Learning to Model the World with Language

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

UC Berkeley

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 learn to execute simple language instructions from task rewards, we aim to build agents that leverage diverse language that conveys general knowledge, describes the state of the world, provides interactive feedback, and more. Our key idea is that language helps agents predict the future: what will be observed, 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 present Dynalang, an agent that learns a multimodal world model to predict future text and image representations and learns to act from imagined model rollouts. Unlike traditional agents that use language only to predict actions, Dynalang acquires rich language understanding by using past language also to predict future language, video, and rewards. In addition to learning from online interaction in an environment, Dynalang can be pretrained on datasets of text, video, or both without actions or rewards. From using language hints in grid worlds to navigating photorealistic scans of homes, Dynalang utilizes diverse types of language to improve task performance, including environment descriptions, game rules, and instructions.

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 [68]. Current embodied agents can follow simple, low-level instructions like "get the blue block" [48] or "go past the elevator and turn right" [5]. However, to communicate freely interactive agents should understand the full range of ways people use language beyond the "here and now" [30]: transmitting 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." Much of what we read in text or hear from others communicates *knowledge about the world*, either about how the world works or about the current state of the world.

How could we enable agents to use diverse types of language? One way to train language-conditioned agents to solve tasks is reinforcement learning (RL). However, current language-conditioned RL methods primarily learn to generate *actions* from task-specific instructions, e.g. taking a goal description like "pick up the blue block" as an input and outputting a sequence of motor controls. When we consider the diversity of functions that natural language serves in the real world, directly mapping language to optimal actions presents a challenging learning problem. 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. When language does not talk about the task, it is only weakly correlated with optimal actions the agent should take. Mapping language to actions, particularly using task reward alone, is therefore a weak learning signal for learning to use diverse language inputs to accomplish tasks.

Website: dynalang.github.io Correspondence to: jessy_lin@berkeley.edu

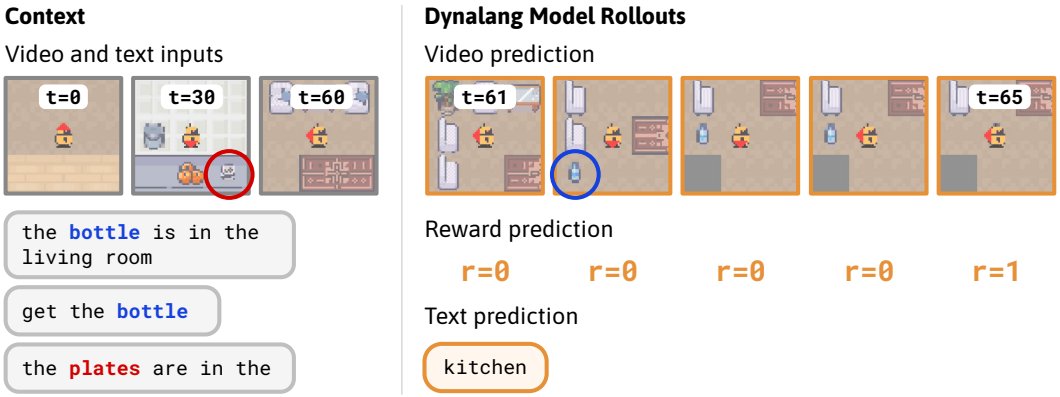


Figure 1: Dynalang learns to use language to make predictions about future (text + image) observations and rewards, which helps it solve tasks. Here, we show real model predictions in the HomeGrid environment. The agent has explored various rooms while receiving video and language observations from the 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."

Instead, we propose that a unifying way for agents to use language is to help them *predict the future*. The utterance "I put the bowls away" helps agents make better predictions about future observations (i.e., that if it takes actions to open the cabinet, it will observe the bowls there). Much of the language we encounter can be grounded in visual experience in this way. Prior knowledge such as "wrenches can be used to tighten nuts" helps agents predict environment transitions. Statements such as "the package is outside" help agents predict future observations. This framework also unifies standard instruction following under predictive terms: instructions help agents predict how they will be rewarded. Similar to how next-token prediction allows language models to form internal representations of world knowledge [52], we hypothesize that predicting future representations provides a rich learning signal for agents to understand language and how it relates to the world.

We present Dynalang, an agent that learns a world model of language and images from online experience and uses the model to learn how to act. Dynalang decouples learning to model the world with language (supervised learning with prediction objectives) from learning to act given that model (reinforcement learning with task rewards). The world model receives both visual and textual inputs as observation modalities and compresses them to a latent space. We train the world model to predict future latent representations with experience collected online as the agent acts in the environment. We train the policy to take actions that maximize task reward, taking the latent representation of the world model as input. Because world modeling is separated from action, Dynalang can be pretrained on single modalities (text-only or video-only data) without actions or task reward. Additionally, *language generation* can also be unified in our framework: the agent's perception can inform an agent's language model (i.e., its predictions about future tokens), enabling it to speak about the environment by outputting language in the action space.

