ReSeeding Latent States for Sequential Language Understanding

Anonymous ACL submission

Abstract

We introduce Refeeding State Embeddings aligned using Environmental Data (RESEED), a novel method for grounding language in environmental data. While large language models (LLMs) excel at many tasks, they continue to struggle with multi-step sequential reasoning. RESEED addresses this by producing latent embeddings aligned with the true state of the environment and *refeeding* these embeddings into the model before generating its output. To evaluate its effectiveness, we develop three new sequential reasoning benchmarks, each with a training set of paired state-text trajectories and several text-only evaluation sets that test generalization to longer, unseen trajectories. Across all benchmarks, RESEED significantly improves generalization and scalability over a text-only baseline. We further show that RESEED outperforms commercial LLMs on our benchmarks, highlighting the value of grounding language in the environment.

1 Introduction

The continued scaling of large language models (LLMs) has led to impressive capabilities across a range of natural language tasks. Yet, the field is nearing the limits of available high-quality text data (Villalobos et al., 2024), and models trained solely on text data exhibit persistent limitations in their compositional reasoning (Dziri et al., 2023), planning (Valmeekam et al., 2023), and length generalization (Xiao and Liu, 2025). These challenges motivate the integration of non-text modalities—often referred to as *grounding*—to enhance model capabilities. However, existing grounding approaches either explicitly depend on auxiliary modules at inference, or implicitly align encoder-only models that lack generative capacity. We introduce Refeeding State Embeddings aligned using Environmental Data (RESEED), a flexible framework to directly ground decoder-based LLMs in structured environment data, leveraging both implicit and explicit signals. We show that RESEED improves sample efficiency, length generalization, and compositional reasoning in long-horizon sequential tasks.

Modern LLMs are trained in three stages: unsupervised causal language modeling (CLM) (Radford et al., 2018), a supervised finetuning with CLM, and alignment via preference optimization (Ouyang et al., 2022). Throughout this process, models are exposed to *text data* and *human preference data*. While both have been instrumental to recent progress, they omit key elements required for human-like language understanding. Text offers linguistic structure and encodes world knowledge (Bisk et al., 2020), but abstracts away key spatial, temporal, and causal relationships between concepts (Bender and Koller, 2020). Moreover, text tends to omit self-evident information, resulting in reporting bias that negatively impacts language modeling (Grice, 1975; Paik et al., 2021). Human preference data, while useful for alignment, is both sparse—a single bit for sequences of text—and subjective—some humans may prefer a more grammatical output, while others a more factual output. *To this end, we posit that a third kind of data is required:* ***data from the environment***. Motivated by research outlining the necessary role a human's interaction with their environment plays in language understanding (Glenberg and Kaschak, 2002; Gallese and Lakoff, 2005), we hypothesize that structured environmental signals can improve language modeling. Environment data, which we define as sequences of states that capture how an environment changes, complements text and human preference data in four key ways: (1) it preserves spatial and temporal relations; (2) it is concrete and fully specified, avoiding abstraction and reporting bias; (3) it provides a dense and informative training signal; and (4) it is consistent and objective.

Existing grounding work has demonstrated the benefits of grounding in improving the reasoning capabilities of LLMs, with two main directions

emerging. The first direction augments the system with a separate external model (Yang et al., 2022; Liu et al., 2023; Zellers et al., 2021); these are used to generate explicit modality-aware outputs which are fed into an LLM. While these works provide insights into the value of non-text modalities, they are inherently limited by their external model and masks rather than addresses the lack of grounding in the underlying language model. This is an important distinction because more complex and abstracted concepts may be difficult to simulate, but may still require the foundational grounded components to correctly interpret. The second line of work used for grounding is the use of additional modalities during training to align internal representations (Hsu et al., 2022; Tan and Bansal, 2020; Tang et al., 2021). These provide a direct signal to language models to improve their alignment with the environment. However these works rely on implicitly improving internal alignment; at inference, there is no clear representation of the environment to leverage. Further, these works focus on encoder-only models. Notably, modern LLM architecture has favored decoder architectures as the ability to generate open-ended outputs vastly increases the range of tasks they can accomplish.

RESEED combines the strengths of implicit alignment and explicit representations by training an LLM to predict latent state representations, which are then refed to the LLM to guide the language generation in a way that reflects with the true state of the environment. This approach provides the foundation for a scalable, grounded language model that operates in a manner consistent with modern LLMs. To evaluate RESEED, we require datasets that have paired text–trajectory data for training, but can test language models on text-only tasks. As these requirements cannot be found in existing benchmarks, we introduce three sequential reasoning datasets focused on cardinal direction navigation (ABCDs), block stacking (CUBES), and household object interactions (HOUSE). These tasks span increasingly complex state and action spaces. Compared to a text-only baseline, RESEED yields substantial gains in generalization and sample efficiency in sequential reasoning tasks.

Our contributions are: 1) RESEED, a novel grounding mechanism for decoder LLMs; 2) three new sequential reasoning benchmarks; 3) empirical validation of RESEED, demonstrating improved sample efficiency and generalizability; and 4) ablations analyzing the components of RESEED.

2 Related Work

2.1 Grounding with External Models

A subset of existing systems enhance language-based reasoning by incorporating external modality-specific models. Wang et al. (2023) leverages CLIP (Radford et al., 2021) to retrieve relevant images which are used to improve question answering. Tang et al. (2023); Yang et al. (2022) remove the need for an image database by using text-to-image diffusion models, while Zhang et al. (2024) directly leverages CLIP's text-model embeddings. While images offer rich *spatial* information, they cannot properly capture *temporal* information, which is key to sequential reasoning. To address this, Liu et al. (2023) feeds outputs from a physics simulation engine into an LLM to improve physical reasoning. In all these approaches, language models are augmented with other modalities, rather than grounded to other modalities. We believe this distinction is critical, as we posit that grounded models can compositionally build on observed interactions, whereas augmented models face end-to-end training challenges and are constrained by the capacity of their external modules. PIGLeT (Zellers et al., 2021) partially addresses these issues by using a trainable action prediction module to reason about household tasks. However, PIGLeT requires access to the ground-truth start state and only performs single-step reasoning. In contrast, RESEED operates on text-alone and is designed for multi-step reasoning.

2.2 Grounding through Internal Alignment

A complementary line of work focuses on aligning an LLM's internal representations across text and auxiliary modalities. Like RESEED, these methods use additional cross-modal modules during training, which are then discarded. We refer to these as implicit internal alignment methods.

