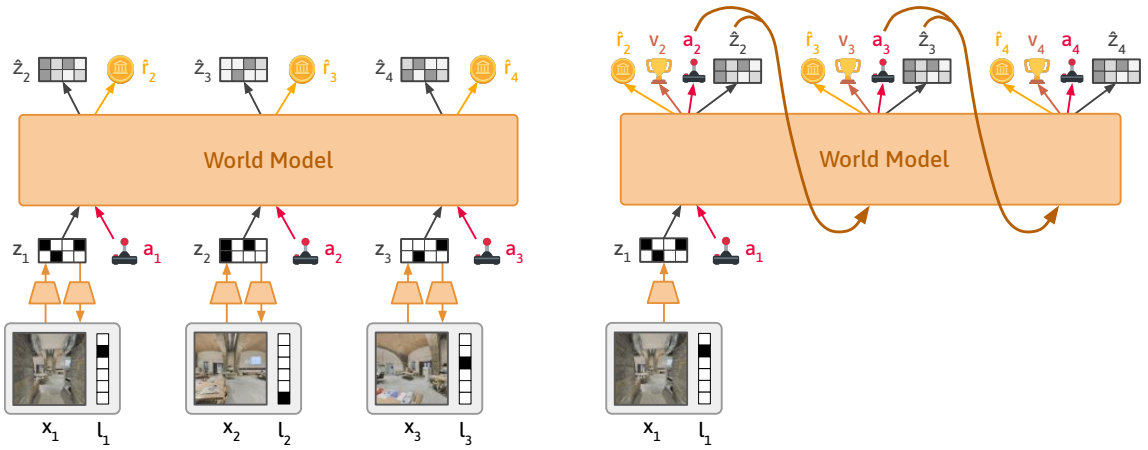
3.1 World Model Learning

The world model learns representations of all sensory modalities that the agent receives and then predicts the sequence of these latent representations given actions. Predicting future representations not only provides a rich learning signal to ground language in visual experience but also allows planning and policy optimization from imagined sequences. The world model is shown in Figure 3a. At each time step, it receives an image x_t, a language token l_t, and an action a_t. The image and language observations are compressed into a discrete representation z_t and fed together with the action into the sequence model to predict the next representation $\hat{z}_{t+1}$. The multimodal world model consists of the following components, where h_t is a recurrent state:

Sequence model:	$\hat{z}_t,\, h_t = \text{seq}(z_{t-1},\, h_{t-1},\, a_{t-1})$
Multimodal encoder:	$z_t \sim \text{enc}(x_t,\, l_t,\, h_t)$
Multimodal decoder:	$\hat{x}_t,\, \hat{l}_t,\, \hat{r}_t,\, \hat{c}_t = \text{dec}(z_t,\, h_t)$

We implement the world model as a Recurrent State Space Model (RSSM Hafner et al., 2018), where the sequence model is implemented as GRU (Cho et al., 2014) with recurrent state h_t, but other sequence models such as Transformers can also be used as the backbone (Robine et al., 2023). The decoder is trained to reconstruct observations and other information, thus shaping the model representations. The world model is trained jointly to minimize a representation learning loss $\mathcal{L}_{\text{repr}}$ and a future prediction loss $\mathcal{L}_{\text{pred}}$, which we describe below.

Multimodal representations The world model learns to compress inputs images x_t and language tokens l_t into stochastic latent representations z_t through a variational autoencoding objective (Kingma & Welling, 2013; Rezende et al., 2014). Reconstructing the input observations encourages the model to compress information from all modalities into its representations. We also predict the reward, $\hat{r}_t$, and whether the episode continues, $\hat{c}_t$, so that the policy can be learned directly on top of the latent representations, as discussed in the next section. Finally, the representations are regularized towards the predicted distribution over $\hat{z}_t$ as a prior, essentially regularizing the representations to be predictable. We denote the categorical cross entropy loss as catxent, the binary cross entropy loss as binxent, the stop gradient operator as sg, and $\beta_{\text{reg}} = 0.1$ is a hyperparameter.



(a) World Model Learning (b) Learning to Act by Latent Imagination

Figure 3. During world model learning, the model compresses observations of image frames and text to a latent representation. The model is trained to predict the next representation and reconstruct observations from the representation. During policy learning, imagined rollouts are sampled from the world model and the policy is trained to maximize imagined rewards.

The representation learning loss $\mathcal{L}_{\text{repr}}$ is thus the sum of:

$$\begin{aligned}
&\text{Image loss:} && \mathcal{L}_x = \|\hat{x}_t - x_t\|_2^2 \\
&\text{Language loss:} && \mathcal{L}_l = \text{catxent}(\hat{l}_t, l_t) \\
&\text{Reward loss:} && \mathcal{L}_r = \text{catxent}(\hat{r}_t, \text{twohot}(r_t)) \\
&\text{Continue loss:} && \mathcal{L}_c = \text{binxent}(\hat{c}_t, c_t) \\
&\text{Regularizer:} && \mathcal{L}_{\text{reg}} = \beta_{\text{reg}} \max(1, \text{KL}[z_t \parallel \text{sg}(\hat{z}_t)])
\end{aligned}$$

We choose a strided CNN image encoder, a strided CNN as image decoder, and MLPs for all other model components. We evaluate our method both with one-hot token observations (i.e., learning the embeddings from scratch) and pretrained embeddings from T5 (Raffel et al., 2020). One-hot representations are reconstructed with the cross entropy loss above and pretrained embeddings are reconstructed with a squared error. For more details on world model learning, refer to Appendix A.

Future prediction The world model learns to predict the sequence of multimodal representations, which enables it to plan and ground language. The sequence model produces $\hat{z}_t$ from the current model state (z_{t-1}, h_{t-1}) and the current action a_{t-1}, which is trained to match the actual representation at the next timestep z_t. Concretely, the future prediction objective is:

$$\text{Prediction loss:} \quad \mathcal{L}_{\text{pred}} = \beta_{\text{pred}} \max(1, \text{KL}[\text{sg}(z_t) \parallel \hat{z}_t])$$

where the gradient around the target distribution for z_t is stopped since it is also a learned representation and $\beta_{\text{pred}} = 0.5$ is a hyperparameter. Intuitively, the codes z_t contains both information from current observation and additional information that may be required to predict the reward and episode continuation. By training the world model to make predictions $\hat{z}_t$ of its future representations, it effectively learns to predict future images, language, and rewards, encouraging the agent to extract information from language and learn the correlations between its multiple modalities. For example, when the language input describes that "the book is in the bedroom" and the agent later on visually observes the book, the agent will learn this multimodal association even if the reward signal does not directly relate the two.The world model is trained to optimize the overall loss $\mathcal{L}_{\text{repr}} + \mathcal{L}_{\text{pred}}$ with respect to all its parameters.

Single-Modality Pretraining One potential benefit of separating world modeling from policy learning is that the world model can be trained offline, benefitting from large-scale text-only and video-only datasets without actions. To pretrain the world model with text-only data as in Section 4.6, we zero out the image and action inputs and set the image, reward, and continuation decoder loss coefficients to 0 so the pretraining focuses on learning to represent text and text dynamics (i.e. language modeling). Dynalang can then be finetuned on experience with all modalities (language, images, and actions) by initializing the actor and critic from scratch, while continuing to train the world model. Note that unlike the typical language modeling objective, the model is not explicitly trained to predict the next token from the prefix, except through the next-*representation* prediction.

3.2 Policy Learning

To select actions, we train an actor critic algorithm (Williams, 1992) purely from imagined sequences of multimodal representations (Sutton, 1991), as shown in Figure 3b. The critic estimates the discounted sum of future rewards for each state to guide actor learning. Both networks are MLPs:

$$\text{Actor net:} \quad \pi(a_t | h_t, z_t) \qquad \text{Critic net:} \quad \text{V}(h_t, z_t)$$

We do not modify the policy learning algorithm of DreamerV3 and refer to Appendix B for details.

4 Experiments

Our experiments test the following hypotheses:

H1) Aligning image and language as single (image, token) pairs per timestep outperforms other methods for incorporating language into DreamerV3 (Section 4.1).

H2) Dynalang can better utilize diverse types of language to improve task performance over language-conditioned policies. To test this, we investigate whether Dynalang performance improves when provided with different kinds of language hints in HomeGrid (Section 4.2) and game manuals in Messenger (Section 4.3), compared to model-free RL baselines.

H3) Incorporating instructions into a world model is no worse than directly learning a language-conditioned policy. To test this, we compare performance to baselines with task-only language in HomeGrid and on vision-language navigation (Section 4.4).

H4) The multimodal generative model enables Dynalang to handle tasks that require grounded language generation (Section 4.5) and pretraining on offline text-only data (Section 4.6).

Language encodings We tokenize all text with the T5 tokenizer (Raffel et al., 2020), with a vocabulary size of 32,100. In HomeGrid we use one-hot token encodings. In Messenger and VLN-CE, where agents must generalize to synonyms and linguistic variations, we embed each sentence with T5-small (60M parameters) and use the last hidden layer representation for each token.

