FOSSIL: Harnessing Feedback on Suboptimal Samples for Data-Efficient Generalisation with Imitation Learning for Embodied Vision-and-Language Tasks

Sabrina McCallum[1,2], Amit Parekh[2], Alessandro Suglia[1]
[1]University of Edinburgh
[2]Heriot-Watt University
{s.mccallum-exner, asuglia}@ed.ac.uk, a.parekh@hw.ac.uk

Abstract

Current approaches to embodied AI tend to learn policies from expert demonstrations. However, without a mechanism to evaluate the quality of demonstrated actions, they are limited to learning from optimal behaviour, or they risk replicating errors and inefficiencies. While reinforcement learning offers one alternative, the associated exploration typically results in sacrificing data efficiency. This work explores how agents trained with imitation learning can learn robust representations from both optimal and suboptimal demonstrations when given access to constructive language feedback as a means to contextualise different modes of behaviour. We directly provide language feedback embeddings as part of the input sequence into a Transformer-based policy, and optionally complement the traditional next action prediction objective with auxiliary self-supervised learning objectives for feedback prediction. We test our approach on a range of embodied Vision-and-Language tasks in our custom BABYAI-XGEN environment and show significant improvements in agents' compositional generalisation abilities and robustness, suggesting that our data-efficient method allows models to successfully convert suboptimal behaviour into learning opportunities. Overall, our results suggest that language feedback is a competitive and intuitive alternative to intermediate scalar rewards for language-specified embodied tasks.

1 Introduction

Embodied AI presents a significant advancement in artificial intelligence, emphasising the importance of physical embodiment and the resulting ability to interact directly with the environment to accomplish goals. By perceiving and reasoning about the effect of their actions on their surroundings, embodied AI systems gain access to rich learning signals, which enable more robust and context-aware models of the world (Deitke et al., 2022). For such systems to be useful across real-world deployment scenarios, they must be able to translate language instructions into meaningful sequences of actions. Embodied agents that understand language not only unlock an additional learning modality, but also enable richer, situational representations of behaviour by grounding language into additional sensory modalities such as vision. This facilitates the creation of language-guided pixel-based policies for Embodied AI (e.g., Team et al., 2024).

While imitation learning (IL) methods are commonly used to train such agents, it is typically assumed that the underlying demonstration data reflects optimal or near-optimal behaviour (Min et al., 2022). This presents three key limitations. First, models trained exclusively on optimal trajectories may learn that there is exactly one valid solution for a given task. Second, such models never encounter recoverable errors and their corrections. Third, IL lacks mechanisms to evaluate the quality of actions when faced with multiple valid behavioural modes for the same observation. In contrast, reinforcement learning (RL) algorithms explicitly incorporate feedback in the form of scalar rewards to distinguish between varied, more or less optimal behaviours (Levine et al., 2020; Sutton, 2018). However, the associated exploration in RL also means that it is typically less sample-efficient than IL. This is exacerbated for sparse rewards, which is typically the case in embodied AI.

In this work, we investigate to what extent Transformer-based IL policies can benefit from the inclusion of suboptimal demonstrations in the training data when the effect of actions is contextualised with constructive language feedback, turning mistakes into learning opportunities. We focus on this family of models due to their ability to better represent multimodal inputs and long input sequences that are common in Embodied AI tasks, and to align our work with the foundational architecture and training regimen used for current Vision-Language-Action (VLA) models in robotics (Black et al., 2024; Kim et al., 2024; Nvidia et al., 2025; Shukor et al., 2025). We test whether our

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 18077–18101
November 4-9, 2025