We evaluate Dynalang on a broad range of domains with different types of language context. In a multi-task home cleanup environment, Dynalang learns to use language hints about future observations, environment dynamics, and corrections to accomplish tasks more efficiently. On the Messenger benchmark [29], Dynalang can read game manuals to fit the most challenging stage of the game, outperforming task-specific architectures. In vision-language navigation [38], we demonstrate that Dynalang can learn to follow instructions in visually and linguistically complex domains.

Our contributions are as follows:

- We propose Dynalang, an agent that grounds language to visual experience via future prediction.
- We demonstrate that Dynalang learns to understand diverse kinds of language to solve a broad range of tasks, often outperforming state-of-the-art RL algorithms and task-specific architectures.
- We show that the Dynalang formulation enables additional capabilities: *language generation* can be unified in the same model, as well as text-only pretraining without actions or task rewards.

Figure 2: We consider a wide range of environments that feature visual inputs and diverse types of language. HomeGrid is a challenging visual grid world 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, where the agent has to locate objects in hundreds of scenes. LangRoom is a simple visual grid world with partial observability, where the agent needs to produce both motor actions and language.

2 Related Work

Much work has focused on teaching reinforcement learning agents to utilize language to solve tasks by directly conditioning policies on language [5, 61, 47] or by augmenting agents with large language models (LLMs) [44, 1, 31]. While most of these agents focus on learning from high-level specifications of goals or step-by-step guidance, relatively few works have addressed learning to use broader types of language such as descriptions of how the world works [10, 69, 29]. Instead of directly learning a language-conditioned policy, we learn a language-conditioned world model and demonstrate its ability to understand diverse kinds of language about the world in a single model. Additionally, in contrast to LLM-based policies, Dynalang is multimodal, extending the next-token prediction paradigm to observations of both language and images rather than relying on translating observations to text. Dynalang can also be updated online, allowing the agent to continually learn language and how it relates to the world. We refer to Appendix C for detailed related work.

3 Dynalang

Dynalang learns to utilize diverse types of language in visual environments by encoding multiple modalities into compressed representations and then predicting the sequence of future representations given actions. For our algorithm, we build on the model-based reinforcement learning algorithm DreamerV3 [28] and extend it to process, and optionally produce, language. The world model is continuously 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 algorithm from sequences of representations imagined by the world model. The algorithm is summarized in Algorithm 1.

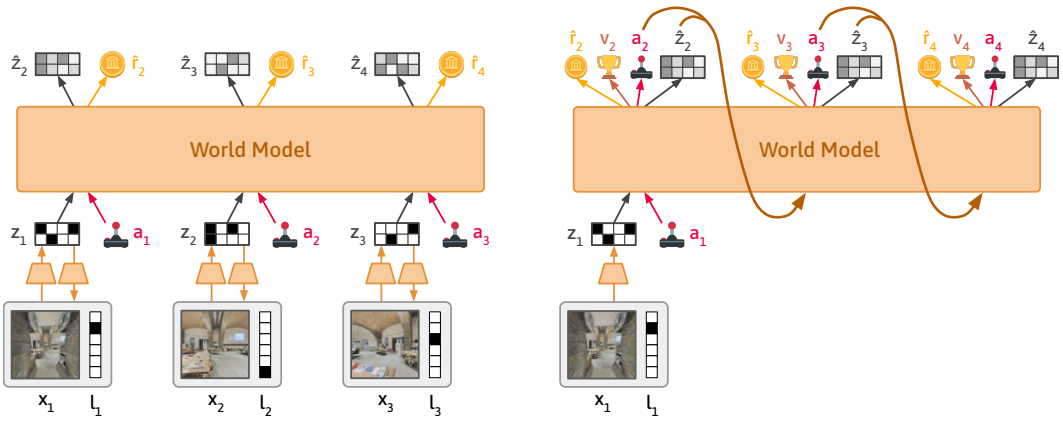
Problem setting To perform interactive tasks, an agent chooses actions a_t to interact with an environment that responds with rewards r_t, a flag for whether the episode

Algorithm 1: Dynalang

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 $\mathcal{L}_{\text{pred}} + \mathcal{L}_l$.



(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.

continues c_t, and observations o_t. In this paper, we consider multimodal environments where $o_t = (x_t, l_t)$ consist of an image x_t and a language token 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 We consider a diverse range of environments, summarized in Figure 2, where agents receive a continuous stream of video and text observations. While previous settings specify that language such as instructions arrive at the beginning of an episode, we are interested in enabling agents to act in more flexible settings where they face a continuous stream of video and text, as in the real world. For humans, reading, listening, and speaking extends over time, during which we receive new visual inputs and can perform motor actions. Analogously, we provide our agent with one video frame and one language token at each time step and the agent produces one motor action, and in applicable environments one language token, per time step. An additional benefit of providing one language token per time step is that the algorithm does not have to decide on an arbitrary way to segment language temporally, enabling Dynalang to be applied across a range of tasks with diverse kinds of language. We show in Section 4.6 that token-level representations substantially outperform sentence-level representations.