Certain approaches in this space use additional modalities to produce more relevant text data. Carta et al. (2023) adapts the BabyAI environment (Chevalier-Boisvert et al., 2019) to a text-based version, giving LLMs the ability to explore the environment in text. Xiang et al. (2023) generates goal-oriented and random exploration experiences in VirtualHome (Puig et al., 2018), and uses templates to create a home-navigation fine-tuning dataset. Li et al. (2023) create state annotations in TextWorld (Côté et al., 2019) and TRIP Storks

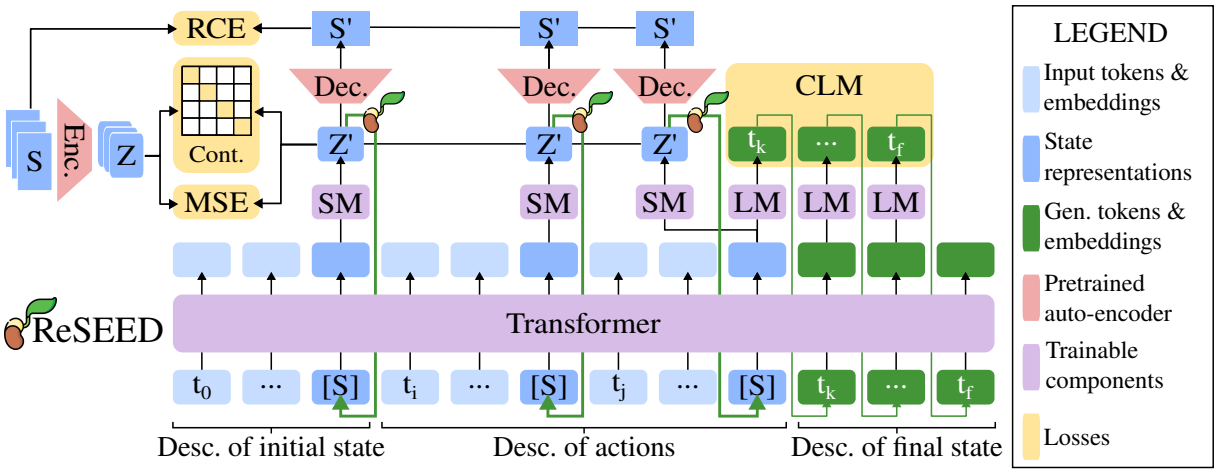


Figure 1: Architecture of RESEED. RESEED is comprised of a transformer with a language modeling head (LM) and a state modeling (SM) head (in purple). It leverages a pre-trained and frozen state auto-encoder (in red) during training. RESEED requires two forward passes. The first pass (in blue) encodes the special *[S]* input tokens and uses the output of these tokens to generate state representations Z'. In the second pass (in green), the special tokens are replaced with linear projections of Z', which are used to generate the description of the final state. The alignment of Z' is trained using a Reconstructive Cross-Entropy (RCE), a Contrastive (Cont.), and a Mean Squared-Error (MSE) loss (in yellow). A Causal Language modeling (CLM) is used to train the generation.

et al. (2021) to generate more coherent outputs. However, these methods remain limited by the abstraction and reporting bias inherent in text data.

Other approaches incorporate auxiliary losses conditioned on other modalities. Tan and Bansal (2020) adds a visual token (voken) classification objective in pre-training. Hsu et al. (2022) introduces a cross-modal adaptation phase with joint MLM, voken classification, and image-text matching. Most similar to our approach, Tang et al. (2021) train a teacher model using MLM and a contrastive cosine similarity task between video and text embeddings and then distill this knowledge into a student model. Jin et al. (2022) combine the voken classification and distillation tasks to further improve results. However, these methods are all designed for encoder-only architectures, which are not well-suited for text generation. In contrast, RESEED is developed for generative decoder-based models. More importantly, we identified that during implicit internal alignment, RESEED was producing embeddings that were aligned with the state of the environment, and that these could be effectively re-used rather than being discarded.

3 Method

Our method, Refeeding State Embeddings aligned using Environmental Data (RESEED), is depicted in Fig. 1. It can be broken down into three stages: 1) pre-training a state auto-encoder (Section 3.2), 2) generating latent state representations using special tokens (Section 3.3) 3) re-feeding these tokens before generating the output (Section 3.4).

3.1 Prerequisites

RESEED requires access to paired text-trajectory data. Specifically, for a given sequence of states ($s \in \{s_0, s_1, ..., s_f\}$), there should be a text description of the initial state ($d_0 \leftrightsquigarrow s_0$), a description of actions applied ($d_i \leftrightsquigarrow \Delta(s_{i-1}, s_i)$), and a description of the final state ($d_f \leftrightsquigarrow s_f$). In Section 3.5 we outline the datasets we use.

3.2 State Auto-Encoder

To create salient latent representations of the states, Z in our environment, we first train an auto-encoder (AE) using a reconstruction loss. Our AE is comprised of a 3-layer encoder multi-layer perceptron (MLP) and a 3-layer decoder MLP, both with dropout and trained using a cross-entropy reconstruction loss. The size of the latent representations is a hyperparameter h_{dim}, which we sweep $h_{dim} \in \{16, 64, 128, 256, 512\}$ for each dataset. We freeze the parameters of the AE when training the LLM and discard it after training is completed.

3.3 Generating Latent Representations

Our grounded language model adopts the convention of modern LLMs as a causal transformer. Given a description of the initial state and a sequence of actions, the model is trained to infer its

own latent representation of the resulting states, denoted as Z', which should align with the true latent states Z. To enable this, we inject a special token *[S]* after each input description d_i. The corresponding output embedding is passed through a single-layer state modeling head, projecting it to h_{dim}. We additionally pass the produced latent state through the pre-trained decoder to produce a prediction of the full state, $S' = Dec(Z')$.