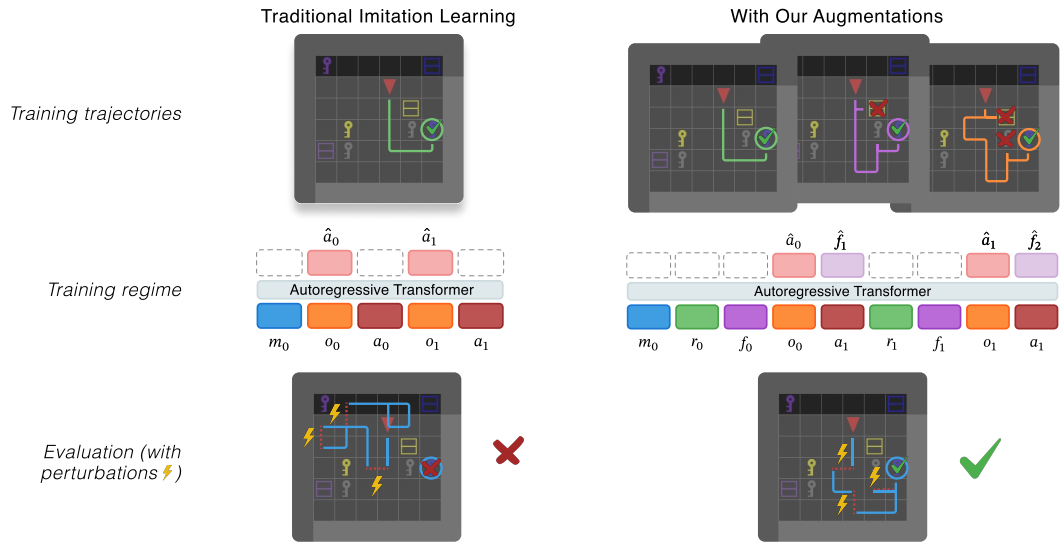


Figure 1: Our method leverages both optimal and suboptimal trajectories for a given task instance by contextualising modes of behaviour with feedback signals. We leverage different types of feedback and additional self-supervised auxiliary tasks to learn highly generalisable and robust representations of behaviour in a data-efficient manner.

approach can yield improved generalisation capabilities of language-guided IL policies, while simultaneously ensuring sample-efficient learning.

We introduce a training regime where we systematically augment the training data with suboptimal variations of the base demonstrations. The use of language feedback is intuitive in the light of possible real-world deployment scenarios for embodied AI, and mirrors the modality of language instructions. In addition, we test if language feedback is a competitive substitute or suitable complement for shaped scalar rewards when conditioning IL on feedback. We further test the efficacy of a self-supervised auxiliary task in the form of feedback prediction. The motivation for introducing the feedback prediction objectives is two-fold: 1. To encourage more robust representations of actions, and 2. To give the model the ability to tap into an internal world model of the consequences of actions when no feedback is provided at inference. To allow us to test fine-grained compositional generalisation capabilities systematically, we develop BabyAI-XGen, a modified version of the BabyAI (Chevalier-Boisvert et al., 2018) environment, bridging the gap between procedural generation and granular control over task configurations and environment parameters.

Our findings indicate that Transformer-based IL policies that are trained from scratch with language feedback-conditioned suboptimal demonstrations generalise significantly better in compositional tasks than a baseline trained only on optimal trajectories, by turning mistakes and inefficient strategies into learning opportunities. We find this effect is consistent across different amounts of training data. Notably, we observe that language feedback and scalar rewards provided with similar frequency yield comparable performance, offering practical flexibility depending on which signal is easier to provide. While combining language feedback and scalar rewards does not significantly improve task success rates, it does improve robustness to input perturbations during inference, indicating their complementary strengths. The same is observed for the auxiliary prediction task. Lastly, we present further evidence for the sample efficiency of our method by demonstrating that an online RL baseline trained with PPO (Schulman et al., 2017) and using the same reward function achieves near-random generalisation performance when trained on equivalent numbers of data points.

In summary, the contributions of this work are as follows:

- We present `FOSSIL` (**F**eedback on **S**uboptimal **S**amples in **I**mitation **L**earning), a framework that leverages feedback to unlock the learning potential in suboptimal demonstrations for IL.
- We release `BabyAI-XGen`, a modified version of BabyAI for the procedural generation of custom tasks for compositional generalisation.

- We define a range of evaluation settings to assess compositional generalisation, robustness to perturbations and data efficiency, and present results that suggest that language feedback and shaped scalar rewards can be equally effective.

2 Related work

Language-guided embodied AI. We consider our work in the context of language-guided embodied AI, specifically, mobile manipulation tasks. Compared with manipulation-only tasks (Jiang et al., 2023; Mees et al., 2022, etc.), and navigation-only tasks (Anderson et al., 2018), the combination of navigation and object interaction creates additional degrees of freedom for behaviour to deviate from the optimal solutions, and hence, more meaningful opportunities to provide feedback can arise. Existing datasets (Gao et al., 2022; Puig et al., 2018; Shridhar et al., 2019, 2020; Yenamandra et al., 2023) tend to annotate planner-generated trajectories with language instructions, which means that agents are trained to rely on optimal behaviour, while suboptimal behaviour is seen as something to mitigate, not leverage (Min et al., 2022).