Baselines We compare against two off-policy model-free RL baselines: IMPALA (Espeholt et al., 2018) and R2D2 (Kapturowski et al., 2019). The architecture for both algorithms consists of an LSTM that takes in input embeddings from a CNN image encoder and an MLP language encoder. We use the implementations from the SeedRL repository (Espeholt et al., 2019). We pass the same language observations to the baselines as to our method (token embeddings or one-hot encodings). We also try providing the baselines with sentence embeddings from a pretrained `all-distilroberta-v1` model from the Sentence Transformers library (Reimers & Gurevych, 2019) and did not find a consistent improvement across our tasks. Both baseline models are ∼10M parameters, and we did not find that these models benefit from scaling parameter count.

4.1 Aligning Language, Vision, and Action in a World Model

First, we isolate the effect of our design choices for the multimodal architecture in Dynalang from the base DreamerV3 architecture by comparing to other ways of conditioning DreamerV3 on language. We use Messenger's Stage 1 as a testbed, where the agent must learn to read a text manual in order to achieve high reward (see Section 4.3 for details).

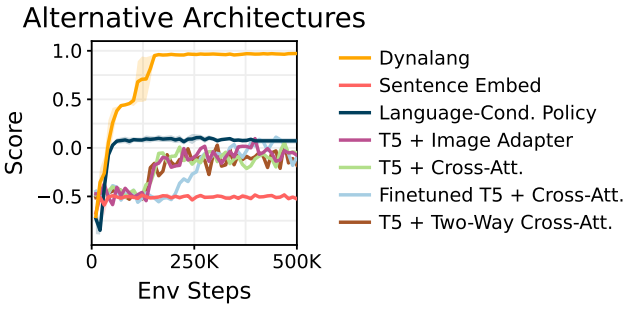


Figure 4. Comparison of ways to equip the world model with language inputs on Messenger S1. We compare ways of conditioning DreamerV3 on language and find that Dynalang substantially outperforms other model-based approaches, even those based on pretrained T5, despite training from scratch.

We implement several common language conditioning methods in DreamerV3 from previous work, comparing their performance to Dynalang in Figure 4:

1. **Language-Conditioned Policy** Embed tokens with a GRU and condition the policy on the final hidden state. This baseline is similar to how model-free approaches implement language conditioning (Shridhar et al., 2022), but the agent still learns to model images with a world model. This ablates the effect of learning joint representations of language and images in the world model.
2. **Sentence Embed** Input text into the world model one sentence at a time, using SentenceBERT to embed each sentence of the manual. This ablates the effect of inputting one token per timestep.
3. **T5 with Image Adapter** Train the image encoder to map image features to the embedding space of a pretrained T5 model, following multimodal models such as LLaVA (Liu et al., 2023). This allows the agent to leverage the pretrained representations of the LLM. We map the CNN output at each timestep to 10 language tokens with an MLP, attend over the image tokens and the tokens of the language input with a frozen T5 encoder, and condition the world model on the last hidden state of T5.
4. **T5 with Cross-Attention** Embed the entire manual with T5 and then fuse with the image observation by attending to the sequence of token embeddings with the image embedding output by the image encoder, similar to the original Messenger baseline (Hanjie et al., 2021).
5. **Finetuned T5 with Cross-Att.** (3.), but finetune T5.
6. **T5 with Two-Way Cross-Att.** (3.), but additionally map images to a fixed number of latents and attend to it with the pooled text embedding, following multimodal models such as Lu et al. (2016).

We find that Dynalang outperforms these alternatives even with token embeddings initialized from scratch while also being simple and efficient to train, supporting **H1**.

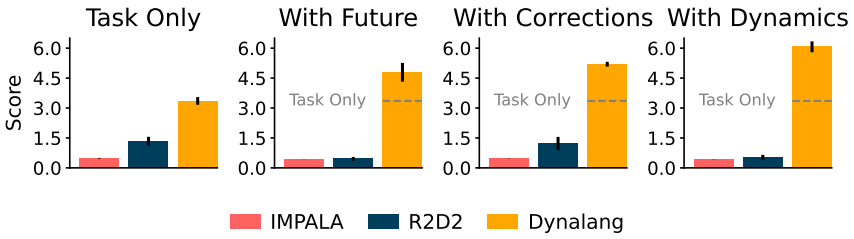


Figure 5. HomeGrid performance after 50M steps (2 seeds). Dynalang learns to use all types of language hints to score higher than when just provided with the task information, outperforming language-conditioned IMPALA and R2D2, where we see performance decrease with language hints.

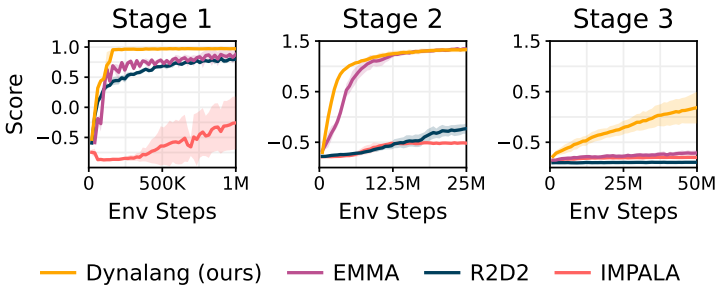


Figure 6. Messenger training performance (2 seeds). Dynalang outperforms language-conditioned IMPALA and R2D2, as well as the task-specific EMMA architecture, fitting the most complex stage of the game where other methods fail to achieve non-trivial performance.

4.2 HomeGrid: Language Hints

As most standard RL benchmarks do not provide language beyond instructions, we introduce a new environment, HomeGrid, that evaluates how well agents can ground diverse types of language to solve tasks. HomeGrid is a multitask gridworld where agents receive language task specifications and *language hints*. Hints provide prior knowledge about world dynamics, information about world state, or corrections that assist the agent. The agent can otherwise acquire the same information through its own interaction with the environment, as in standard RL. Agents can achieve higher performance if they learn to ground language.

There are five task types involving objects and trash bins (find, get, clean up, rearrange, open) for a total of 38 tasks. Agents get pixel observations with a partially observed view of the environment and can move and interact with objects and trash bins. Object locations, bin locations, and bin dynamics (*i.e.,* which action correctly opens the bin) are randomized on each episode. Objects are also randomly moved throughout the episode. Agents receive task specifications in language. When a task is completed, the agent gets a reward of 1 and a new task is sampled. To achieve a high score, agents must complete as many tasks as possible before the episode terminates in 100 steps. We also provide hints at random points throughout the episode that are provided token-by-token while the agent continues to act. We script the following language hints, with examples shown in Figure D.1 in the appendix:

- **Future observations** Descriptions of where objects are in the world. Without language, the agent must explore the environment to find objects.

- **Dynamics** Descriptions of the correct action to open each trash bin. Without language, the agent can try all the different actions, although taking the wrong action can disable the trash can for a variable number of timesteps.

- **Corrections** Tell the agent "no, turn around" when its distance to the current goal object is increasing. Without language, the agent must explore on its own.

Section 4.1 shows that Dynalang benefits from all kinds of language, achieving higher scores with hints relative to just using instructions. Notably, agents *never receive direct supervision* about what the hints mean in HomeGrid, and hints are often far removed from the objects or observations they refer to. We show the agent's imagined rollouts in Appendix F, which provide qualitative evidence that Dynalang learns to ground language to the environment purely via the future prediction objective. IMPALA struggles to learn the task at all, while R2D2 learns to use the types of language that are correlated with reward (tasks and corrections). Interestingly, we find that while R2D2's performance drops as it receives more diverse language, while Dynalang improves with more language information, supporting **H2** and our initial motivation that learning actions from diverse language is a difficult learning problem.

Dynalang also outperforms baselines even with task-only language, supporting **H3**. Language-conditioned policies in prior work often rely on supervised learning on expensive language-action datasets; while Dynalang may also benefit from offline data to further boost performance, it can achieve non-trivial performance even when learning from scratch.

4.3 Messenger: Game Manuals

Next, we evaluate Dynalang on the Messenger game environment (Hanjie et al., 2021), which tests whether agents can read text manuals describing game dynamics to achieve high scores. In Messenger, the agent must retrieve a message from one of the entities in the environment and deliver it to another entity, while avoiding enemies. In each episode, the agent receives a manual describing the randomized entity

Figure 7. VLN-CE results. **(left)** A portion of a trained agent trajectory, given the instruction "Exit the bedroom, go straight down the hallway, make a right into the doorway of the bathroom and stop". **(right)** Success rate during RL training, averaged across 3 seeds for Dynalang and 2 seeds for R2D2.

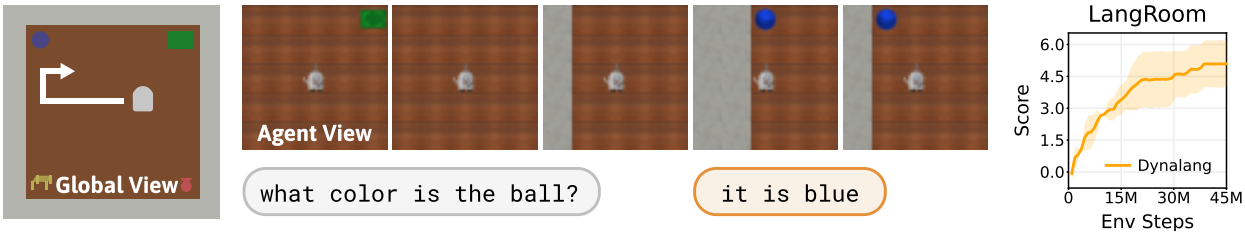


Figure 8. LangRoom results. **(left)** A real trajectory from a trained agent. The agent learns to take information-gathering actions from reward. When asked "what color is the ball?" the agent walks to the corner with the ball and generates the tokens "it is blue." **(right)** Training curve. The agent learns to answer more questions accurately.

roles and movement dynamics as a series of rules, requiring multi-hop reasoning over both visual and text inputs (e.g. combining the manual information that the goal entity is a "fleeing wizard" with observations of entity identities and movement dynamics). Messenger has three stages of increasing length and difficulty.