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:

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

We implement the world model as a Recurrent State Space Model [RSSM 26], where the sequence model is implemented as GRU [15] with recurrent state h_t. Using a recurrent model has the benefit that the policy does not have to integrate information over time anymore, but other sequence models such as Transformers can also be used [13, 58]. At each timestep, the encoder conditions on the observations and model state h_t, effectively learning to compress observations to codes z_t relative

to the history. The sequence model then conditions on the encoded observations z_t to integrate new observations into the next model state. 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 [36, 57]. The representations are shaped by reconstructing the input observations, providing a rich learning signal for grounding. 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 a prior distribution over codes. We use the predicted distribution over $\hat{z}_t$ as this prior, essentially regularizing the representations to be predictable. The representation learning loss $\mathcal{L}_{\text{repr}}$ is the sum of terms:

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

Here, we denote the categorical cross entropy loss as $\operatorname{catxent}$, the binary cross entropy loss as $\operatorname{binxent}$, the stop gradient operator as sg, and $\beta_{\text{reg}} = 0.1$ is a hyperparameter. As the network architecture 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 [54]. 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}, whih is trained to match the actual representation at the next timestep z_t. Concretely, the future prediction objective is:

$$\text{Prediction loss:} \quad \mathcal{L}_{\text{pred}} = \beta_{\text{pred}} \max(1, \operatorname{KL}\big[\operatorname{sg}(z_t) \;\big\|\; \hat{z}_t\big]) \tag{3}$$

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 contain information from current observation, but also 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 from its inputs, 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 living room" 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. This objective provides a rich learning signal for grounding.

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 4.5, 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 prediction of the *representation* at the next timestep.

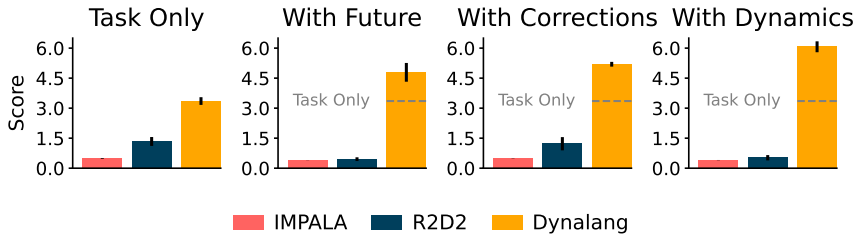


Figure 4: HomeGrid performance after 50M steps (2 seeds). Dynalang learns to use all types of language hints to achieve higher scores than when just provided with the task information, outperforming language-conditioned IMPALA and R2D2, where we see performance decrease when we include language hints.

3.2 Policy Learning

To select actions, we train an actor critic algorithm [67] purely from imagined sequences of multimodal representations predicted by the world model [64]. Unlike many other language-conditioned RL agents, our policy network is not directly conditioned on language. Instead, it leverages the rich multimodal representations learned by the world model, which contain compressed information about both visual inputs and language. The purpose of the actor network is to predict a distribution over actions, which could be a categorical over possible movement directions or language tokens to speak, or a factorized action space of both. The purpose of the critic is to estimate the discounted sum of future rewards for each state to guide the actor learning. Both networks are MLPs:

$$\text{Actor network:} \quad \pi(a_t|h_t, z_t) \qquad \text{Critic network:} \quad \mathrm{V}(h_t, z_t) \tag{4}$$

We do not modify the policy learning algorithm of DreamerV3 and refer to Appendix B for details. In short, during training time, we generate imagined rollouts of length $T = 15$ to train the policy. Starting from states sampled from the replay buffer, we sample actions from the actor network and observations from the world model. The world model also predicts rewards and continuation flags, from which we compute λ-returns. The critic network is trained to regress these return estimates, whereas the actor network is trained to maximize them by REINFORCE [67]. During environment interaction, we sample actions from the actor without planning.

4 Experiments

Our experiments aim at investigating the following hypotheses:

H1) Dynalang enables agents to leverage language beyond instructions to improve task performance, without having to learn about the world via trial and error. To test this, we investigate whether adding different kinds of language hints in HomeGrid improves task performance (Section 4.1), and whether Dynalang can learn from game manuals in Messenger (Section 4.2).

H2) It is more useful to ground diverse language with the future prediction objective than to predict actions directly. To test this, we compare our method against model-free RL baselines.

H3) Interpreting instructions as future reward prediction is no worse than learning to predict actions directly from instructions, as is typically done. To test this, we compare performance to baselines with task-only language in HomeGrid and on vision-language navigation (Section 4.3).

H4) The Dynalang formulation additionally enables the agent to generate language (Section 4.4).

Language encodings We tokenize all text with the T5 tokenizer [54], 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 model-free RL baselines: IMPALA [22], an off-policy actor critic algorithm, and R2D2, an off-policy DQN-like algorithm [35]. 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 [23]. We pass the same language observations to the baselines as to our method (token-by-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 [56] and did not find a consistent improvement across our tasks. Both baseline models are $\sim$10M parameters, and we did not find that these models benefit from scaling parameter count.