Language as feedback. Previous work uses LLMs as judges (Pang et al., 2024; Wu et al., 2024), trains models to provide language feedback (Zhong et al., 2024) or leverages LLMs to increase the diversity of language feedback (Xi et al., 2024a) for a range of tasks. In contrast, we investigate *how to utilize language feedback systematically to contextualise different modes of behaviour*, rather than on the quality or generation mode of the feedback. While definitions of feedback in previous work range widely, it broadly falls into two categories: 1) general language hints (Lin et al., 2023) or foresight (Xi et al., 2024a)—which often takes the form of granular instructions—and 2) hindsight feedback (Xi et al., 2024a). We focus on a form of hindsight feedback. To reduce the number of confounding variables, we procedurally populate a multi-part template which reflects both a judgment of the action and, importantly, an explanation of the judgment. Our work is also in juxtaposition to previous work that derives a reward function from language, e.g. by computing the similarity between observations and subgoals described in language (Adeniji et al., 2023; Du et al., 2023). Instead, we use the language feedback embeddings directly as input into a Transformer-based (Vaswani et al., 2017) policy, following work by McCallum et al. (2023) and Xi et al. (2024a). While both are antecedents to our work, they only compare language feedback with rewards corresponding to the binary failure / success scenarios typically associated with embodied AI tasks, and disregard the interplay of the language feedback with the optimality of the trajectories in the training data, which consists either of demonstration datasets (Xi et al., 2024a) or randomly generated trajectories (McCallum et al., 2023). In contrast, we facilitate a fair comparison of language feedback and rewards by aligning their frequency, and systematically construct datasets composed of both optimal and suboptimal trajectories. Another important strength of our work is that, differently from Xi et al. (2024a), we focus on purely pixel-based policies rather than studying environments where agents have access to symbolic state representation [1].

Self-supervised auxiliary tasks for grounded representations. While predicting the observations and rewards resulting from actions is inherent to world models (Du et al., 2023; Hafner et al., 2019, 2020, 2023; Zhang et al., 2024), where this ability is commonly referred to as *latent imagination*, the utility of predicting tokens other than actions as self-supervised auxiliary tasks (Jaderberg et al., 2016) in IL is still under-explored. Results from work on other multi-modal tasks, such as multi-modal translation (Elliott and Kádár, 2017) or guessing games (Suglia et al., 2020), suggest that such training objectives are a promising avenue for learning well-grounded multi-modal representations. We adapt this approach for IL for embodied instruction following tasks.

Compositional generalisation in embodied AI. Previous work, such as VIMA (Jiang et al., 2023) for tabletop manipulation, predominantly investigates *Systematicity*, or the ability of models to systematically recombine known components and rules to novel combinations (Hupkes et al., 2019). However, other dimensions of compositional generalisation, such as *Productivity*, remain under-explored in the context of embodied agents and interactive environments. In lieu of existing suitable settings, we design a multi-dimensional framework to evaluate the compositional generalisation abilities of models trained with our method, and facilitate this with our BABYAI-XGEN environment.

[1] In Xi et al. (2024a)'s setup, only one environment over four uses pixel-based representations.

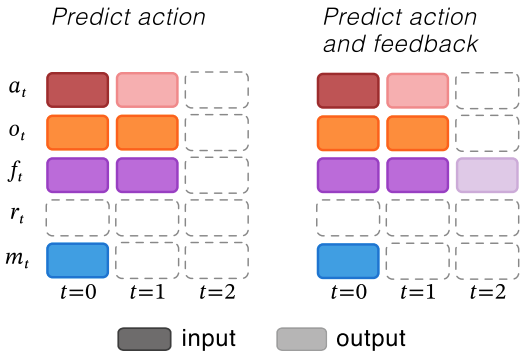


Figure 2: Input and output tokens for a model conditioning action generation on initial instructions and language feedback, with the option to predict language feedback at the next time step. m_i=instructions, f_i=language feedback, r_i=returns-to-go/rewards, o_i=observations, a_i=actions.

3 Method

Inspired by Decision Transformers (Chen et al., 2021) and `Uni[MASK]` (Carroll et al., 2022), we model different types of IL as sequential decision problems, with the option to condition action generation on relevant additional tokens corresponding to data elements that may accompany state-action pairs, such as instructions, scalar rewards and/or language feedback. All our models share the same base architecture, consisting of an autoregressive Transformer backbone with a simple action prediction head, with optional additional heads to predict tokens for auxiliary tasks. Figure 2 illustrates an example configuration using our flexible architecture. The token masks for all models used in our experiments and additional possible configurations can be found in Appendix B. We further deviate from the original Decision Transformer and previous work applying Decision Transformer to similar domains (Xi et al., 2024a) by using Llama2 (Touvron et al., 2023) as our reference backbone architecture[2] to facilitate faster training and inference, as well as support learning from entire trajectories, rather than sub-trajectories of limited context lengths. Language inputs are encoded using a frozen pretrained Sentence-BERT language model (Reimers and Gurevych, 2019), which allows us to condense sentences of arbitrary length into compact vector representations. Image observations are encoded into a single token using a simple CNN network trained concurrently with the policy. We provide full implementation details in Appendix B.