In addition to the baselines above, we compare Dynalang to EMMA, the original baseline for the benchmark that uses a specialized grid-based architecture and learns a language-conditioned policy with PPO (Schulman et al., 2017). As seen in Figure 6, Dynalang achieves higher performance and learns more efficiently than EMMA, IMPALA and R2D2. While other methods fail to fit S3 at all, our method learns to interpret the manuals to achieve non-trivial performance on the most challenging stage, further supporting **H2**.

4.4 Vision-Language Navigation: Instruction Following

To evaluate how Dynalang performs in more complex environments, we apply it to the popular Vision-Language Navigation (VLN) (Anderson et al., 2018) benchmark. Agents must navigate Matterport3D panoramas captured in real homes (Chang et al., 2017), following crowd-sourced natural language instructions that indicate where the agent should navigate to, such as "Go past the bed to the door. Enter the hallway,..." We focus on the more challenging variant, VLN in Continuous Environments (VLN-CE) (Krantz et al., 2020), in which agents take low-level discrete actions (left, forward, ...) rather than relying on a waypoint navigation graph as in the original VLN task. In this task, our goal is to demonstrate that Dynalang can learn to follow language instructions (e.g. via predicting future rewards).

Each episode randomly samples a language instruction and corresponding scene from the training dataset out of 10,819 unique instructions total. The agent gets a dense reward based on relative position to the current goal, a success reward when taking the `stop` action at the correct location, and a penalty otherwise.

As shown in Figure 7, Dynalang succeeds at significantly more instructions compared to the model-free R2D2 baseline, supporting **H3**. While Dynalang successfully learns to ground instructions from scratch, performance is not yet competitive with state-of-the-art VLN methods (many of which use expert demonstrations or specialized architectures), and further work is needed to close the gap.

4.5 LangRoom: Embodied Question Answering

Next, show how Dynalang can also *generate language* in the same framework. On the other benchmarks, language is used to inform agents' future predictions about the world, but perception can also inform future predictions about *what might be said*. For example, agents could predict that they will hear descriptive utterances such as "the stove is on" that are consistent with its own observations of the burner producing flames.

We introduce the LangRoom embodied question answering environment to demonstrate a proof-of-concept of this capability. We expand the action space to include language

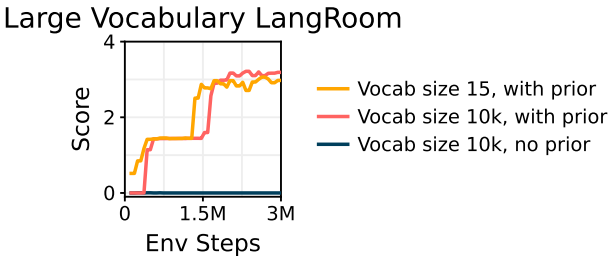


Figure 9. We scale Dynalang to a vocabulary size of 10000 on LangRoom by regularizing the language action towards the 1-step world model predictions. We match QA performance of an agent with a vocab size of 15, while naively scaling up the vocabulary size to 10000 with no prior fails to learn.

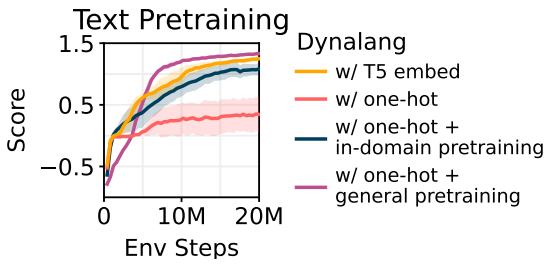


Figure 10. One-hot token encodings underperform pretrained embeddings on S2, but pretraining Dynalang with a small amount of text-only data closes much of the gap.

by allowing the agent to output one language token per timestep as an action. The environment contains a room with objects with fixed positions but randomized colors. The language observations from the environment are *questions* `"what color is the <object>?"`. The agent must move to the object and emit a language action saying the correct color. Results are shown in Figure 8. The agent learns to answer more questions correctly with task reward by taking information gathering actions to observe the color of the object and generating text consistent with the world state.

Critically, learning an *generative model* of language is important for scaling up to larger vocabularies, as seen in Figure 9, supporting **H4**. Naively increasing the vocabulary size to 10k (by adding dummy tokens) fails to learn, likely because selecting actions from such a large action space with RL is intractable. We recover performance by using the world model to guide language generation, regularizing the language action towards the predicted next token in the world model by adding $\mathrm{KL}\big[\pi(l_t^{\text{act}} \mid h_t, z_t) \;\big\|\; \mathrm{sg}(p(\hat{l}_{t+1} \mid \hat{h}_{t+1}, \hat{z}_{t+1}))\big]$ to the entropy regularizer. Intuitively, this regularizes the generated language actions towards valid and likely utterances in the current context, similar to how LLMs are trained with RL (Ziegler et al., 2019).

4.6 Text-only Pretraining

Our experiments thus far have investigated how Dynalang can learn from multimodal experience *online*, demonstrating that the world modeling objective enables the agent to understand language in the context of vision and action even with no prior or pretrained initialization. However, we expect that large-scale offline training will be necessary to scale to realistic settings. Thus, in this section, we investigate whether the generative model in Dynalang can be pretrained on single-modality data to benefit downstream learning.

To evaluate this capability, we zero out the other modality and action inputs and pretrain Dynalang from scratch on text-only corpora: (1) **in-domain text** with manuals from Messenger S2 games (2) **domain-general text** with TinyStories (Eldan & Li, 2023), a dataset of 2M short stories (∼500M tokens) generated by GPT-4. We evaluate on Messenger S2, where models that learn to embed one-hot token observations from scratch struggle to learn the complex language in S2 without pretraining on S1. We compare S2 task performance with learned embeddings to using pretrained T5 embeddings, training all methods from scratch on S2 without initializing from S1. Results are shown in Figure 10. Dynalang is able to benefit from offline pretraining, supporting **H4**. Even a small amount of in-domain text closes much of the gap between training text embeddings from scratch and using T5 embeddings. Furthermore, pretraining on TinyStories *exceeds* the final performance of using T5 embeddings, likely because pretraining allows the model to learn text dynamics offline rather than during environment interaction. Although the model is not trained explicitly to do language modeling except through next-representation prediction, we show generated language samples in Appendix E. These results suggest that our paradigm can leverage the benefits of large-scale offline pretraining, providing a way to unify offline and online data with language, vision, and action in a single agent.

5 Conclusion

Taken together, our results suggest a way forward for multimodal agents that can understand language and vision and act in the world. We presented Dynalang, an agent that does so through future prediction as a rich self-supervised objective. In contrast to model-free methods that struggle with increased language perplexity, we demonstrated in four diverse environments how Dynalang can understand various types of language and what they mean in the world, even when learning from scratch from its own experiences. Additionally, we show the same world model architecture can be pretrained on offline text data in a way that benefits downstream learning from experience, paving the way towards a self-improving multimodal agent that interacts with humans in the world.

Acknowledgments

We thank Aditi Raghunathan, Michael Chang, Sanjay Subramanian, and Nicholas Tomlin for helpful discussions, and Kuba Grudzien, Jacob Steinhardt, Meena Jagadeesan, and Alex Wei for draft feedback. We thank the TPU Research Cloud (TRC) program and Danat Pomeranets from Google for access to compute resources. This work was funded in part by The Berkeley Center for Human Compatible AI (CHAI) and the Office of Naval Research (ONR-YIP).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Abramson, J., Ahuja, A., Barr, I., Brussee, A., Carnevale, F., Cassin, M., Chhaparia, R., Clark, S., Damoc, B., Dudzik, A., et al. Imitating interactive intelligence. *arXiv preprint arXiv:2012.05672*, 2020.

Ahn, M., Brohan, A., Brown, N., Chebotar, Y., Cortes, O., David, B., Finn, C., Gopalakrishnan, K., Hausman, K., Herzog, A., Ho, D., Hsu, J., Ibarz, J., Ichter, B., Irpan, A., Jang, E., Ruano, R. J., Jeffrey, K., Jesmonth, S., Joshi, N., Julian, R., Kalashnikov, D., Kuang, Y., Lee, K.-H., Levine, S., Lu, Y., Luu, L., Parada, C., Pastor, P., Quiambao, J., Rao, K., Rettinghouse, J., Reyes, D., Sermanet, P., Sievers, N., Tan, C., Toshev, A., Vanhoucke, V., Xia, F., Xiao, T., Xu, P., Xu, S., and Yan, M. Do as I can and not as I say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

Alayrac, J.-B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al. Flamingo: A visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

Ammanabrolu, P. and Riedl, M. O. Playing text-adventure games with graph-based deep reinforcement learning. *arXiv preprint arXiv:1812.01628*, 2018.

An, D., Wang, H., Wang, W., Wang, Z., Huang, Y., He, K., and Wang, L. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *arXiv preprint arXiv:2304.03047*, 2023.