To guide alignment, we apply three complementary losses: a contrastive (Cont.) loss (Oord et al., 2018) between Z' and Z, a mean-squared error (MSE) loss between Z' and Z, and reconstruction cross-entropy (RCE) loss between S' and S.

$$\mathcal{L}_{Cont.} = \mathbb{E}_i \left[-\log \frac{\exp(\text{sim}(Z'_i, Z_i)/\tau)}{\sum_{j=1}^{N} \exp(\text{sim}(Z'_i, Z_j)/\tau)} \right]$$

$$\mathcal{L}_{MSE} = \mathbb{E}_i \left[\|Z'_i - Z_i\|_2^2 \right]$$

$$\mathcal{L}_{RCE} = \mathbb{E}_i \left[-\sum_m S_i^{(m)} \log \left(S_i'^{(m)} \right) \right]$$

where $N = \sum_{k=1}^{B} |S_k|$, B is the batch size and $|S_k|$ is the number of states in sequence k, i.e., we use in-batch and in-sequence negatives in our contrastive loss.[1] A comparison of the impact of each loss is shown in Table 1.

3.4 Refeeding Embeddings

Sections 3.2 and 3.3 produce an LLM that is implicitly aligned and capable of generating salient latent representations of states. Motivated by the idea that these latent representations carry useful information about the environment, we develop a *refeeding* mechanism, in which a second forward pass is performed with the special *[S]* tokens being replaced with linear projections of Z'. This enables the model to explicitly condition its generation on its own representation of the environment. On this second pass we apply the traditional causal language modeling loss on the final state description:

$$\mathcal{L}_{CLM} = -\sum_{t=k}^{T} \log P(x_t \mid x_{<t})$$

where k indexes of the first token of the final state description and T is the total number of tokens.

We note here three clear differences with the most related work of VidLanKD (Tang et al., 2021). The first is the use of a causal language modeling which enables text generation. Second, unlike VidLanKD that uses a single embedding to encode the entire sequence, we leverage separate embeddings for each timestep in the sequence. This provides two benefits: 1) it allows the LLM to align itself multiple times per sequence, providing a denser learning signal, and 2) it provides more useful negatives in the contrastive loss as the model has to identify the impact of the actions to be able to differentiate different states from the same sequence. Without these more difficult negatives, the model may be able to rely on more surface level features—e.g., the objects in the scene—to differentiate embeddings and lose the specificity required for successful grounding. Third, instead of relying solely on implicit internal alignment, the refeeding provides an explicit mechanism to make use of our aligned representations. We report the impact using multiple state representations and explicit refeeding in Tables 2 and 3, respectively.

3.5 Datasets

RESEED is designed to leverage the rich information found in environments during training, while relying solely on text during inference. Naturally, this requires datasets that provide paired natural language and trajectory data for training, along with language-only evaluation sets. While prior work in natural language task specification—such as Mees et al. (2022); Zeng et al. (2020); Collaboration et al. (2024)—offers partially aligned training data, their evaluation protocols remain grounded in agent-based execution, lacking the necessary text-only test conditions. To bridge this gap, we introduce three new datasets that span distinct domains: cardinal direction navigation, block stacking, and interaction with common household objects. We describe each in detail below, and an example question and trajectory of each is shown in Fig. 2.

ABCDs: Asking 'Bout Cardinal Directions The first domain requires an model to understand navigation of cardinal directions. In ABCDs, an agent starts facing one of the four cardinal directions—{North, East, South, West}—then the agent performs a sequences of turns, and is then asked which direction it is facing at the completion of all turns. To create the text component of the dataset we use a template and create a mapping between a set of natural language action phrases and the equivalent base action. For example, the action phrase "turn 270 degrees clockwise" would map to the action turn left. We can then combine a descrip-

[1] i, j, k are overloaded and used as general indexing terms.

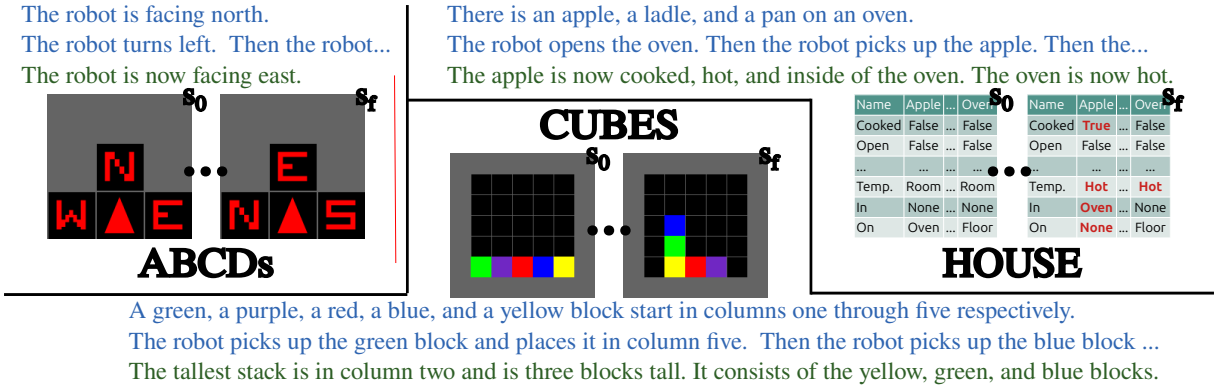


Figure 2: A sample from the ABCDs, CUBES, and HOUSE datasets. The blue text defines the initial state (s_o) and the actions performed (truncated for space). The orange text defines the final state (s_f). The model is also provided with access to intermediate states, which are collapsed into ellipses in the figure due to space.

tion of the start state with a sequence of A action phrases, and a description of the end state. For the trajectories, we create a small grid environment using gym-minigrid (Chevalier-Boisvert et al., 2023) with compass-style markers in the walls to indicate the direction and use an egocentric grid representation to encode each state.

For training and validation, we use up to $A \leq 5$ action phrases. To evaluate length generalization, we construct five evaluation sets, each containing 2000 samples and using a fixed number of action phrases, with $A \in 6, 7, 8, 9, 10$. We report exact match accuracy on each of these held-out sets.

This dataset provides a test bed where the state and action spaces are small, with only four different observations and underlying actions. This leads to a domain where the syntax of the language is similar and difficult to distinguish, but the semantics contained within the trajectory are clear and easily distinguishable.

Comprehensive Understanding of Block-stacking and Effects of Sequences (CUBES) CUBES tests a models ability to identify the tallest stack of blocks after a sequence of stacking actions. An initial state is presented with five different-colored blocks in a random order. A series of A stacks are then performed. Similarly to ABCDs, we use templates for the language component and gym-minigrid to create paired state trajectories. Unlike ABCDs, we use a full view of the all the blocks.