Training Objectives and Loss Functions. We follow previous work on Decision Transformers (Chen et al., 2021) and learn to predict actions in the environment as a next token prediction task by minimising cross-entropy loss of the actions. Beyond the standard cross-entropy loss for action prediction, we incorporate auxiliary self-supervised training objectives that involve predicting feedback signals at the next time step. Specifically, we use additional regression heads with MSE loss to predict scalar reward values and language feedback embeddings at t+1. These auxiliary losses are balanced with the primary action prediction loss through learnable weighting parameters. This multi-objective approach encourages the model to develop richer representations of action consequences, as it must anticipate not only appropriate actions but also the feedback those actions will generate, enabling better understanding of behaviour and context. Note that the predicted feedback tokens are used exclusively to compute auxiliary losses and are not fed back into the model as input. Implementation details for all losses and the loss balancing are provided in Appendix B.

4 Experimental framework

4.1 Evaluation settings

We refer to the framework proposed by Hupkes et al. (2023) and study generalisation in embodied Vision-and-Language tasks along two main axes: (1) *compositional generalisation* and (2) *robustness*. For compositional generalisation, we consider scenarios of Systematicity and Productivity. For robustness, we analyse the robustness of models' goal representations, their robustness to external perturbations and adversarial or missing feedback, and the efficiency of the obtained solutions. We further test whether the ability to generalise compositionally is associated with the ability to effectively leverage the available samples, and investigate generalisation performance at different proportions of the training data, comparing our method with the IL and an online RL baseline.

4.2 Environment: BabyAI-XGen

We develop BabyAI-XGen[3] to fill the gap in embodied environments capable of supporting rigorous compositional generalisation research and facilitate our evaluation settings. BabyAI-XGen is

[2]Since we override the configuration, we do not use Llama2's pretrained weights.

[3]We make the full environment code and all task configurations available at github.com/sabraaap/fossil.

the result of various key modifications to BabyAI's (Chevalier-Boisvert et al., 2018) backend, aimed at giving users control over a comprehensive range of environment parameters, allowing them to configure missions corresponding to distinct train and test scenarios. We outline all configuration parameters and their intended use in Appendix D.

We choose BabyAI as our starting point since, despite its apparent simplicity, it remains a popular and challenging testbed for embodied agents that understand and learn from natural language instructions (Gu et al., 2024; Huang et al., 2024; Lu et al., 2024, etc.). Out of a range of candidate grid world environments that support language (Lin et al., 2023; Wang and Narasimhan, 2021; Zholus et al., 2022), only BabyAI fulfils all desiderata required to facilitate our training and evaluation settings: partial observability, pixel observations, navigation with obstruction, multiple types of object interactions, the ability to chain together tasks, as well as objects which can be defined along several attribute axes. Furthermore, AgentBoard (Ma et al., 2024) and AgentGym (Xi et al., 2024b), two recent LLM agent benchmarks, include BabyAI and rate its difficulty on par with more visually realistic 3D environments such as ALFWorld (Shridhar et al., 2020). Note that both benchmarks target text-only models and rely on the symbolic observations, while we use the pixel observations of the partial, egocentric views for a more challenging setting that requires vision and language. Importantly, BabyAI is highly reproducible, extensible and scalable, as it does not depend on an external game or physics engines, which makes the procedural generation of training data lightweight and reliable.

Using BabyAI-XGen, we define new level configurations to directly isolate and study different types of compositional generalisation. We focus on variations of Pickup and PutNext tasks, since these require navigation and interaction. To make the setting even more challenging, we enforce a range of stricter settings, including shorter timeouts and a guarantee that missions for a given level consistently require the same skill set, such as understanding locations or temporal order in instructions, or navigating to a goal object located in a different room. This addresses a limitation associated with mission generation mechanism in the original BabyAI suite, which samples missions post-hoc based on the sampled objects, making it impossible to ensure that all missions for a given level strictly require the skill set attributed to that level. In contrast, we sample missions and construct the objects in the scene around them.