4.1 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 task specifications in language but also *language hints*, as depicted in Figure 5. 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 autonomous interaction with the environment, as in standard RL. Agents can achieve higher performance if they learn to ground language to the environment.

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 over 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. In addition to task specifications, hints are sampled at random points throughout the episode and are provided token-by-token while the agent continues to act. We script the following language hints:

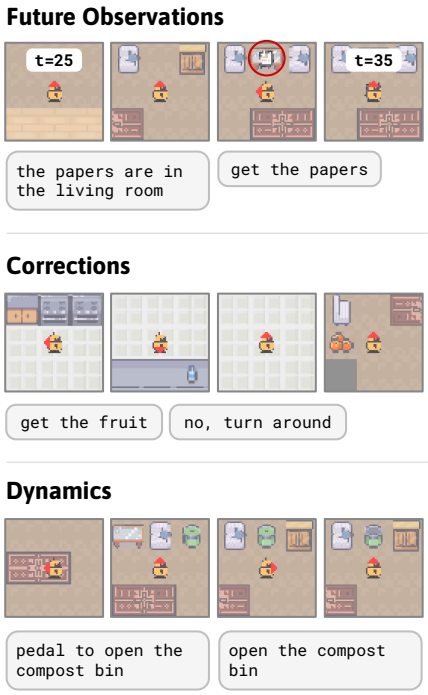


Figure 5: In HomeGrid, the agent is provided with language hints in addition to task specifications. We show real trajectories from a trained agent using language to maximize reward.

- **Future observations** Descriptions of where objects are in the world or where they have been moved. 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 or potentially the rest of the episode (irreversible dynamics).
- **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.

Figure 4 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. Dynalang learns to ground language to the environment purely via the future prediction objective. Language-conditioned 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 gets overwhelmed with more diverse language, while Dynalang improves across the board, supporting **H1** and **H2**. We hypothesize that additional language input makes it more difficult for the model-free methods to learn to process observations to solve the task.

4.2 Messenger: Game Manuals

Next, we evaluate Dynalang on the Messenger game environment [29], 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 roles and movement dynamics. The challenge is grounding the text references to the environment, which requires 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 (S1, S2, S3).

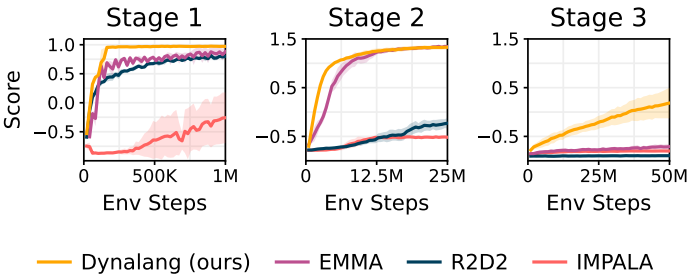


Figure 6: Messenger training performance (2 seeds). Dynalang outperforms language-conditioned IMPALA and R2D2, as well as the task-specific EMMA architecture, fitting the most complex stage of the game where other methods fail to achieve non-trivial performance.

The 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 presented as a symbolic grid of entity IDs, and the agent takes discrete actions to move. We input the manual into Dynalang by inputting the manual into the world model before the episode begins.

In addition to the baselines above, we compare the performance of Dynalang to EMMA, the original baseline for the benchmark that uses a specialized grid-based architecture for the task and learns a language-conditioned policy with PPO [59]. The architecture 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. Since we are not focused on studying generalization and distribution shift on this particular task, we compare models on the Messenger train environments rather than the benchmark held-out variations. 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**.

4.3 Vision-Language Navigation: Instruction Following

To evaluate how Dynalang performs in more complex environments, we apply it to the popular Vision-Language Navigation (VLN) [5] benchmark. Agents must navigate Matterport3D panoramas captured in real homes [12], following crowd-sourced natural language instructions that indicate where the agent should navigate to, such as "Go past the end of the bed to the door on the left. Enter the hallway,..." We focus on the more challenging variant, Vision-and-Language Navigation in Continuous Environments (VLN-CE) [38]. Rather than providing a waypoint navigation graph as in the original VLN task (which generally are not available when navigating in real homes), in VLN-CE agents move freely in a continuous environment. The best-performing methods on this task use expert demonstrations [4] or train navigation-specialized hierarchical agents [39]. In this task, our goal is to demonstrate that Dynalang can learn policies in this challenging instruction-conditioned RL setting while interpreting instructions as predicting future rewards.

Each episode randomly samples a language instruction and corresponding scene from the training dataset, which is comprised of 10,819 unique instructions total. The agent is trained with a dense reward based on relative positions to the current goal, a success reward when taking the `stop` action at the correct location, and a penalty otherwise. As shown in the example trajectory in Figure 7, the agent learns to follow naturally phrased navigation instructions in visually realistic home environments. Compared to the model-free R2D2 baseline, Dynalang succeeds at a significantly higher portion of the training instructions, 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.

Exit the bedroom, go straight down the hallway, make a right into the doorway of the bathroom and stop.