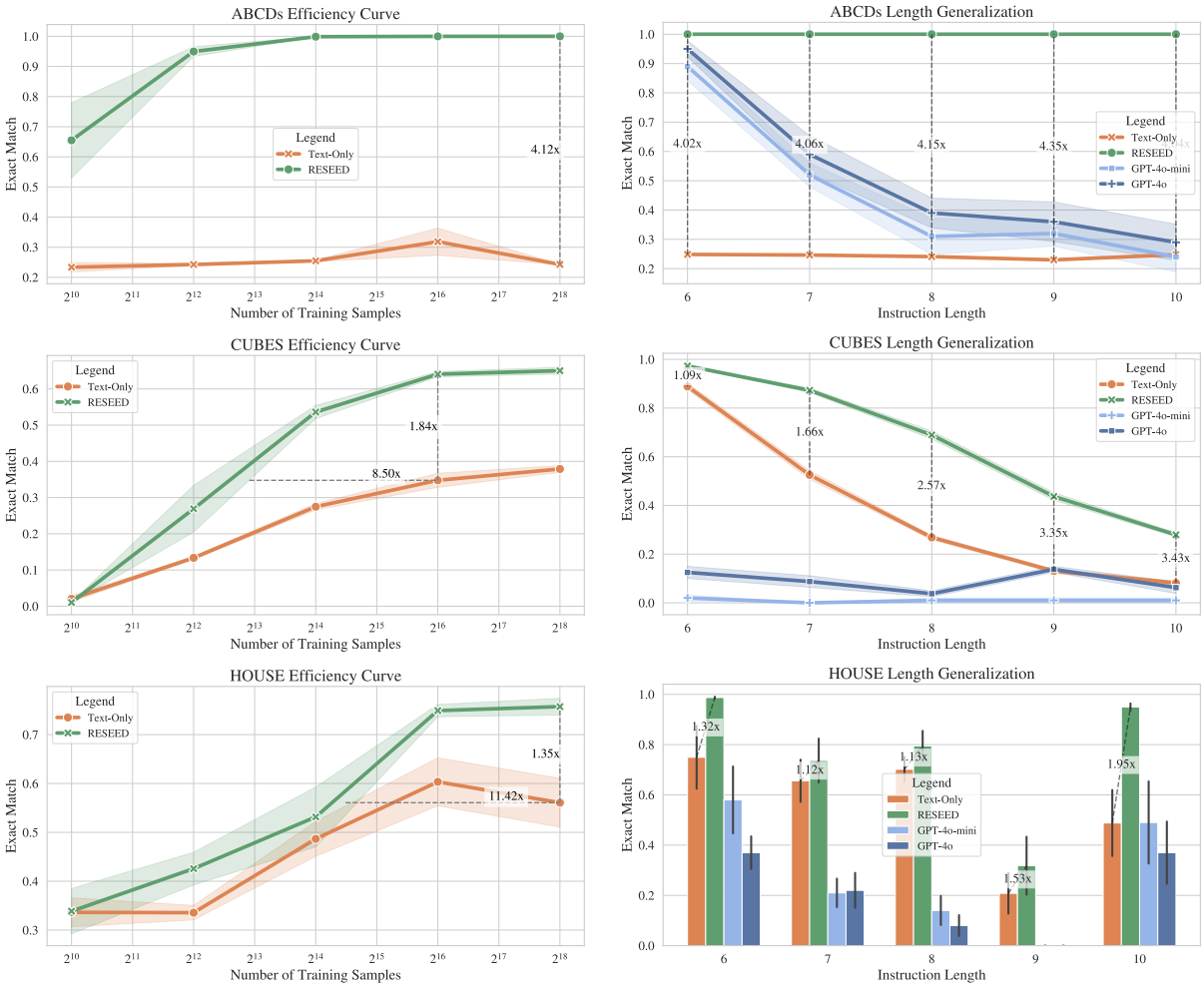
We match the setup for ABCD, with $A \leq 5$ action phrases for training and validation, and five length generalization sets which use $6 \leq A \leq 10$. We report the exact match accuracy on these length generalization sets.

Compared to ABCDs, the state space and action spaces are significantly larger, however the language still only requires a small vocabulary. The syntax of the language is still difficult to distinguish, however the semantics contained within the trajectory are less distinguishable than in ABCDs.

HOUSE: Household Object Use in Sequential Execution HOUSE is inspired by the PigPen dataset used in the Piglet framework (Zellers et al., 2021). In this dataset, a series of tasks are carried out using 100 common household objects with varying affordances. HOUSE consists of 9 atomic actions (e.g., pick up object, toggle object on, ...), which we compose into 10 low-level tasks (e.g., put X in Y, heat X, ...) and 10 high-level tasks (e.g., 'brew tea', 'water plants', ...). The low-level tasks are comprised of 2-5 atomic actions, while the high-level tasks are themselves composed of 2-3 low level tasks with a total of 6-10 atomic actions. Each task uses up to four objects, and the state space is defined as the state of the four objects, including the object name and the current features of the objects. A full description of the dataset, including a comparison to PigPen, the set of atomic actions, low-level tasks, and high-level tasks is outlined in Appendix A.

Mirroring ABCDs and CUBES, we train and validate using the low-level tasks, which include sequences of $2 \leq A \leq 5$, and evaluate the LLMs on the high-level tasks. There are two high-level tasks for each $A \in 6, 7, 8, 9, 10$, and we use 1000 samples per high-level task. We report the exact match accuracy on the high-level task sets.

HOUSE provides a step toward more general tasks that includes a wide range of objects and actions. Compared to ABCD and CUBES, the vo-



(a) Sample efficiency evaluation

(b) Length generalization evaluation

Figure 3: Sample efficiency and length generalization results on the three benchmarks.

cabulary, action space, and state space are all larger, which increases difficulty. However, the syntatic variation is also larger, making the impact of actions more apparent. Lastly, whereas ABCDs and CUBES evaluate length generalization by repeatedly applying the same kind of actions, HOUSE's evaluations requires compositionally applying observed sequences, which is an additional challenge.