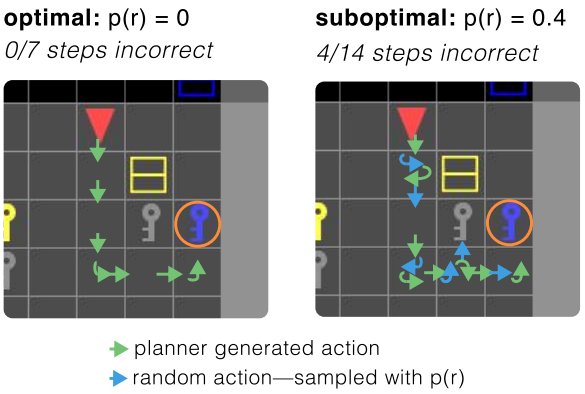


Figure 3: Optimal trajectories generated by a planner, and suboptimal trajectories obtained by replacing planner actions with random actions, which the planner must correct if necessary. The given `p(r)` is exemplary.

4.3 Training dataset generation

We generate two types of trajectory datasets for *each task* used in our multi-task training setup: one with ~12K optimal paths using BFS planning, and another composed of ~4K optimal and ~8K suboptimal paths with mistakes and inefficiencies[4]. For suboptimal trajectories, we uniformly sample from the action space and replace the planner action with probability p, where the subsequent planner action corrects any suboptimal behaviour. The process is depicted in Figure 3.

4.4 Feedback augmentation

We generate language feedback and scalar rewards according to the same underlying heuristics, by building on the rule-based feedback oracles proposed by McCallum et al. (2023). This approach ensures consistent, reliable and accurate feedback generation, which is essential for our controlled study of how different feedback modalities affect learning from suboptimal trajectories. Our rule-based feedback generation pipeline leverages privileged information about environment states and task objectives to provide comprehensive coverage of relevant feedback scenarios. This systematic approach allows us to isolate the effects of feedback modality and quality without introducing variability from potential inconsistencies in model-generated feedback (Xi et al., 2024a).

[4]All datasets and the code for generating trajectories are publicly available at huggingface.co/fossil-eai/datasets. and github.com/sabraaap/fossil, respectively.

Task feedback We extend the oracle for "task feedback" from McCallum et al. (2023) to return not only *positive* language feedback for *desired* interactions with *goal objects* that lead to the agent completing a subgoal, but additionally *negative* language feedback on *undesired* interactions with *distractor objects*, and increase diversity and informativeness of our template-based feedback following Xi et al. (2024a). The template, including representative examples, is illustrated in Appendix F.

Affordance feedback We adopt the "rule feedback" oracle from McCallum et al. (2023) without any changes. This feedback triggers when agents attempt actions that have no visible effect on the environment due to its affordances—such as trying to move through closed doors, open walls, or stack objects (which is not permitted in the environment).

Shaped Rewards. We apply reward shaping to BabyAI's original binary reward function and derive intermediate scalar rewards by leveraging the success and action failure detection mechanisms in the feedback oracles. Note that while our rewards may appear dense relative to the original binary rewards, they are comparably sparse in the context of typical RL problems (Bellemare et al., 2013; Cobbe et al., 2019; Tassa et al., 2018; Wydmuch et al., 2018, etc.). Preliminary results showed that conditioning only on the original binary rewards is equivalent to foregoing return-to-go conditioning altogether. For further details, refer to Appendix F.

4.5 Baselines and variants

Following the Uni[MASK] approach (Carroll et al., 2022), we use the same base architecture for all our models. All model configurations are illustrated in Appendix B. We obtain IL baselines (where feedback is NONE) by masking all additional tokens except the one-off mission instructions. We unmask relevant additional tokens to achieve variants, specifically *returns-to-go* for variant SCALAR or *language feedback* for LANG, as well as *both* for COMBO, and optionally predict the scalar reward or language feedback. For comprehensive details on training see Appendix G.

5 Results

5.1 Compositional Generalisation

Systematicity. We test if models can generalise to unseen combinations of goal colour and shape at test time. Table 1 shows that performance using language feedback is comparable to the performance with scalar rewards when the model is trained to predict its own feedback, while combining language feedback and scalar rewards slightly surpasses the performance achieved with either feedback modalities when trained with the auxiliary feedback prediction objective. The performance of models trained with suboptimal trajectories and some form of feedback is more than four times higher than for the IL baselines.

Productivity. In the context of our embodied setting, we define productivity as the ability to extrapolate to unseen values of mission or environment parameters. We isolate dimensions of compositionality in the mission in the form of categorical variables such as goal attributes, while in the environment, we identify compositionality w.r.t numerical variables, specifically the room dimensions or the number of obstacles. Note that while for compositionality in the mission, we use the language feedback and scalar rewards corresponding to the *task feedback* oracle, we measure compositionality in the environment in the context of *affordance feedback*.