Anderson, P., Wu, Q., Teney, D., Bruce, J., Johnson, M., Sünderhauf, N., Reid, I. D., Gould, S., and van den Hengel, A. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pp. 3674–3683. IEEE Computer Society, 2018. doi: 10.1109/CVPR.2018.00387. URL http://openaccess.thecvf.com/content_cvpr_2018/html/Anderson_Vision-and-Language_Navigation_Interpreting_CVPR_2018_paper.html.

Andreas, J. and Klein, D. Alignment-based compositional semantics for instruction following. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1165–1174, Lisbon, Portugal, 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1138. URL https://www.aclweb.org/anthology/D15-1138.

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In Precup, D. and Teh, Y. W. (eds.), *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pp. 166–175. PMLR, 2017. URL http://proceedings.mlr.press/v70/andreas17a.html.

Bara, C.-P., CH-Wang, S., and Chai, J. MindCraft: Theory of mind modeling for situated dialogue in collaborative tasks. In Moens, M.-F., Huang, X., Specia, L., and Yih, S. W.-t. (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 1112–1125, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.85. URL https://aclanthology.org/2021.emnlp-main.85.

Bengio, Y., Léonard, N., and Courville, A. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

Bisk, Y., Holtzman, A., Thomason, J., Andreas, J., Bengio, Y., Chai, J., Lapata, M., Lazaridou, A., May, J., Nisnevich, A., Pinto, N., and Turian, J. Experience grounds language. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 8718–8735, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.703. URL https://aclanthology.org/2020.emnlp-main.703.

Branavan, S., Zettlemoyer, L., and Barzilay, R. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 1268–1277, Uppsala, Sweden, 2010.

Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P10-1129.

Brohan, A., Brown, N., Carbajal, J., Chebotar, Y., Chen, X., Choromanski, K., Ding, T., Driess, D., Dubey, A., Finn, C., et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

Carta, T., Romac, C., Wolf, T., Lamprier, S., Sigaud, O., and Oudeyer, P.-Y. Grounding large language models in interactive environments with online reinforcement learning. In Krause, A., Brunskill, E., Cho, K., Engelhardt, B., Sabato, S., and Scarlett, J. (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 3676–3713. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/carta23a.html.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niessner, M., Savva, M., Song, S., Zeng, A., and Zhang, Y. Matterport3d: Learning from rgb-d data in indoor environments. *arXiv preprint arXiv:1709.06158*, 2017.

Chen, J., Guo, H., Yi, K., Li, B., and Elhoseiny, M. Visualgpt: Data-efficient adaptation of pretrained language models for image captioning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 18030–18040, 2022.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Dagan, G., Keller, F., and Lascarides, A. Learning the effects of physical actions in a multi-modal environment. In *Findings of the Association for Computational Linguistics: EACL 2023*, pp. 133–148, Dubrovnik, Croatia, May 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.findings-eacl.10.

Das, A., Datta, S., Gkioxari, G., Lee, S., Parikh, D., and Batra, D. Embodied Question Answering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Dasgupta, I., Kaeser-Chen, C., Marino, K., Ahuja, A., Babayan, S., Hill, F., and Fergus, R. Collaborating with language models for embodied reasoning. *arXiv preprint arXiv:2302.00763*, 2023.

Driess, D., Xia, F., Sajjadi, M. S., Lynch, C., Chowdhery, A., Ichter, B., Wahid, A., Tompson, J., Vuong, Q., Yu, T., et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

Du, Y., Watkins, O., Wang, Z., Colas, C., Darrell, T., Abbeel, P., Gupta, A., and Andreas, J. Guiding pretraining in reinforcement learning with large language models, 2023a.

Du, Y., Yang, M., Dai, B., Dai, H., Nachum, O., Tenenbaum, J. B., Schuurmans, D., and Abbeel, P. Learning universal policies via text-guided video generation. *arXiv e-prints*, pp. arXiv–2302, 2023b.

Eisenstein, J., Clarke, J., Goldwasser, D., and Roth, D. Reading to learn: Constructing features from semantic abstracts. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pp. 958–967, Singapore, 2009. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D09-1100.

Eldan, R. and Li, Y. Tinystories: How small can language models be and still speak coherent english?, 2023.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pp. 1407–1416. PMLR, 2018.

Espeholt, L., Marinier, R., Stanczyk, P., Wang, K., and Michalski, M. Seed rl: Scalable and efficient deep-rl with accelerated central inference. *arXiv preprint arXiv:1910.06591*, 2019.

Guo, J., Li, J., Li, D., Tiong, A. M. H., Li, B., Tao, D., and Hoi, S. From images to textual prompts: Zero-shot visual question answering with frozen large language models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10867–10877, 2023.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems 31*, pp. 2451–2463. 2018.

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., and Davidson, J. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

Hafner, D., Lillicrap, T., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

Hanjie, A. W., Zhong, V., and Narasimhan, K. Grounding language to entities and dynamics for generalization in reinforcement learning. In Meila, M. and Zhang, T. (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021,18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 4051–4062. PMLR, 2021. URL http://proceedings.mlr.press/v139/hanjie21a.html.

Hockett, C. F. and Hockett, C. D. The origin of speech. *Sci. Am.*, 203(3):88–97, 1960.

Huang, W., Abbeel, P., Pathak, D., and Mordatch, I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*. PMLR, 2022a.

Huang, W., Xia, F., Xiao, T., Chan, H., Liang, J., Florence, P., Zeng, A., Tompson, J., Mordatch, I., Chebotar, Y., Sermanet, P., Brown, N., Jackson, T., Luu, L., Levine, S., Hausman, K., and Ichter, B. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022b.

Jiang, Y., Gu, S. S., Murphy, K. P., and Finn, C. Language as an abstraction for hierarchical deep reinforcement learning. *Advances in Neural Information Processing Systems*, 32, 2019.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. Vima: General robot manipulation with multimodal prompts. *arXiv preprint arXiv:2210.03094*, 2022.

Kapturowski, S., Ostrovski, G., Quan, J., Munos, R., and Dabney, W. Recurrent experience replay in distributed reinforcement learning. In *International conference on learning representations*, 2019.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems*, 29, 2016.

Krantz, J., Wijmans, E., Majumdar, A., Batra, D., and Lee, S. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*, pp. 104–120. Springer, 2020.

Krantz, J., Gokaslan, A., Batra, D., Lee, S., and Maksymets, O. Waypoint models for instruction-guided navigation in continuous environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15162–15171, 2021.

Ku, A., Anderson, P., Patel, R., Ie, E., and Baldridge, J. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4392–4412, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.356. URL https://aclanthology.org/2020.emnlp-main.356.

Li, B. Z., Nye, M., and Andreas, J. Implicit representations of meaning in neural language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1813–1827, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.143. URL https://aclanthology.org/2021.acl-long.143.

Li, B. Z., Chen, W., Sharma, P., and Andreas, J. Lampp: Language models as probabilistic priors for perception and action. *arXiv e-prints*, 2023a.

Li, J., Li, D., Savarese, S., and Hoi, S. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023b.

Li, K., Hopkins, A. K., Bau, D., Viégas, F., Pfister, H., and Wattenberg, M. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *The Eleventh International Conference on Learning Representations*, 2023c. URL https://openreview.net/forum?id=DeG07_TcZvT.

Li, S., Puig, X., Du, Y., Wang, C., Akyurek, E., Torralba, A., Andreas, J., and Mordatch, I. Pre-trained language models for interactive decision-making. *arXiv preprint arXiv:2202.01771*, 2022.

Liu, H., Li, C., Wu, Q., and Lee, Y. J. Visual instruction tuning. In *NeurIPS*, 2023.

Lu, J., Yang, J., Batra, D., and Parikh, D. Hierarchical question-image co-attention for visual question answering. In Lee, D., Sugiyama, M., Luxburg, U., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016. URL https://proceedings.neurips.cc/paper_files/paper/2016/file/9dcb88e0137649590b755372b040afad-Paper.pdf.

Lu, J., Clark, C., Zellers, R., Mottaghi, R., and Kembhavi, A. Unified-io: A unified model for vision, language, and multi-modal tasks. *arXiv preprint arXiv:2206.08916*, 2022.

Luketina, J., Nardelli, N., Farquhar, G., Foerster, J. N., Andreas, J., Grefenstette, E., Whiteson, S., and Rocktäschel, T. A survey of reinforcement learning informed by natural language. In Kraus, S. (ed.), *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, pp. 6309–6317. ijcai.org, 2019. doi: 10.24963/ijcai.2019/880. URL https://doi.org/10.24963/ijcai.2019/880.

Lynch, C. and Sermanet, P. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems*, 2021. URL https://arxiv.org/abs/2005.07648.

Mirchandani, S., Karamcheti, S., and Sadigh, D. Ella: Exploration through learned language abstraction. *Advances in Neural Information Processing Systems*, 34: 29529–29540, 2021.

Mu, J., Zhong, V., Raileanu, R., Jiang, M., Goodman, N. D., Rocktäschel, T., and Grefenstette, E. Improving intrinsic exploration with language abstractions. In *NeurIPS*, 2022.

Nair, S., Mitchell, E., Chen, K., Ichter, B., Savarese, S., and Finn, C. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*, 2021. URL https://api.semanticscholar.org/CorpusID:237385309.

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

Padmakumar, A., Thomason, J., Shrivastava, A., Lange, P., Narayan-Chen, A., Gella, S., Piramuthu, R., and Gokhan Tur and, D. H.-T. TEACh: Task-driven Embodied Agents that Chat. In *Conference on Artificial Intelligence (AAAI)*, 2022. URL https://arxiv.org/abs/2110.00534.