3.6 Experimental Setup

The baseline for our experiments is a text-only (*TO*) model that shares the same underlying transformer and language modeling head and only differs in its lack of a state modeling head and projection layer. With an h_{dim} of 256 (the largest used across our datasets), RESEED only uses 0.4% more parameters (83.5M vs 83.9M parameters). The *TO* model is trained using the standard causal language modeling loss. For both RESEED and *TO*, we initialize the transformer and language modeling head using HuggingFace's (Wolf et al., 2020) pretrained gpt2-base (Radford et al., 2019). The state modeling head is randomly initialized. We then finetune the model on the datasets until convergence is reached on a validation set that is 12.5% ($1/8^{th}$) the size of the training set. We define convergence as having no improvement for more than 5 epochs. To enable evaluation context lengths that are longer than those seen in the training dataset, we freeze the positions ids of the pre-trained gpt2. We train the models using an AdamW optimizer (Loshchilov and Hutter, 2019) with an exponential learning rate decay. We tuned the *TO* model using a grid search on the learning rate, batch size, decay rate, and warm up steps, and used the same parameters for RESEED. We run each experiment using five random seeds, and report the standard error across seeds. Additional details can be found in Appendix D. We will open source all code at the end of the anonymity period.

4 Research Questions

RQ1: Does the grounding provided by* RESEED *improve sample efficiency? As RESEED has access to additional rich and unabstracted information during training in the form of environment data, we hypothesize that RESEED will be more sample efficient than an LLM trained solely on text, *TO*. To test this, for each dataset, we generate six different training sets, with number of samples $\in \{1024, 4096, 16384, 65536, 262144\}$ respectively.

RQ2: Does the grounding provided by* RESEED *improve length generalization? RESEED's generation of latent state representations, Z', enables it to produce estimates of the true states at regular intervals before decoding the final state, which we hypothesize will allow it to maintain a more consistent interpretation of the environment even in longer time horizons. To test this, we compare the results of RESEED and *TO* on the different length generalization evaluation sets we created when trained on the full 2^{18} samples.

RQ3: Which alignment signal—contrastive, reconstructive, or mean square error—best grounds language? A crucial step in our process is the alignment of the latent state representations produced by the LLM with latent state produced from our auto-encoder. To this end, we compare RESEED (RS for short), which uses all three losses, to variations that use one of the three alignment losses (RS_{Cont}, RS_{MSE}, RS_{RCE}), and a variation which uses no alignment loss (RS_{None}). This last variation is equivalent to providing the refeeding mechanism to the *TO* baseline.

RQ4: Does providing alignment at each state improve grounding? One of the core differences with Tang et al. (2021) is the alignment of our model at each state compared to a single alignment per text/state sequence pair. To understand the impact of this difference, we generate three variations of each dataset. The first uses the existing setup, where *[S]* tokens are added for each state in the ground-truth trajectory. The second uses a *[S]* token only for the first and last state in the true trajectory. The third, only uses a *[S]* token for the last state; this final variation most closely resembles the token setup in Tang et al. (2021), albeit for a decoder transformer.

RQ5: How beneficial is explicit refeeding compared to implicit alignment? A second core difference with Tang et al. (2021) is our method of explicitly refeeding the aligned representations before decoding. To ablate the benefits of explicit refeeding, we compare RESEED with an implicitly aligned version that is trained using all the same losses, but only performs a single forward pass.

5 Results & Discussion

RQ1: Does the grounding provided by* RESEED *improve sample efficiency? The graphs on the left side of Fig. 3 show the average results across all evaluation splits of the text-only baseline (in orange) and of RESEED (in green). While there is a small benefit when using a small amount of data, the benefit continues to grow larger after this point. Notably, once a minimum amount of data is reached, RESEED is able to leverage the environment data to improve upon text-only training. This leads to RESEED scaling better than than text-only training, which is an extremely promising result.

RQ2: Does the grounding provided by* RESEED *improve length generalization? The graphs on the right side of Fig. 3 show the results on each of the evaluation splits when using all training samples (2^{18}) for both the text-only baseline (in orange) and of RESEED (in green). Once again, we see RESEED outperforming the text-only baseline on every evaluation split. In addition to instruction length, the range of the underlying atomic actions and low-level tasks in the HOUSE dataset directly impact the complexity of the high-level tasks, adding an additional dimension to the evaluation splits. For this reason, we present the results in a bar chart rather than the line chart used in ABCDs and CUBES. The varying complexity of actions also explains the noisier trends seen in the HOUSE dataset results.

Model	ABCDs	CUBES	HOUSE
Text-Only	24.3 ± 0.1	37.9 ± 0.9	56.1 ± 5.0
RS_{None}	12.1 ± 4.0	39.4 ± 1.5	52.7 ± 6.1
RS_{Cont}	10.8 ± 4.3	**66.5** ± 1.1	68.2 ± 2.4
RS_{MSE}	99.7 ± 0.3	33.4 ± 1.3	60.7 ± 4.1
RS_{RCE}	81.0 ± 19.0	59.7 ± 1.8	68.2 ± 3.1
RS_{All3}	**100.0** ± 0.0	65.0 ± 0.9	**75.7** ± 1.7

Table 1: Comparison of alignment losses used in RESEED (RS). All3 indicates a combination of all three alignment losses. Results are the avg. accuracy and std. error across 5 seeds.

RQ3: Which alignment signal—contrastive, reconstructive, or mean square error—best grounds language? From Table 1, we see that alignment is necessary; RS_{None} performs similarly to the *TO* model. Interestingly, the alignment signal that is

the most beneficial varies per dataset, but using all alignment signals, $\texttt{RS}_{All3}$, provides competitive results in all three datasets. As such, this is the setup we use for all other experiments.

Model	ABCDs	CUBES	HOUSE
$\texttt{TO}_{\texttt{Final}}$	24.6 ± 0.2	37.3 ± 0.9	69.9 ± 1.1
$\texttt{TO}_{\texttt{Init\&Final}}$	24.5 ± 0.1	36.6 ± 1.1	68.7 ± 1.1
$\texttt{TO}_{\texttt{Per Phrase}}$	24.3 ± 0.1	37.9 ± 0.9	56.1 ± 5.0
$\texttt{RS}_{\texttt{Final}}$	24.2 ± 0.0	61.9 ± 1.1	68.1 ± 2.9
$\texttt{RS}_{\texttt{Init\&Final}}$	33.0 ± 8.6	62.7 ± 1.9	71.9 ± 3.4
$\texttt{RS}_{\texttt{Per Phrase}}$	**100.0** ± 0.0	**65.0** ± 0.9	**75.7** ± 1.7

Table 2: Comparison of RESEED (RS) and a text-only (TO) baseline with varying [S] token frequencies. Results are the average accuracy and standard error across 5 seeds.