The results in Table 1 indicate that language feedback is most effective when extrapolating to unseen goal colours, more complex layouts or higher numbers of obstacles. In these cases, language feedback performs roughly on par with the scalar rewards. In contrast, language feedback is less effective when tested on object locations that never appeared in the training data. We hypothesise that the models struggle to correctly ground language describing spatial relations and therefore, language feedback hinders more than it helps. We also test whether models trained on simple sequential instructions can extrapolate to more complex, compound instructions with multiple connectors. Specifically, models must understand the different connectors between two sequences ('and', 'then', or 'after'), which only occur in isolation in the training data. Surprisingly, learning from multiple trajectories per mission task improves performance of the baseline on unseen sequence tasks. It is conceivable that seeing successful sequences connected by 'and' in different orders helps the model learn differences between order-agnostic coordinating conjunctions from temporal adverbs and subordinating conjunctions ('then' and 'after'). However, we find that language feedback only provides comparable ben-

Feedback	Enhancement		Systematicity	Productivity					
	ST	FP	Color-Shape	Color†	Location†	Sequence†	Layout*	Obstacles*	All
NONE	×	×	16.9	13.5	4.9	10.0	70.3	82.2	33.1
	✓	×	15.1	14.1	2.8	44.4	75.1	83.1	39.1
SCALAR	✓	×	69.6	68.0	19.7	69.8	80.6	87.3	65.8
	✓	✓	69.4	68.8	21.3	71.9	80.0	85.6	66.2
LANG	✓	×	68.0	66.5	13.8	59.1	80.9	88.3	62.8
	✓	✓	69.7	69.1	14.6	66.1	79.2	85.9	64.1
COMBO	✓	×	70.3	66.6	17.7	67.3	78.0	83.9	64.0
	✓	✓	71.7	68.6	20.3	69.5	78.5	81.6	65.0

Table 1: Success rates (%) across different dimensions of compositional generalisation: Systematicity (combinatorial interpolation to novel combinations) and Productivity (extrapolating to unseen values). We average the performance across multiple tasks, each requiring both navigation and object interaction/manipulation. A breakdown of tasks per generalization setting can be found in Table E.1 in the appendix. †compositionality in the mission, *compositionality in the environment. ST=suboptimal trajectories, FP=feedback prediction.

efits to scalar rewards when are trained with the auxiliary feedback prediction task.

5.2 Robustness

Representations of subgoals. We see evidence that models trained with language feedback or scalar rewards learn more robust representations of how to solve *individual* subgoals. We find that models trained without any form of feedback partially complete many tasks but rarely complete *all* subgoals required for task success, whereas those trained with feedback signals and several more or less optimal solutions for a given task appear to be significantly more successful at fully completing tasks, as illustrated in Figure 4a. Equipping models with language feedback and requiring them to learn how to predict their own feedback can further reduce the number of tasks that are only partially completed. Unlike the scalar rewards, language feedback reflects whether actions lead to *subgoal* success or to *task* success. Therefore, language feedback-conditioned models which are also trained to predict their own feedback can develop a more nuanced understanding of the difference between partial and full task completion.

Robustness to lack of optimal data. To simulate scenarios where optimal trajectories are categorically not available, we further test to what extent our method allows us to rely exclusively on suboptimal trajectories. Table 2 illustrates how, when we omit the optimal trajectories, the advantage of the models that have access to some form of feedback becomes even clearer. We note the most pronounced relative performance drop for the baseline trained without step-level feedback (NONE), while removing the optimal data has the least impact on the COMBO model trained on scalar and language feedback. This suggests that the step-level feedback effectively enables trajectory stitching from different suboptimal solutions.

Feedback	ST + OT	ST only	Drop
NONE	15.1	12.1	-19.9%
SCALAR	69.6	62.3	-10.5%
LANG	68.0	61.2	-10.0%
COMBO	70.3	65.6	-6.7%

Table 2: Relative drop in success rate (%) when using only suboptimal trajectories (ST) vs suboptimal + optimal trajectories (ST+OT). Results are averaged for in-distribution Pickup and PutNext missions.