Piantadosi, S. T. and Hill, F. Meaning without reference in large language models. *ArXiv*, abs/2208.02957, 2022.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Reed, S., Zolna, K., Parisotto, E., Colmenarejo, S. G., Novikov, A., Barth-Maron, G., Gimenez, M., Sulsky, Y., Kay, J., Springenberg, J. T., et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

Reimers, N. and Gurevych, I. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL http://arxiv.org/abs/1908.10084.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

Robine, J., Höftmann, M., Uelwer, T., and Harmeling, S. Transformer-based world models are happy with 100k interactions. *arXiv preprint arXiv:2303.07109*, 2023.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

Sharma, P., Torralba, A., and Andreas, J. Skill induction and planning with latent language. *arXiv preprint arXiv:2110.01517*, 2021.

Shridhar, M., Thomason, J., Gordon, D., Bisk, Y., Han, W., Mottaghi, R., Zettlemoyer, L., and Fox, D. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020a. URL https://arxiv.org/abs/1912.01734.

Shridhar, M., Yuan, X., Côté, M.-A., Bisk, Y., Trischler, A., and Hausknecht, M. Alfworld: Aligning text and embodied environments for interactive learning. *arXiv preprint arXiv:2010.03768*, 2020b.

Shridhar, M., Manuelli, L., and Fox, D. Cliport: What and where pathways for robotic manipulation. In *Conference on Robot Learning*, pp. 894–906. PMLR, 2022.

Singh, I., Singh, G., and Modi, A. Pre-trained language models as prior knowledge for playing text-based games. *arXiv preprint arXiv:2107.08408*, 2021.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4): 160–163, 1991.

Sutton, R. S. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Tam, A. C., Rabinowitz, N. C., Lampinen, A. K., Roy, N. A., Chan, S. C. Y., Strouse, D., Wang, J., Banino, A., and

Hill, F. Semantic exploration from language abstractions and pretrained representations. In *NeurIPS*, 2022.

Thomason, J., Murray, M., Cakmak, M., and Zettlemoyer, L. Vision-and-dialog navigation. In *Conference on Robot Learning*, 2019. URL https://api.semanticscholar.org/CorpusID:195886244.

Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., and Anandkumar, A. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv: Arxiv-2305.16291*, 2023.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

Winograd, T. Understanding natural language. *Cognitive Psychology*, 3(1):1–191, 1972. ISSN 0010-0285. doi: https://doi.org/10.1016/0010-0285(72)90002-3. URL https://www.sciencedirect.com/science/article/pii/0010028572900023.

Wu, Y., Min, S. Y., Prabhumoye, S., Bisk, Y., Salakhutdinov, R., Azaria, A., Mitchell, T., and Li, Y. SPRING: Studying papers and reasoning to play games. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=jU9qiRMDtR.

Yang, M., Du, Y., Ghasemipour, K., Tompson, J., Schuurmans, D., and Abbeel, P. Learning interactive real-world simulators. *arXiv preprint arXiv:2310.06114*, 2023.

Zhong, V., Rocktäschel, T., and Grefenstette, E. RTFM: generalising to new environment dynamics via reading. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SJgob6NKvH.

Zhong, V., Mu, J., Zettlemoyer, L., Grefenstette, E., and Rocktaschel, T. Improving policy learning via language dynamics distillation. In *Thirty-sixth Conference on Neural Information Processing Systems*, 2022.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. Fine-tuning language models from human preferences. *ArXiv*, abs/1909.08593, 2019.

A World Model Learning

Representation Learning The discrete codes z_t are vectors of one-hot categoricals that are sampled during the forward pass and optimized using straight-through gradients on the backward pass (Bengio et al., 2013; Hafner et al., 2020).

Two-hot Reward Prediction We follow DreamerV3 in predicting rewards using a softmax classifier with exponentially spaced bins that regresses the twohot encoding of the real-valued rewards and in clipping the regularizer at 1 free nat (Kingma et al., 2016). The two-hot regression decouples the gradient scale from the arbitrary scale of the rewards and free nats prevent over-regularization, known as posterior collapse.

B Actor Critic Learning

Because we optimize the policy from imagined rollouts, all involved quantities are predictions rather than environment observations. For simplicity, we omit the hats from the notation now and e.g. write z_t instead of $\hat{z}_t$. To train the actor and critic networks, we predict a sequence of $T = 15$ representations z_t by sampling from the world model and the actor network. The sequences start at all representations computed from the world model training step. From a sequence of representations z_t and recurrent states h_t, we fill in the rewards r_t and episode continuation flags c_t by applying their two MLPs, without invoking the image or language decoders. Given the quantities, we compute a λ-return (Sutton & Barto, 2018) that estimates the discounted sum of future rewards:

$$R_t = r_t + \gamma c_t \Big((1-\lambda) V(z_{t+1}, h_{t+1}) + \lambda R_{t+1} \Big) \qquad R_T \doteq V(z_T, h_T) \tag{1}$$

The return estimate R_t serves as a prediction target for the critic network, which uses discrete regression using a categorical cross entropy loss towards the twohot encoded targets. The actor network is trained to maximize the return estimates subject to an entropy regularizer on the action distribution:

$$\begin{aligned} \mathcal{L}_V &= \mathrm{catxent}(V_t(h_t, z_t),\, \mathrm{sg}(\mathrm{twohot}(R_t))) \\ \mathcal{L}_\pi &= -\,\mathrm{sg}(R_t - V(z_t, h_t)) / \max(1, S) \log \pi(a_t \mid h_t, z_t) - \eta\, \mathrm{H}\left[\pi(a_t \;\big|\; h_t, z_t)\right] \end{aligned} \tag{2}$$

To trade off the two actor loss terms without having to tune hyperparameters, the actor loss normalized returns that exceed a magnitude of 1 are normalized by an exponential moving average of the 5th to 95th percentile range of returns, $S = \mathrm{ema}(\mathrm{per}(R_t, 95) - \mathrm{per}(R_t, 5))$. When interacting with the environment, we choose actions by incorporating the new observation into the world model representation and then sampling from the actor network.

C Detailed Related Work

Language and Embodied Agents Language can be used in embodied settings in a variety of ways (Luketina et al., 2019). In instruction following, agents must interpret language specifications of high-level goals or step-by-step guidance (Branavan et al., 2010; Andreas & Klein, 2015; Anderson et al., 2018; Shridhar et al., 2020a; Lynch & Sermanet, 2021). Language can also be used as an abstraction to assist learning or decision-making, e.g. for planning by decomposing high-level tasks into low-level subgoals (Andreas et al., 2017; Jiang et al., 2019; Ahn et al., 2022; Huang et al., 2022a; Li et al., 2022; Sharma et al., 2021) or guiding exploration (Mirchandani et al., 2021; Tam et al., 2022; Mu et al., 2022; Du et al., 2023a). Instead of using language as a scaffolding mechanism for planning or exploration, our model treats language as another modality in observation space and plans in latent space. Additionally, human language is far richer than imperative commands. Other work looks to diversify the utterances that agents can understand with *supervised learning on human-annotated or expert datasets* in embodied environments, e.g. to understand more complex instructions (Ku et al., 2020; Shridhar et al., 2020a) or environment descriptions (Zhong et al., 2022), ask for assistance in simulated navigation or household tasks (Thomason et al., 2019; Padmakumar et al., 2022; Abramson et al., 2020), answer questions (Das et al., 2018), communicate domain knowledge (Eisenstein et al., 2009; Branavan et al., 2010; Narasimhan et al., 2018; Zhong et al., 2020), or collaborate dynamically on a shared goal (Bara et al., 2021). While these works consider more realistic natural language, they often rely on supervised approaches and expensive human data (often with the assumption of aligned language annotations and expert demonstrations).

Our work investigates how to unify these settings so that agents can learn from all kinds of language they might encounter in the world, including instructions and descriptions. While most of these works directly condition policies on language to generate actions (model-free), our algorithm uses language for future prediction, learning a world model that is then used for planning and acting.

Multimodal Models Developing agents that can leverage both vision and text observations requires training multimodal models. Previous works develop vision-language models (VLMs) by augmenting LLMs with visual encoders (Alayrac et al., 2022; Li et al., 2023b; Chen et al., 2022; Guo et al., 2023) or training models jointly over all modalities (Lu et al., 2022) However, because VLMs are prohibitively expensive to query and finetune, recent work on using VLMs as policies has focused on supervised learning from demonstrations (Driess et al., 2023; Jiang et al., 2022), rather than using them in embodied agents that can learn online. Reed et al. (2022) trains a multimodal embodied agent across various tasks, modalities, and embodiments by additionally learning to generate actions. Perhaps most similar, a recent line of work (Du et al., 2023b; Yang et al., 2023) trains text-conditioned video models for planning. Their approach works by using a video model trained on expert demonstrations to generate a video plan conditioned on a text instruction, and then imputing the actions to execute that plan. From a generative modeling perspective, our approach differs in that it also learns to generate language instead of being solely input text-conditioned, enabling text-only pretraining, interactive dialogue, and future possibilities for learning shared representations of text and video. Additionally beyond both VLMs and text-conditioned video models, our approach enables learning from online experience in addition to offline pretraining, allowing agents to improve their behaviors and understanding of the world autonomously rather than being inherently limited to an offline dataset.