RQ4: Does providing alignment at each state improve grounding? From Table 2, for the text-only baseline, including additional *[S]* tokens either has minimal impact, or, in the case of HOUSE, is deteriorates performance. In this latter case, we hypothesize the additional token(s) can be used by the model to further overfit to the training data. In contrast, for RESEED, we see a clear trend of improvement when including additional state representations, with $\texttt{RS}_{\texttt{Per Phrase}}$ providing the best result and lowest standard error in each dataset.

Model	ABCDs	CUBES	HOUSE
Text-Only	24.3 ± 0.1	37.9 ± 0.9	56.1 ± 5.0
$\texttt{RS}_{\texttt{Implicit}}$	24.5 ± 0.1	34.3 ± 3.1	59.1 ± 4.3
$\texttt{RS}_{\texttt{Explicit}}$	**100.0** ± 0.0	**65.0** ± 0.9	**75.7** ± 1.7

Table 3: Comparison of RESEED with and without explicit refeeding. Results are the average accuracy and standard error across 5 seeds.

RQ5: How beneficial is explicit refeeding compared to implicit alignment? Table 3 demonstrates that explicitly refeeding the learned representations is core to the performance of RESEED. Unlike prior work, implicit alignment provides little to no benefit in our experiments. As the system was tuned for explicit refeeding, it is possible that implicit alignment could be improved if different subsets of losses or hyper parameters are used, or if additional methods, such as Tang et al. (2021)'s teacher-student distillation, are integrated. However, given the more direct signal it provides and the results in Table 3, we believe explicit refeeding is a stronger mechanism to ground language. We note that refeeding does comes at the cost of a second forward pass, increasing compute and training time. However, this is a relatively small cost for improved generalizability of the model.

5.1 Comparison to State of the Art

The primary motivation of this paper was the inherent limitations of text-only training. To this end, we evaluate several State-of-the-Art LLMs from the Qwen2.5 (Yang et al., 2024) and GPT4o (OpenAI, 2023) family on our benchmarks. In each instance, we provide a prompt describing the task and provide 10 in-context examples. The full prompt can be found in Appendix B. We report the results in Table 4. The results are in line with other work (Dziri et al., 2023; Valmeekam et al., 2023), which demonstrate that current text-only LLMs struggle on tasks involving multi-step reasoning. Notably RESEED outperforms every model on every dataset, while being orders of magnitudes smaller.

Model	Size	ABCDs	CUBES	HOUSE
RESEED	84M	**100.0** ± 0.0	**65.0** ± 0.9	**75.7** ± 1.7
Qwen2.5	0.5B	31.4 ± 1.0	0.2 ± 0.2	1.8 ± 0.6
Qwen2.5	3B	42.8 ± 1.6	0.6 ± 0.2	4.0 ± 0.5
Qwen2.5	7B	40.0 ± 2.2	0.4 ± 0.4	26.4 ± 0.4
GPT4o	mini	45.6 ± 2.1	1.0 ± 0.5	28.4 ± 1.8
GPT4o		51.6 ± 2.5	9.0 ± 0.7	20.8 ± 1.4

Table 4: Comparison of RESEED (RS) to modern LLMs. Modern LLMs are provided 10 in-context examples and are evaluated on a subset of 100 examples divided evenly across evaluation splits. Results are the average accuracy and standard error across 5 seeds.

6 Conclusion

In this paper, we present a novel grounding mechanism, RESEED, which produces and then refeeds latent states embeddings to improve the sequential reasoning of LLMs. We then evaluate RESEED and a text-only baseline on three sequential reasoning benchmarks that we developed—ABCDs, CUBES, and HOUSE—and demonstrate that RESEED not only substantially improves the ability of LLMs to generalize to longer instruction lengths, but also scales better than text-only training. These results underscore the importance of grounding language in structured environmental feedback. However, progress in this area is currently limited by the scarcity of high-quality, paired text-trajectory datasets. To accelerate advances in grounded reasoning and robust generalization, we call on the community to prioritize the creation and open dissemination of diverse, richly annotated text-trajectory datasets. Such resources will become critical for training models that can reason over actions, states, and sequences in ways that align more closely with real-world dynamics.

Limitations

RESEED faces two primary limitations.

First, it introduces additional computational overhead due to the need for two forward passes. This cost is most significant during training, as the longer gradient path requires more memory and the addtional forward pass increase the time taken to complete one epoch. At inference time, no gradients are used used and iterative generation is already standard. To mitigate memory requirements, we explored a two-stage optimization procedure: one forward and backward pass to align latent states, followed by a second separate forward and backward pass to train generation. As shown in Appendix C, this approach still outperforms the baseline and only slightly underperforms the one-stage procedure, making a viable alternative is memory constraints exist.

Second, RESEED requires access to paired text-trajectory data for training. While this limits applicability in domains lacking such resources, our results demonstrate the substantial value of this supervision signal. We hope this work encourages the development of more diverse and scalable text-trajectory datasets, and we view this as a necessary step for progress in grounded language understanding.

Finally, we note an additional limitation of our evaluation benchmarks, which while diverse in structure, are still limited in scope. All three are deterministic, template-based, and operate in relatively constrained state and action spaces. In contrast, real-world environments often involve ambiguity, stochasticity, and varied linguistic expression. Moreover, even our largest benchmark contains only 2^{18} examples—small relative to modern pre-training corpora. Extending RESEED to broader, more complex, and non-deterministic domains is an important direction for future work. Doing so, however, will require scaling up dataset creation efforts accordingly.

References

Anthropic. 2024. Introducing the next generation of claude.

Emily M. Bender and Alexander Koller. 2020. Climbing towards NLU: On meaning, form, and understanding in the age of data. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 5185–5198, Online. Association for Computational Linguistics.

Yonatan Bisk, Ari Holtzman, Jesse Thomason, Jacob Andreas, Yoshua Bengio, Joyce Chai, Mirella Lapata, Angeliki Lazaridou, Jonathan May, Aleksandr Nisnevich, Nicolas Pinto, and Joseph Turian. 2020. Experience grounds language. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 8718–8735, Online. Association for Computational Linguistics.

Thomas Carta, Clément Romac, Thomas Wolf, Sylvain Lamprier, Olivier Sigaud, and Pierre-Yves Oudeyer. 2023. Grounding large language models in interactive environments with online reinforcement learning. CoRR.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2019. BabyAI: First steps towards grounded language learning with a human in the loop. In International Conference on Learning Representations.