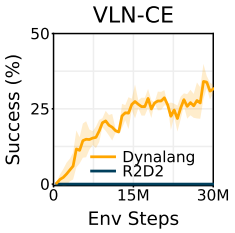


Figure 7: VLN-CE results. **(left)** A portion of a trained agent trajectory, given the instruction "Exit the bedroom, go straight down the hallway, make a right into the doorway of the bathroom and stop". **(right)** Success rate during RL training, averaged across 3 seeds for Dynalang and 2 seeds for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

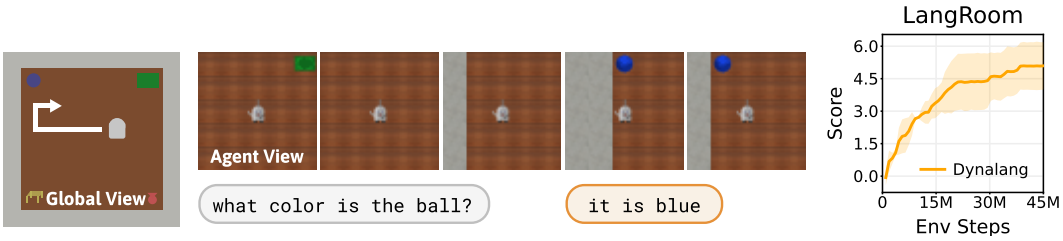


Figure 8: LangRoom results. **(left)** A real trajectory from a trained agent. The agent learns to take information-gathering actions from reward. When asked "what color is the ball?" the agent walks to the corner with the ball and generates the tokens "it is blue." **(right)** Training curve. The agent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

4.4 LangRoom: Embodied Question Answering

Finally, we 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. In contrast to language models, Dynalang is multimodal, enabling the agent to ground its language generation to the real world. 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 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 only has a partial view over the environment, so it must move to the object. When prompted by the environment, the agent is rewarded for emitting a language action saying the correct color. See Appendix D.3 for details on the task.

As shown in Figure 8, the agent learns to answer more questions correctly with task reward, supporting **H4**. We show an example trajectory demonstrating that the agent has learned to take information gathering actions to observe the color of the object and generate text consistent with the world state.

4.5 Text-only Pretraining

Dynalang can be pretrained on single-modality data by zeroing out the other modality and action inputs. This provides a way for RL agents to benefit from large-scale offline data in a single architecture. To evaluate this capability, we pretrain Dynalang from scratch on (1) **in-domain text** with manuals from Messenger S2 games (2) **domain-general text** with TinyStories [21], a dataset of 2M short stories generated by GPT-3.5 and 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 use the T5 vocabulary and compare S2 task performance with learned embeddings to using pretrained T5 embeddings, training all methods from scratch on S2 without initializing from S1. As shown in Figure 9, Dynalang is able to benefit from offline pretraining on text-only data. 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 can generate language from the world model by doing rollouts in latent space and reconstructing the token from the latent representation. One consequence of this approach is that the model can potentially do planning in *latent space* rather than token space. In Appendix E we show the model's preliminary language generation capabilities after pretraining on TinyStories, which suggest that Dynalang could potentially be trained and used as a language model.

4.6 Further Analysis

Language updates model predictions in interpretable ways. Figure 11 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

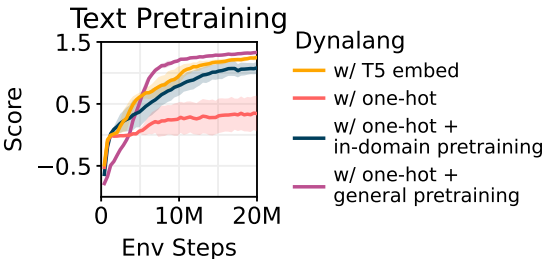


Figure 9: One-hot token encodings underperform pretrained embeddings on S2, but pretraining Dynalang with a small amount of text-only data closes much of the gap.

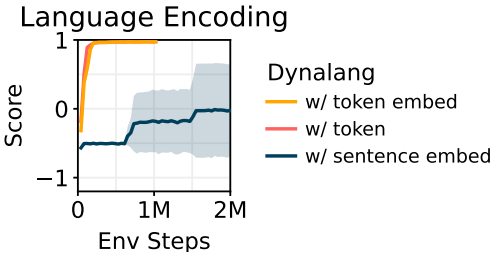


Figure 10: Sentence embeddings lead to much slower learning, even on S1 where both one-hot and pretrained token encodings quickly reach ceiling performance.

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 information described in 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.

Token representations outperform sentence representations. Figure 10 shows that consuming one sentence of the manual per timestep causes the agent to learn much more slowly, compared to our model which reads one token per timestep. We use embeddings from the Sentence Transformers `all-distilroberta-v1` model [56]. We hypothesize that the sentence encoder output could be a lossy bottleneck, making it difficult for Dynalang to extract information from the text particularly when the sentences contain a lot of information.