Robustness to external perturbations. We simulate external perturbations, such as hardware failure, using a variant of *sticky actions* (Machado et al., 2017). However, rather than delaying execution by k steps, we simply randomly replace the current action with the sticky action. As a result, we can test the agent's ability to recover without inflating the number of steps taken on the environment. Figure 4b shows that models trained with suboptimal trajectories but without any form of feedback (NONE W/ ST) to contextualise mistakes are more the most adversely affected by external perturbations, with performance dropping to under 70%. For models trained on language feedback, it is paramount to be able to predict feedback (LANG + FP), as this allows them to robustly anticipate the outcome of a given action, even if it is not the action that was originally predicted. This means the model can retain 75% of the original

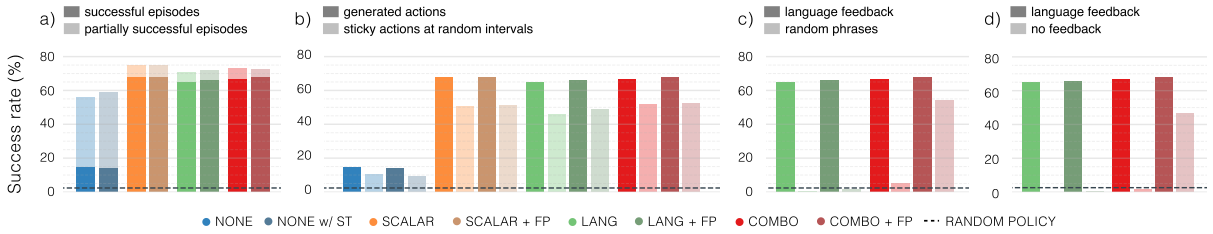


Figure 4: Comparison of success rates under various robustness evaluation settings. We report robustness results on in-distribution data and average over Pickup and PutNext tasks. From left to right: a) representations of subgoals, b) external perturbations, c) adversarial feedback, d) missing feedback. b)-d) correspond to alternative inference scenarios. ST=suboptimal trajectories, FP=feedback prediction.

performance, making it equally as robust to external perturbations as models trained with scalar rewards (SCALAR and SCALAR + FP). Contextualising the language feedback with scalar rewards (COMBO + FP) can further aid recovery, leading to the most robust model variant, which retains 78% of the original performance when faced with external perturbations.

Robustness to adversarial or missing language feedback. We test the sensitivity of models trained with language feedback to scenarios where feedback is either unhelpful or *adversarial*, or entirely unavailable. We simulate the adversarial case by providing random English sentences at random time steps. Figure 4c-d show that in these settings, the performance of models trained with language feedback collapses. While we consider this a positive sign that models do not simply learn to rely on spurious correlations (Parekh et al., 2024), it is ultimately desirable for them to be effective and safe in a range of deployment scenarios. Our experiments further reveal that when models are trained on both language and the corresponding scalar rewards and additionally required to learn to predict the resulting language feedback for their actions (COMBO + FP), they retain 70%–80% of their original performance. We attribute this effect specifically to the *combination* of scalar rewards as context and the ability to anticipate feedback, as in isolation, these augmentations do not provide any tangible benefits. For additional adversarial test scenarios, refer to Appendix H.

Solution efficiency. We normalise the path lengths *of successful episodes* by the corresponding oracle path lengths and use this as a proxy for how efficient solutions were. Table 3 shows that all models, including the baseline that only learns from optimal trajectories, on average favour solutions that are less efficient than the oracle paths. At the same time, all models are able to find more optimal paths than the oracle[5], including for missions seen during training. As expected, including suboptimal trajectories but no feedback makes solutions the least efficient, while only learning from optimal trajectories results in the highest path efficiency. The results indicate that when models are trained with language feedback and the auxiliary feedback prediction task, they are able to learn the most efficient solutions from suboptimal data.

	Enhancement		ON Path Lengths		
Feedback	ST	FP	Mean	Min	Min (train)
NONE	×	×	220.9%	94.1%	91.7%
	✓	×	253.8%	75.0%	89.5%
SCALAR	✓	×	244.1%	69.2%	86.2%
	✓	✓	239.1%	69.2%	89.5%
LANG	✓	×	240.8%	69.2%	89.5%
	✓	✓	233.6%	75.0%	89.5%
COMBO	✓	×	245.4%	69.2%	86.2%
	✓	✓	236.1%	75.0%	89.5%

Table 3: Oracle-normalised path lengths of successful episodes for in-distribution missions, averaged over Pickup and PutNext tasks. <100% corresponds to more efficient solutions, and >100% to less efficient solutions. ST=suboptimal trajectories, FP=feedback prediction.