Maxime Chevalier-Boisvert, Bolun Dai, Mark Towers, Rodrigo Perez-Vicente, Lucas Willems, Salem Lahlou, Suman Pal, Pablo Samuel Castro, and Jordan Terry. 2023. Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks. In Advances in Neural Information Processing Systems 36, New Orleans, LA, USA.

Embodiment Collaboration, Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, and 262 others. 2024. Open x-embodiment: Robotic learning datasets and rt-x models. Preprint, arXiv:2310.08864.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. Textworld: A learning environment for text-based games. In Computer Games, pages 41–75, Cham. Springer International Publishing.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, Jena D. Hwang, Soumya Sanyal, Xiang Ren, Allyson Ettinger, Zaid Harchaoui, and Yejin Choi. 2023. Faith and fate: Limits of transformers on compositionality. In Thirty-seventh Conference on Neural Information Processing Systems.

Vittorio Gallese and George Lakoff. 2005. The brain's concepts: The role of the sensory-motor system in conceptual knowledge. Cognitive neuropsychology, 22(3-4):455–479.

Arthur M Glenberg and Michael P Kaschak. 2002. Grounding language in action. Psychonomic bulletin & review, 9(3):558–565.

Herbert Paul Grice. 1975. Logic and conversation. Syntax and semantics, 3:43–58.

Chan-Jan Hsu, Hung-yi Lee, and Yu Tsao. 2022. XDBERT: Distilling visual information to BERT from cross-modal systems to improve language understanding. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), pages 479–489, Dublin, Ireland. Association for Computational Linguistics.

Woojeong Jin, Dong-Ho Lee, Chenguang Zhu, Jay Pujara, and Xiang Ren. 2022. Leveraging visual knowledge in language tasks: An empirical study on intermediate pre-training for cross-modal knowledge transfer. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2750–2762, Dublin, Ireland. Association for Computational Linguistics.

Belinda Z. Li, Maxwell Nye, and Jacob Andreas. 2023. Language modeling with latent situations. In Findings of the Association for Computational Linguistics: ACL 2023, pages 12556–12571, Toronto, Canada. Association for Computational Linguistics.

Ruibo Liu, Jason Wei, Shixiang Shane Gu, Te-Yen Wu, Soroush Vosoughi, Claire Cui, Denny Zhou, and Andrew M. Dai. 2023. Mind's eye: Grounded language model reasoning through simulation. In The Eleventh International Conference on Learning Representations.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In International Conference on Learning Representations.

Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. 2022. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. IEEE Robotics and Automation Letters (RA-L), 7(3):7327–7334.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. arXiv preprint arXiv:1807.03748.

OpenAI. 2023. Gpt-4 technical report. arXiv preprint arXiv:2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Gray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In Advances in Neural Information Processing Systems.

Cory Paik, Stéphane Aroca-Ouellette, Alessandro Roncone, and Katharina Kann. 2021. The World of an Octopus: How Reporting Bias Influences a Language Model's Perception of Color. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 823–835, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. 2018. Virtualhome: Simulating household activities via programs. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 8494–8502.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 8748–8763. PMLR.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. Available at https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Shane Storks, Qiaozi Gao, Yichi Zhang, and Joyce Chai. 2021. Tiered reasoning for intuitive physics: Toward verifiable commonsense language understanding. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 4902–4918, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hao Tan and Mohit Bansal. 2020. Vokenization: Improving language understanding with contextualized, visual-grounded supervision. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2066–2080, Online. Association for Computational Linguistics.

Tianyi Tang, Yushuo Chen, Yifan Du, Junyi Li, Wayne Xin Zhao, and Ji-Rong Wen. 2023. Learning to imagine: Visually-augmented natural language generation. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 9468–9481, Toronto, Canada. Association for Computational Linguistics.

Zineng Tang, Jaemin Cho, Hao Tan, and Mohit Bansal. 2021. VidlanKD: Improving language understanding via video-distilled knowledge transfer. In Advances in Neural Information Processing Systems.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023. On the planning abilities of large language models - a critical investigation. In Thirty-seventh Conference on Neural Information Processing Systems.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. 2024. Position: will we run out of data? limits of llm scaling based on human-generated data. In Proceedings of the 41st International Conference on Machine Learning, ICML'24. JMLR.org.

Weizhi Wang, Li Dong, Hao Cheng, Haoyu Song, Xiaodong Liu, Xifeng Yan, Jianfeng Gao, and Furu Wei. 2023. Visually-augmented language modeling. In The Eleventh International Conference on Learning Representations.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, and 3 others. 2020. Transformers: State-of-the-art natural language processing. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online. Association for Computational Linguistics.

Jiannan Xiang, Tianhua Tao, Yi Gu, Tianmin Shu, Zirui Wang, Zichao Yang, and Zhiting Hu. 2023. Language models meet world models: Embodied experiences enhance language models. In Thirty-seventh Conference on Neural Information Processing Systems.

Changnan Xiao and Bing Liu. 2025. Generalizing reasoning problems to longer lengths. In The Thirteenth International Conference on Learning Representations.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, and 22 others. 2024. Qwen2.5 technical report. arXiv preprint arXiv:2412.15115.

Yue Yang, Wenlin Yao, Hongming Zhang, Xiaoyang Wang, Dong Yu, and Jianshu Chen. 2022. Z-LaVI: Zero-shot language solver fueled by visual imagination. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 1186–1203, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. PIGLeT: Language grounding through neuro-symbolic interaction in a 3D world. 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 2040–2050, Online. Association for Computational Linguistics.

Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. 2020. Transporter networks: Rearranging the visual world for robotic manipulation. Conference on Robot Learning (CoRL).

Xinyun Zhang, Haochen Tan, Han Wu, and Bei Yu. 2024. Towards versatile and efficient visual knowledge integration into pre-trained language models with cross-modal adapters. Preprint, arXiv:2305.07358.

A HOUSE details

Atomic Actions	Low-Level Tasks	High-Level Tasks
PickupObject	put_X_on_Y	stack_3
PutObject	put_X_in_Y	stack_4
OpenObject	heat_X	water_plants_using_X
CloseObject	fill_X	make_iced_coffee_in_X
ToggleObjectOn	brew_X	brew_tea_in_X
ToggleObjectOff	clean_X	toast_X
PourFromObject	slice_X	cook_X
SliceObject	pour_X_onin_Y	cook_and_remove_X
WipeObject	wipe_X_dry	clean_and_dry_X
	wipe_X_clean	clean_large_X

Table 5: List of atomic actions, low-level tasks, and high-level tasks.