5 Discussion

Limitations Our recurrent architecture may make optimization challenging in extremely long horizon environments. Our design decision to interleave vision and language tokens one-to-one allows the agent to act while communicating but may cause sequence length to be the bottleneck for learning in some tasks. While Dynalang can generate text, the generation quality is not competitive with pure language models and further work will be needed to close that gap.

Conclusion We present Dynalang, an agent that grounds language to visual experiences, actions, and rewards through future prediction as a rich self-supervised objective. Dynalang learns to act based on various types of language across a diverse range of tasks, often outperforming model-free methods that struggle with increased language complexity. The ability to pretrain on video and text without actions or rewards suggests that Dynalang could be scaled to large web datasets, paving the way towards a self-improving multimodal agent that interacts with humans in the world.

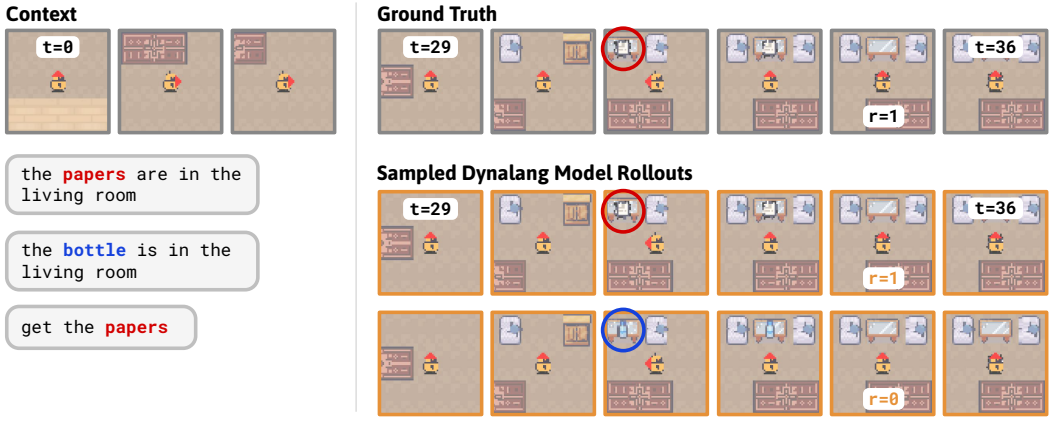


Figure 11: Imagined rollouts from the world model. Conditioned on a language description, the task, and the same action sequence, we sample rollouts of the world model's imagined trajectories. Since the papers and bottle can be in any of multiple possible locations in the living room, the model samples 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.

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).

References

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

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

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

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

[5] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel. 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*, pages 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.

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

[7] J. Andreas, D. Klein, and S. Levine. Modular multitask reinforcement learning with policy sketches. In D. Precup and Y. W. Teh, editors, *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*, pages 166–175. PMLR, 2017. URL http://proceedings.mlr.press/v70/andreas17a.html.

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

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

[10] S. Branavan, L. Zettlemoyer, and R. Barzilay. 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*, pages 1268–1277, Uppsala, Sweden, 2010. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P10-1129.

[11] T. Carta, C. Romac, T. Wolf, S. Lamprier, O. Sigaud, and P. Oudeyer. Grounding large language models in interactive environments with online reinforcement learning. *CoRR*, abs/2302.02662, 2023. doi: 10.48550/arXiv.2302.02662. URL https://doi.org/10.48550/arXiv.2302.02662.

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

[13] C. Chen, Y.-F. Wu, J. Yoon, and S. Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

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

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

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

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

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

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

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

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

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

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

[24] L. Fan, G. Wang, Y. Jiang, A. Mandlekar, Y. Yang, H. Zhu, A. Tang, D.-A. Huang, Y. Zhu, and A. Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. In *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022. URL https://openreview.net/forum?id=rc8o_j8I8PX.

[25] J. Guo, J. Li, D. Li, A. M. H. Tiong, B. Li, D. Tao, and S. Hoi. 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*, pages 10867–10877, 2023.

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

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

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

[29] A. W. Hanjie, V. Zhong, and K. Narasimhan. Grounding language to entities and dynamics for generalization in reinforcement learning. In M. Meila and T. Zhang, editors, *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*, pages 4051–4062. PMLR, 2021. URL http://proceedings.mlr.press/v139/hanjie21a.html.

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

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

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

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

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

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

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

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

[38] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee. 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*, pages 104–120. Springer, 2020.

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

[40] B. Z. Li, M. Nye, and J. Andreas. 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)*, pages 1813–1827, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.143. URL https://aclanthology.org/2021.acl-long.143.

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

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

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

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

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

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

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

[48] C. Lynch, A. Wahid, J. Tompson, T. Ding, J. Betker, R. Baruch, T. Armstrong, and P. Florence. Interactive language: Talking to robots in real time. *arXiv preprint arXiv:2210.06407*, 2022.

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

[50] J. Mu, V. Zhong, R. Raileanu, M. Jiang, N. D. Goodman, T. Rocktäschel, and E. Grefenstette. Improving intrinsic exploration with language abstractions. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/db8cf88ced2536017980998929ee0fdf-Abstract-Conference.html.