Decision-making with Large Language Models Large language models (LLMs) learn about the world via next-token prediction on web-text, implicitly modeling world state (Li et al., 2021; 2023c) and relations between concepts (Piantadosi & Hill, 2022). When acting in purely text-based or symbolic environments, language models can be used as complete world models (Ammanabrolu & Riedl, 2018; Singh et al., 2021). In visual environments, LLMs can be used to break down complex language context into simple instructions for low-level policies (Huang et al., 2022a; Li et al., 2022; Ahn et al., 2022), integrate knowledge from text (Wu et al., 2023), or directly serve as the policy (Driess et al., 2023; Wang et al., 2023; Carta et al., 2023). However, LLMs are not grounded to real environment observations and cannot directly take actions unless observations are translated to text (Shridhar et al., 2020b; Huang et al., 2022b; Dasgupta et al., 2023), and representing visual inputs as text is inherently low bandwidth. Additionally, while LLMs can be used as a prior over actions or observations (Li et al., 2023a), they are difficult to update with feedback from the environment except in limited cases (Carta et al., 2023; Dagan et al., 2023). In contrast, we learn a single multimodal world model from experience with autoregressive prediction on both text and images (predicting both modalities in the future from both modalities as input), thus grounding language to *experience* (Bisk et al., 2020). Our model can also be trained on text-only data as a language model or video-only data as a video prediction model.

D Environment Details

D.1 HomeGrid

The HomeGrid environment is a grid with different objects, receptacles, and rooms. Agents receive pixel observations of 3x3 grid cells centered on the current agent position. The action space is: movement (`left, right, up, down`), object interaction (`pick up, drop`), and trash bin interaction (`get, pedal, grasp, lift`). The agent can carry one object in its inventory by executing the `pick up` action in front of an object or the `get` action in front of a trash bin with an object inside. There are three rooms (living room, dining room, kitchen) indicated by different flooring textures, three possible trash bin types with different colors (blue recycling, black trash, green compost) and four possible trash object types (bottle, fruit, papers, plates). Trash bins can be open, closed, or knocked over (represented visually as toppled over sideways). Each trash bin can be opened with a specific action that is randomly selected from `{pedal, grasp, lift}` in each episode. If agents apply the wrong action on a bin, it becomes broken and cannot be interacted with further until reset by the environment. When a trash bin is open, one object can be dropped into the bin with the `drop` action and the current object in the bin (if any) can be retrieved into the agent's inventory with `get`.

For each episode, the environment is randomly initialized with two objects and two trash bins in random positions. Trash bins are initialized in the open state with probability 0.5. One bin is irreversibly broken if the wrong action is applied and the other bin is reset after 5 timesteps if broken. At each timestep, each object is moved to a new position with probability 0.05 and new objects are spawned with probability $0.1 * \texttt{num\_remaining\_unique\_objects}$ at a random position.

In our experiments, agents are evaluated on setups with different language inputs: task instructions, task instructions + dynamics, task instructions + future observations, and task instructions + corrections. Language for each type is generated with templates from the underlying environment state, with the following semantics:

Tasks

- `find the [object/bin]`: the agent will receive a reward of 1 if it is facing the correct object / bin
- `get the [object]`: the agent will receive a reward of 1 if it has the correct object in inventory
- `put the [object] in the [bin]`: the agent will receive a reward of 1 if the bin contains the object
- `move the [object] to the [room]`: the agent will receive a reward of 1 if the object is in the room
- `open the [bin]`: the agent will receive a reward of 1 if the bin is in the open state

Future Observations: descriptions of environment state the agent may observe in the future

- `[object/bin] is in the [room]`: the object or bin is in the indicated room
- `i moved the [object] to the [room]`: the object has been moved to the room
- `there will be [object] in the [room] later`: the object will spawn in the room in five timesteps

Dynamics: descriptions of environment transitions

- `[action] to open the [bin]`: the indicated action is the correct action to open the bin

Corrections: task-specific feedback about the agent's current trajectory

- `no, turn around`: the agent's distance to the current goal object or bin (given the task) has increased compared to the last timestep

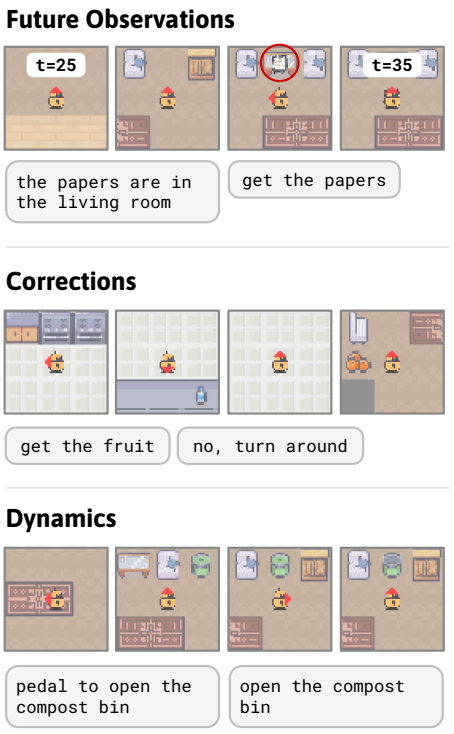


Figure D.1. HomeGrid provides language hints and task specifications. We show real trajectories from a trained agent.

Language is provided to the agent one token per timestep. All language are provided while the agent acts and the environment state is changing, except for dynamics descriptions (which apply to the whole episode). For dynamics descriptions, we randomly shuffle all possible descriptions and input them to the agent in sequence up to a maximum of 28 tokens while the agent is fixed in place. For language provided during the episode, on each timestep, if there is not currently an utterance being provided to the agent, either (1) the task instruction is repeated, every 20 timesteps (2) an utterance describing one of the events that occurred at this timestep is provided (i.e. objects moved or spawned) (3) a description of future observations or dynamics is provided (4) a correction is provided, with probability 0.1. If there is a new task instruction (i.e. the agent just completed the last task), any currently streaming sentence will be interrupted and the agent will immediately receive the tokens of the new instruction. All evaluation setups share the same underlying environment dynamics and parameters (e.g. each trash bin must be operated with the correct action even if the agent does not receive hints about dynamics).

D.2 Messenger

Language in Messenger is generated from human-written templates, resulting in diverse sentences with multiple ways of referring to each entity and a total vocabulary size of 1,125. Observations are a symbolic grid of entity IDs, and the agent takes discrete actions to move. We input the manual into the world model token-by-token before the episode begins.

The EMMA baseline provides a gridworld-specific inductive bias that each text token should map to some region in the current observation, and assumes that the model has access to the spatial locations of entities in the scene. As in the original benchmark, we initialize all models from the converged model trained on the previous game stage.

D.3 VLN-CE

The best-performing methods on VLN-CE use expert demonstrations (An et al., 2023) or train navigation-specialized hierarchical agents (Krantz et al., 2021). The VLN-CE training set consists of 10,819 unique natural instructions total, spread

across 61 scenes. The instruction and corresponding scene are randomly sampled per episode. In addition to language, the agent observes an egocentric RGB and depth image at each timestep. Agents have access to discrete low-level actions (moving forward 0.25 meters, turning left or right 15 degrees), as well as a `stop` action. Crucially, the agent must learn to take the `stop` action when it thinks it has reached the goal to indicate that it recognizes the goal position. This makes the task more challenging, as the agent must learn to only terminate the episode at the appropriate goal locations. The agent receives a dense reward at every timestep based on the delta in position from the goal. Following (Krantz et al., 2021), we provide an additional success reward of 1000 when the agent takes the `stop` action at the correct location, and a penalty of -10 when the agent takes the `stop` action elsewhere.

D.4 LangRoom

In LangRoom, the environment contains four objects in the corners of a room. The positions of the objects are fixed but the colors are randomized. The action space for the agent includes the four cardinal movement actions, stay, and 15 tokens that the agent can say. The language observations from the environment are *questions* "what color is the <object>?" followed by a random silence duration (allowing the agent to find out the answer), followed by the answer "it is <color>". After each question and answer, the colors are randomized and the environment asks a new question, up to a fixed episode length of 200 timesteps. Agents are rewarded $+1$ for saying the correct "<color>" token at the same timestep that the environment produces the "<color>" token, -0.1 for saying the wrong color at that timestep, -0.01 for speaking at other timesteps, and 0 for saying nothing. The agent only has a partial view over the environment, so it must move to the object before the environment starts prompting it for the answer.

E Text Pretraining: Text Generation Samples

Dynalang is not explicitly trained on the language modeling objective, but we can still generate text from the model by sampling rollouts from the world model and decoding the token from the latent representation at each timestep. Here, we show sampled 10-token generations conditioned on a prefix of 50 tokens for validation examples in TinyStories.