State Feature	Related Affordance
ObjectName (100)	–
isWet (2)	wettable
isCooked (2)	cookable
isClean (2)	cleanable
isFilledWithLiquid (2)	fillable
isOpen (2)	openable
isPickedUp (2)	pickupable
isSliced (2)	sliceable
isToggled (2)	toggleable
objectTemperature (3)	canChangeTemp
mass_change (3)	fillable (indirectly)
parentReceptaclesOn (6)	receptacleOn
parentReceptaclesIn (6)	receptacleIn

Table 6: Mapping of state features to their related affordances. Number in (parentheses) denote the range of values the feature can take on.

HOUSE is a dataset inspired by the PigPen dataset used in Zellers et al. (2021). We made the decision to adapt PigPen, rather than using the original dataset, for three primary reasons: 1) PigPen divides each full high-level task trajectory into a single (s_t, a_t, s_{t+1}) transition tuple, whereas we are interested in outcome of multiple sequential steps. 2) We found a range of inconsistencies and non-deterministic outcomes within the PigPen dataset (e.g. toast getting hot when turning ON the toaster in one instance, and the toast getting hot when turning OFF the toaster in one instance). 3) We wanted more control over the compositionality of tasks. A full list of actions and tasks is shown in Table 5.

To this end, we manually crafted a deterministic transition function for each low-level action based on the affordances of each object and used it to create trajectories our trajectories. Matching ABCDs and CUBES, we use templates to create the language description of the trajectory. A full list of state features (used to encode the state) and affordances (unchangeable properties of objects which are not visible, but effect the outcome of the transition function) are shown in Table 6.

B ICL prompt

Fig. 4 outlines the full prompt used for in context learning. <> denote placeholder values. All examples came from the same distribution as the training set. For HOUSE, we ensured that a representative example for each of the 10 low-level tasks were used.

```
system >>>
    You are tasked to solve sequential reasoning problems in which you will be given an initial
    state and a sequence of actions. Your job is to predict the final state after the sequence
    of actions is applied to the initial state. You will be given a list of examples that you
    can use to learn how to solve the problem. You must match the output format of the final
    state exactly. You will be graded on the exact accuracy of your predictions.

    Expected output format:
    <dataset specific formatting>.
    Where terms enclosed in <> should be replaced with the actual output values.

user >>>
    ---Example 1---
    Initial State and Actions:
    <example 1 initial state & actions>

assistant >>>
    Final State:
    <example 1 final state>

    ...

user >>>
    ---Example 10---
    Initial State and Actions:
    <example 10 initial state & actions>

assistant >>>
    Final State:
    <example 10 final state>

user >>>
    ---Problem---
    Initial State and Actions:
    <initial state and actions for problems to solve>
```

Figure 4: In-context learning prompt example.

C Two-Step Training

To explore a more computationally friendly approach, we test a variation of RESEED, named that fully separates the alignment step from the generation step. Specifically, on alternating batches, we either perform a forward and backwards pass using the alignment losses, or we perform a forward pass with no gradients to generate Z' and then use those for the forward pass with gradients which decodes the final state description. A comparison of results between these two approaches is shown in Table 7. While separating the alignment and generation steps does slightly reduce the performance of RESEED, it still outperforms the *TO* model, and does so with no additional memory requirements.

Model	ABCDs	CUBES	HOUSE
Text-Only	24.3 ± 0.1	37.9 ± 0.9	56.1 ± 5.0
$\text{RS}_{\text{Single Pass}}$	**100.0** ± 0.0	**65.0** ± 0.9	**75.7** ± 1.7
$\text{RS}_{\text{Separate Passes}}$	99.8 ± 0.2	64.6 ± 1.5	70.5 ± 2.9

Table 7: Comparison of RESEED (RS) with and without separate backward passes. Results are the average accuracy and standard error across 5 seeds.

D Additional Implementation Details

D.1 Hyper parameters

To tune the *TO*, we performed a grid search on the learning rate ($lr \in \{1e-5, 3e-5, 1e-4\}$, batch size ($bs \in \{32, 64, 128\}$), decay rate (per update step) ($dr \in \{0.9999, 0.99995, 0.99999\}$, and warm up steps ($ws \in \{400, 1000, 2000\}$). We found that across datasets, a batch size of 64 and 1000 warm up steps consistently provided the best results. For learning rate, we found $1e-4$ performed best on CUBES and HOUSE, while a learning rate of $3e-5$ performed best on ABCDs. For decary rate, we found 0.99995 performed best on CUBES and HOUSE, while a decay rate of 0.99999 performed best on ABCDs.

We used the same above hyper parameters for RESEED, only tuning $h_{dim} \in \{16, 32, 64, 128, 256, 512\}$ for each dataset. We found $h_{dim} = 16$, $h_{dim} = 128$, and $h_{dim} = 256$ performed best ABCDs, CUBES, and HOUSE respectively.

For the in-context learning LLMs used, all in-context learning examples came from the training set and were manually verified to be representative examples. The GPT-4o-mini checkpoint used was: gpt-4o-mini-2024-07-18, and the GPT-4o checkpoint used was gpt-4o-2024-08-06.

The five random seeds used were: [9590, 1282, 5742, 4674, 2921].

D.2 Computational Budget

All experiments were run using a single A100 (all experiments fit on a 40GB A100, although 80GB A100s were used as well). To reach convergence on a single run took between 10 minutes (1024 samples) and 16 hours (262144 samples). Compared to the *TO* model, RESEED took between 1.1x and 2x the amount of time to reach convergence. The additional cost is primarily due to the two forward passes, although on the ABCDs dataset, RESEED reached convergence much faster, mitigating the cost substantially. The experiments in this paper involved 20 runs per seed per dataset (with 5 seeds and 3 datasets), for a total of 300 runs.

E AI assistant use

Claude-3.7-Sonnet-Thinking (Anthropic, 2024) was used to develop small portions of the code base. GPT-4o (OpenAI, 2023) was used as to edit the text at a paragraph level. All code and writing output from AI assistants were manually verified and edited as necessary by a human before use.

F Additional Attributions and Attribution Info

The seed icon used in Fig. 1 is from Flaticon.com. All artifacts were used in a manner consistent with their intended use.