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

[52] F. Petroni, T. Rocktäschel, S. Riedel, P. Lewis, A. Bakhtin, Y. Wu, and A. Miller. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China, Nov. 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1250. URL https://aclanthology.org/D19-1250.

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

[54] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. 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.

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

[56] N. Reimers and I. Gurevych. 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.

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

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

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

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

[61] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. IEEE, 2020. doi: 10.1109/CVPR42600.2020.01075. URL https://doi.org/10.1109/CVPR42600.2020.01075.

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

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

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

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

[66] A. C. Tam, N. C. Rabinowitz, A. K. Lampinen, N. A. Roy, S. C. Y. Chan, D. Strouse, J. Wang, A. Banino, and F. Hill. Semantic exploration from language abstractions and pretrained representations. In *NeurIPS*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/a28e024ccd623ed113fb19683fa0910d-Abstract-Conference.html.

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

[68] T. Winograd. 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.

[69] V. Zhong, T. Rocktäschel, and E. Grefenstette. 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.

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 [8, 27].

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 [37]. 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 [65] that estimates the discounted sum of future rewards:

$$R_t = r_t + \gamma c_t \Big((1-\lambda) V(z_{t+1}, h_{t+1}) + \lambda R_{t+1} \Big) \qquad R_T \doteq V(z_T, h_T) \tag{5}$$

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:

$$\begin{aligned} \mathcal{L}_V &= \mathrm{catxent}(V_t(h_t, z_t),\, \mathrm{sg}(\mathrm{twohot}(R_t))) \\ \mathcal{L}_\pi &= -\,\mathrm{sg}(R_t - V(z_t, h_t)) / \max(1, S) \log \pi(a_t \mid h_t, z_t) - \eta\, \mathrm{H}\left[\pi(a_t \;\big|\; h_t, z_t)\right] \end{aligned} \tag{6}$$

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 [46]. In instruction following, agents must interpret language specifications of high-level goals or step-by-step guidance [10, 6, 5, 61, 47]. 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 [7, 33, 1, 31, 44, 60]. Instead of planning in language, our model treats language as another modality in observation space and plans in latent space. Finally, language can be used to describe the world, e.g. to enable semantic exploration [49, 66, 50, 19], to communicate domain knowledge [20, 10, 51, 69, 24], or as feedback from the environment [32]. 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 [2, 42, 14, 25] or training models jointly over all modalities [45] However, because VLMs are prohibitively expensive to query and finetune, recent work on using VLMs as policies has focused on supervised learning from demonstrations [18, 34], rather than using

them in embodied agents that can learn online. More similar to our work, [55] trains a multimodal embodied agent across various tasks, modalities, and embodiments by additionally learning to generate actions. Unlike prior approaches, our algorithm uses a future prediction objective to ground different modalities together, and we show that this enables our model to learn from rich language beyond instructions. Furthermore, our scheme is amenable to both online training and pretraining.

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 [40, 43] and relations between concepts [53]. When acting in purely text-based or symbolic environments, language models can be used as complete world models [3, 63]. In visual environments, LLMs are not grounded to real environment observations and cannot directly take actions, unless observations are translated to text [62, 32, 17]. However, representing visual inputs as text is inherently low bandwidth. Additionally, while LLMs can be used as a prior over actions or observations [41], they are difficult to update with feedback from the environment except in limited cases [11, 16]. 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* [9]. 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 * $`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

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 VLN-CE

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 [39], 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.3 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 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 HomeGrid Training Curves

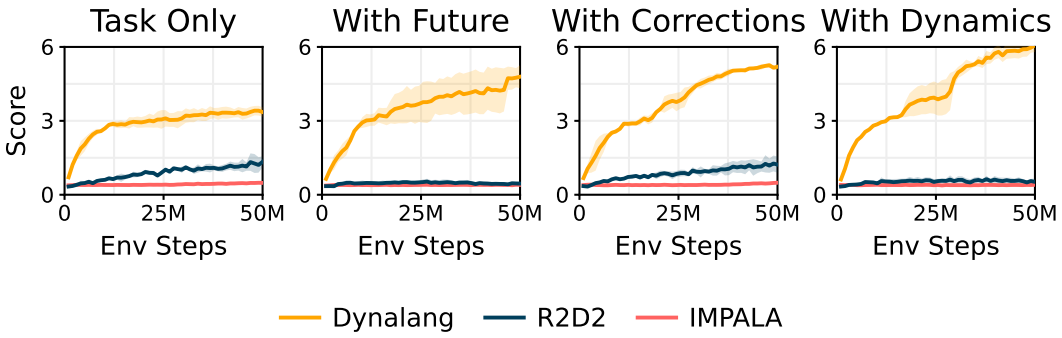
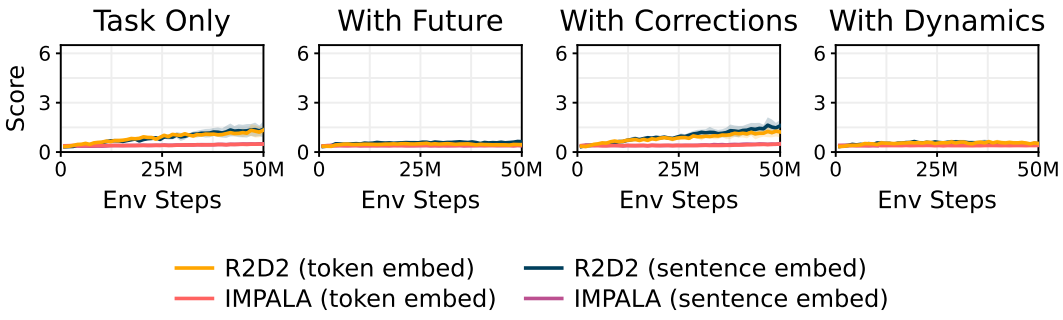