Prompt: Once upon a time, in a big forest, there lived a rhinoceros named Roxy. Roxy loved to climb. She climbed trees, rocks, and hills. One day, Roxy found an icy hill.
True: She had never seen anything like it before. It
Samples:
She wanted to climb down the hill.</s> friends and
It was a steep tree, but she was not
She wanted to own it, but it was too hot
She thought it would be fun banana scary, andffy
She wanted to skip through the. But Once upon

Prompt: Once upon a time, there was a thoughtful girl named Sue. Sue loved to help her mom around the house. One day, her mom asked her to wipe the table after they ate their lunch. Sue was happy to help. As
True: Sue was wiping the table, she saw
Samples:
they her big room. playly remembered her
she was placing,,, she saw a
she got being, she saw hera all she on
she was organizing, she saw the pin case in the
she was their best delicate turkey on, she saw

Prompt: Once upon a time, there was a little girl named Lucy. She had a pet cat named Tom. They loved to play together in the big green park near their house. One sunny day, they went to the park to play.
True: While playing, Tom saw a big s
Samples:
</s> Once upon a time, there was scarf
</s> " Jenny, you are my sweet. You must
</s> Once heard a kind girl and asked Mom to

</s> When taking a small, thin thing he
</s> The. lesson its if can improve and

Prompt: Once upon a time, there was a little boy named Tom. He loved to play with his red ball. One sunny day, Tom went outside to play with his ball in the land near his home. Tom kicked the ball high in
True: the sky. The ball went far, far away
Samples:
the sky and ity it."</s> Once day,
the air and loved then it rain outside. We can
the sky, but was enormous diary to with baby
the sky.</s> red ball went and all game,
the air and ran after to catchMoo. His was

Prompt: Once upon a time, there was a girl named Mia. Mia loved her jewelry. She had a big box full of pretty things. She liked to wear them all day. But at night, she had to sleep. One
True: day, Mia met a talking cat named
Samples:
day, shea was mad. She did not want
night, shea socks out wanted to hurt up.
day, shea could not find her skirt dress She
day, hera's mom came to her.
day, Miaa fell her hair could. It

Prompt: Once upon a time, there was a little boy named Tom. Tom had a special belt that he loved to wear. One day, he could not find his belt and felt very sad. Tom's mom saw him and
True: asked, "Why are you sad, Tom?"
Samples:
frustrated and asked him what was rude.</s> Once upon
asked, "Why are you sad, Tom?"</s>
asked, "Howeny, I did, get
said, "Don't worry, Tom. We
said, "To tree, you look be in

Prompt: Once upon a time, in a small house, there lived a kind and honest girl named Lily. She loved to bake cakes for her friends and family. One day, she made a big, yummy cake for her best friend
True: 's birthday. Lily carefully put the cake
Samples:
, Ben.</s> Tom went Mike opened the and,
, Tom.</s> Oneo decided the biggest ow
, Tim.</s> Once upon a time, there
, Tim.</s> lady.</s> </s> and Lily
, Tom.</s> Once upon a time, there

Prompt: One day, a young boy named Tim found a dull, round rock. He picked it up and looked at it. He thought it was not very fun, but he took it with him to the park. At the park, Tim
True: saw a girl named Sue. She had
Samples:
he met. favorite friend He put it his to
met a girl named Sue. Sue saw the ball
saw a stick top Sam. He kept playing with
played with his friends and but they friends!"</s> Li
met a girl named Lily.</s> ly saw

Prompt: Once upon a time, there was a little boy named Tim. Tim loved candy more than anything else. One day, Tim saw a big candy

store. He was very happy and ran to the store. Inside the store, Tim met
True: a strange man. The man said, "
Samples:
a nice lady named Sue.</s> The thing the
a tall named Max.</s> that the clever
a girl friend named Sue. They said, "
a big dog named Theffy.</s> said
a new prize car two cars. things.</s>

Prompt: Once upon a time, there was a big, heavy alligator. He lived near a small pond. He was very hungry and wanted to eat something. One day, a little bunny came close to the
True: pond. The alligator saw the bun
Samples:
flower. The bunny said, "Hello
kitchen. He thisly and said, "This
bunny and askede "Do, smell you
sunflower. The bun said, "Stop, sunset
bunny. The bunny said, "

F Qualitative Analysis

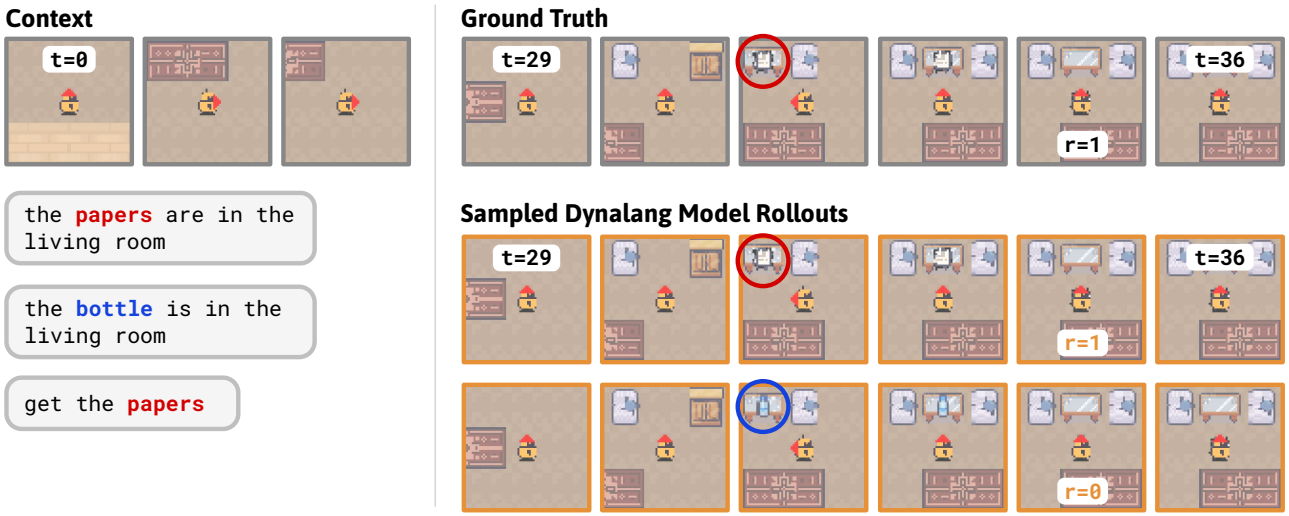


Figure F.1. Imagined rollouts from the world model. Conditioned on a language description, the task, and the same action sequence, we sample rollouts of the world model's imagined trajectories. Since the papers and bottle can be in any of multiple possible locations in the living room, the model samples exhibit uncertainty over the possible futures. In one rollout (top), the agent predicts the papers are on the table and correctly predicts it will get rewarded for picking it up. In the second rollout (bottom), it predicts that the bottle is on the table and that it will not get rewarded.

Figure F.1 shows that we can interpret what the model has learned by rolling out the world model state into the future and reconstructing observations from the latent state, conditioned on some history. We can see that the model represents the information and correctly grounds it to observations: given the information that the papers and bottle are in the living room, different samples from the world model represent different possible futures, both of which are consistent with the text. The model also correctly predicts that in the future where the papers are on the table, it will receive a reward of +1 for doing a pickup action, and that it will not be rewarded if it picks up the bottle.

G HomeGrid Training Curves

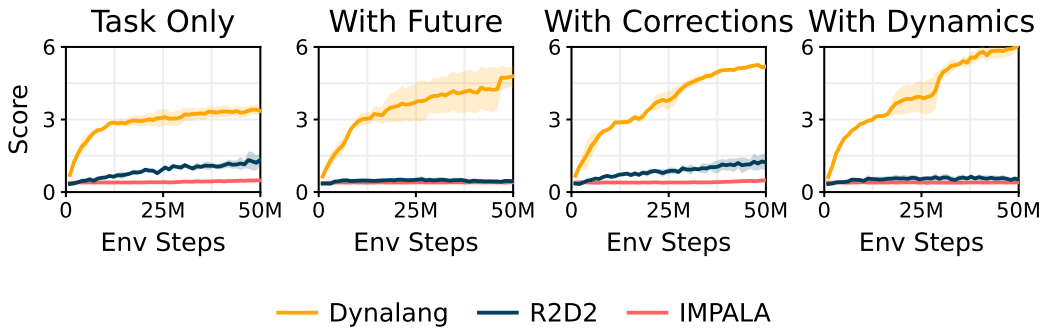
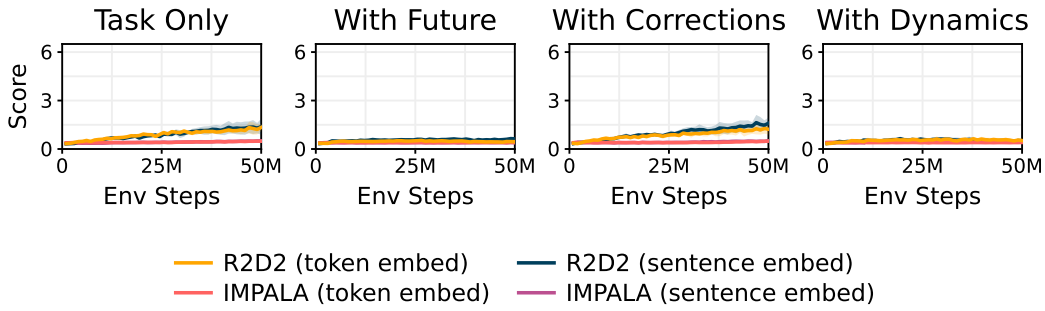


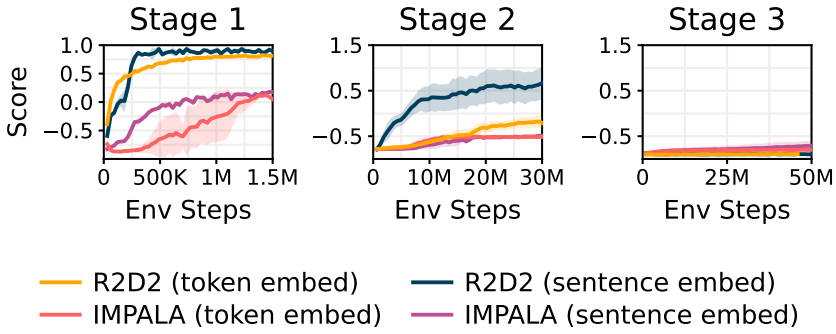
Figure G.1. HomeGrid training curves.