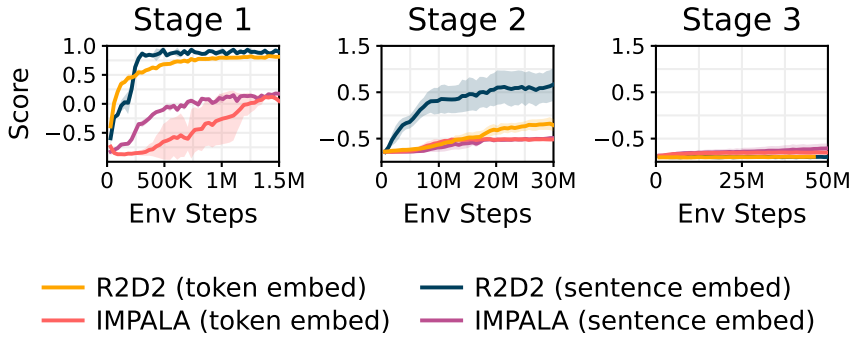
Figure F.1: HomeGrid training curves.

G Additional Baseline Experiments

G.1 Token vs. Sentence Embeddings for Baselines



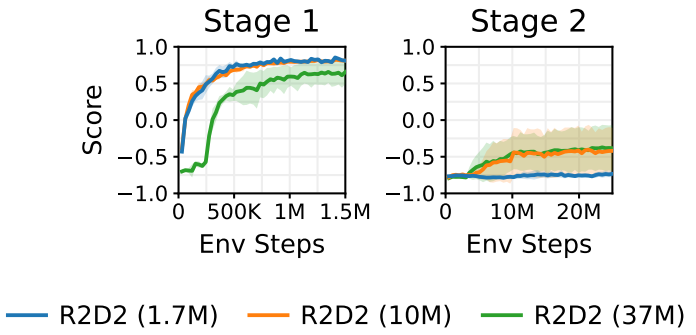
(a) HomeGrid.



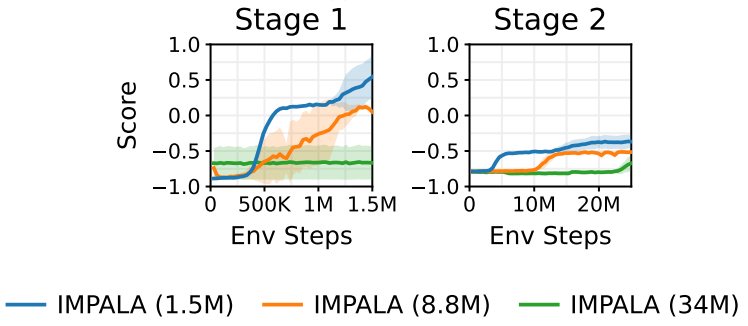
(b) Messenger.

Figure G.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.

G.2 Model Scaling for Baselines



(a) R2D2.



(b) IMPALA.

Figure G.2: Model scaling curves for R2D2 and IMPALA. Performance does not increase with larger model sizes. Stage 2 runs were initialized from scratch.

G.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.

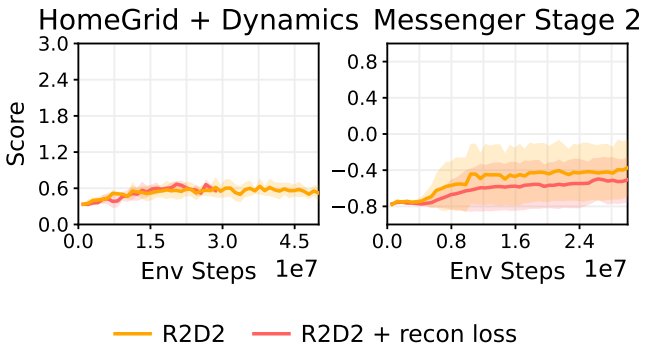


Figure G.3: Model-free R2D2 performance with an auxiliary reconstruction loss.

G.4 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 G.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 G.2: Model hyperparameters and training information for the IMPALA baseline.

G.5 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 G.3: Dynalang hyperparameters and training information for each environment.