H Additional Baseline Experiments

H.1 Token vs. Sentence Embeddings for Baselines



(a) HomeGrid.



(b) Messenger.

Figure H.1. Token vs. sentence embedding performance for IMPALA and R2D2 on all tasks, averaged across 3 seeds. Sentence embeddings help R2D2 perform better on Messenger S1 and S2 but does not help consistently across tasks and methods.

H.2 Model Scaling for Baselines

We find that scaling the baseline R2D2 and IMPALA models does not improve their performance. Stage 2 runs were initialized from scratch.

Model Size	LSTM hidden size	Language MLP size	CNN hidden size	Policy/Value Hidden Size
1.7M	256	256	[16, 32, 32]	None (linear)
10M	1024	512	[16, 32, 32]	[512]
37M	2048	1024	[64, 64, 64]	[1024, 1024]

Table H.1. R2D2 architecture sizes for model scaling experiment.

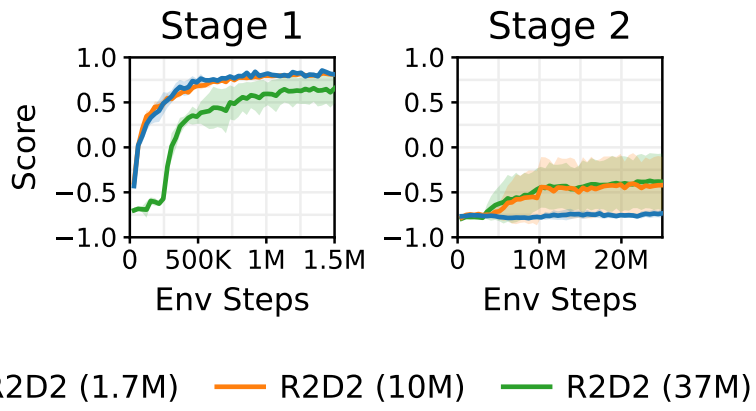


Figure H.2. Model scaling curves for R2D2.

Model Size	LSTM hidden size	Language MLP hidden size	CNN hidden size	Policy/Value Head
1.5M	512	[64]	[16, 32, 32]	None (linear)
8.8M	1024	[512]	[16, 32, 32]	[512]
34M	2048	[1024]	[16, 32, 32]	[1024, 1024]

Table H.2. IMPALA architecture sizes for model scaling experiment.

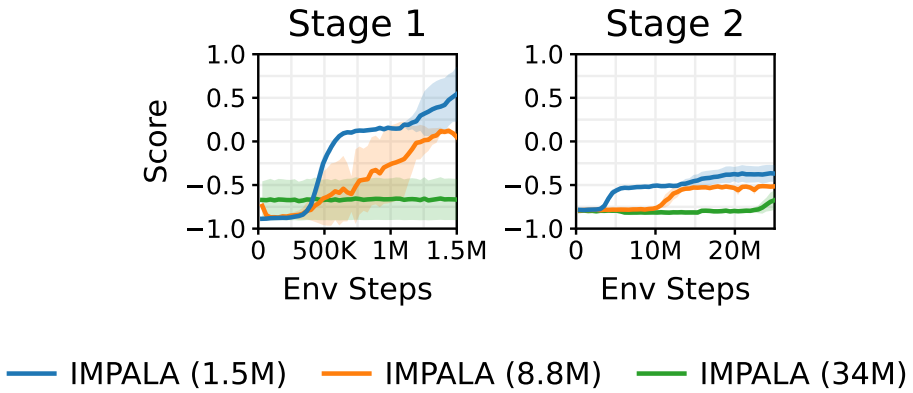


Figure H.3. Model scaling curves for IMPALA.

H.3 Auxiliary Reconstruction Loss for Baselines

We tried adding an auxiliary loss for reconstructing the visual and language observations at the current timestep. The loss was implemented by adding a linear layer that predicts each auxiliary target from the LSTM hidden state. The loss used is MSE (for continuous values) or cross-entropy (for discrete language vocab tokens). The auxiliary loss was added to the RL loss with a loss scale of 1. This did not meaningfully change performance.

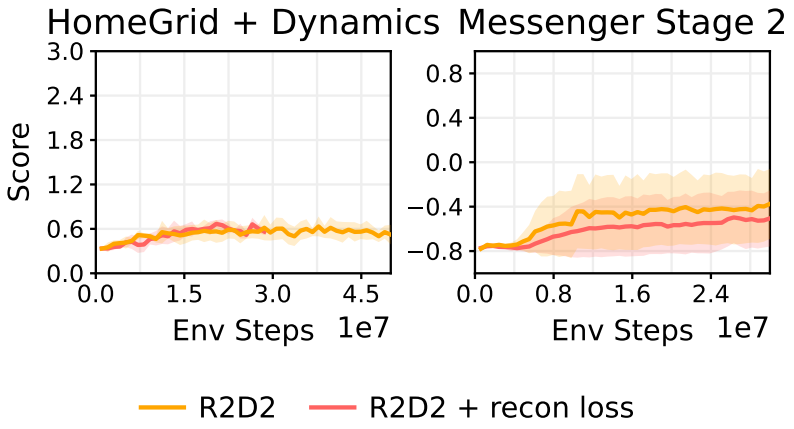


Figure H.4. Model-free R2D2 performance with an auxiliary reconstruction loss.

I Model and Training Details

I.1 Baseline Hyperparameters

	HomeGrid	Msgr S1	Msgr S2	Msgr S3	VLN
Total model parameters	27M	10M	10M	10M	10M
Language inputs	One-hot	T5 Embed	T5 Embed	T5 Embed	T5 Embed
Vocabulary size	32100	n/a	n/a	n/a	n/a
Language MLP layers	1	1	1	1	1
Language MLP units	512	512	512	512	512
Image input	Pixel	Symbol	Symbol	Symbol	Pixel
Image size	(64, 64, 3)	(16, 16, 17)	(16, 16, 17)	(16, 16, 17)	(64, 64, 3)
Replay ratio	7	7	7	7	7
Batch size	32	64	16	16	8
Unroll length	100	100	100	100	100
LSTM recurrent units	1024	1024	1024	1024	1024
Learning rate	4.8e-4	4.8e-4	4.8e-4	4.8e-4	4.8e-4
Buffer Size	1000	1000	1000	1000	1000
Env steps	50M	1M	25M	50M	30M
Number of envs	80	80	80	80	5

Table I.1. Model hyperparameters and training information for the R2D2 baseline.

	HomeGrid	Msgr S1	Msgr S2	Msgr S3
Total model parameters	10M	9M	9M	9M
Language inputs	One-hot	T5 Embed	T5 Embed	T5 Embed
Vocabulary size	32100	n/a	n/a	n/a
Language MLP layers	1	1	1	1
Language MLP units	512	512	512	512
Image input	Pixel	Symbol	Symbol	Symbol
Image size	(64, 64, 3)	(16, 16, 17)	(16, 16, 17)	(16, 16, 17)
Batch size	16	64	64	64
LSTM recurrent units	1024	1024	1024	1024
Learning rate	3e-4	3e-4	3e-4	3e-4
Env steps	50M	1M	25M	50M
Number of envs	80	80	80	80

Table I.2. Model hyperparameters and training information for the IMPALA baseline.

I.2 Dynalang Hyperparameters

We use the default model hyperparameters for the XL DreamerV3 model unless otherwise specified below. For VLN, we use a larger GRU deterministic state and a bottleneck layer of size 1024 between timesteps. To process both one-hot and embedding language inputs, we use a 5-layer MLP with 1024 MLP units in each layer. All models were trained on NVIDIA A100 GPUs.

	HomeGrid	**Msgr S1**	**Msgr S2**	**Msgr S3**	**VLN**	**LangRoom**
Total model parameters	281M	148M	148M	148M	268M	243M
Language inputs	One-hot	T5 Embed	T5 Embed	T5 Embed	T5 Embed	One-hot
Vocabulary size	32100	n/a	n/a	n/a	n/a	15
Language MLP layers	5	5	5	5	5	5
Language MLP units	1024	1024	1024	1024	1024	1024
Image input	Pixel	Symbol	Symbol	Symbol	Pixel	Pixel
Image size	(64, 64, 3)	(16, 16, 17)	(16, 16, 17)	(16, 16, 17)	(64, 64, 3)	(64, 64, 3)
Train ratio	32	64	64	32	32	16
Batch size	16	16	24	24	8	16
Batch length	256	256	512	512	256	64
GRU recurrent units	4096	4096	4096	4096	8192	6144
Bottleneck units	n/a	n/a	n/a	n/a	1024	2048
Env steps	50M	1M	25M	50M	30M	45M
Number of envs	66	16	16	66	8	4
Training time (GPU days)	3.75	2.5	16	24	16	2

Table I.3. Dynalang hyperparameters and training information for each environment.