5.3 Data efficiency and data scaling

We compare the performance on a representative compositional generalisation setting (*Systematicity*) for all models, as well as a PPO (Schulman et al., 2017) baseline[6] when training on 25%, 50% and 100% of the data used in our experiments. The results in Figure 5 suggest that the models con-

[5]While the BFS-planner oracle always yields successful paths that are near-optimal but not guaranteed to be optimal.

[6]We adopt implementation and hyper-parameters in the original BabyAI benchmark, as detailed in Appendix C.

Feedback	Enhancement ST	Enhancement FP	Proportion of data 25%	Proportion of data 25%→50%	Proportion of data 50%→100%
NONE	×	×	13.1	+0.6	+3.3
	✓	×	11.6	+3.0	+0.5
SCALAR	✓	×	53.6	+10.0	+6.0
	✓	✓	55.0	+9.6	+4.9
LANG	✓	×	50.7	+9.5	+7.8
	✓	✓	52.0	+7.7	+10.0
COMBO	✓	×	53.4	+4.6	+12.2
	✓	✓	49.9	+11.1	+10.8

Table 4: Change in success rate (%) when doubling the amount of training data. ST=suboptimal trajectories, FP=feedback prediction.

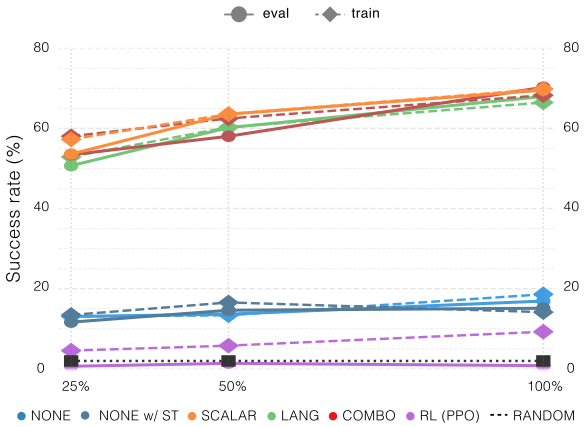


Figure 5: Change in success rate on *Systematicity* tasks as the amount of training data increases, averaged over Pickup and PutNext tasks. ST=suboptimal trajectories

textualising suboptimal training trajectories with any form of feedback are not only more data efficient than either IL from optimal trajectories or PPO, they also show much stronger scaling properties. It appears that the COMBO variant shows the strongest late-stage scaling, which suggests that this setting may require more data to outperform settings using single feedback types.

6 Conclusion

The continued popularity of language modelling objectives for LLMs sparked great interest in revisiting IL techniques for embodied agents (e.g., Ehsani et al., 2024). However, these are limited due to their inability to exploit suboptimal trajectories for learning more generalisable and robust behaviour. In this paper, we demonstrate the potential of language feedback as an efficient and intuitive feedback mechanism for IL when the dataset either contains or consists entirely of suboptimal trajectories, and show its potential as a viable alternative to scalar rewards for tasks specified in language. We define an experimental setup based on a highly configurable 2D grid-world environment for learning from pixel-based, partial observations, which we designed expressly to assess performance along different axes of compositional generalisation (Hupkes et al., 2019). We show that models trained with both optimal and suboptimal data and feedback exhibit superior compositional generalisation abilities and increased robustness. We find that language feedback is not only a suitable alternative for scalar rewards for harnessing the learning potential of suboptimal samples, but also shows how we can increase robustness of models trained with language feedback with auxiliary feedback or by combining the strengths of both approaches. We are optimistic that our findings will inform future work on LLM-based policies and that language feedback can play a role in different fine-tuning regimes and for alignment with human preferences.

Limitations

Environment complexity. This work is conducted in BabyAI-XGen, a grid-world environment that, while providing controlled experimental conditions, falls short of the visual realism, continuous action spaces, and physics fidelity of real-world robotic tasks. We chose this environment to enable precise control over a range of compositional generalization factors, and to ensure reliable trajectory generation, which would be difficult to achieve in more complex 3D environments. However, the central principle of learning from suboptimal trajectories contextualized with feedback is domain-agnostic, and future work should explore scaling to more realistic environments where optimal trajectories may be even harder to obtain.

Limited model architecture. Our approach uses a relatively small Transformer backbone (~90M parameters) trained from scratch, which may not reflect the capabilities of larger foundation models. This choice was made to avoid confounding factors from pre-trained knowledge and ensure fair comparison across feedback conditions. Future work should investigate how pre-trained language models and the corresponding larger architectures might better leverage feedback signals, particularly for more complex reasoning about multi-step errors and corrections (Chiyah-Garcia et al., 2024), keeping in mind that careful scaling of training data to match the increased model capacity will likely

be required, as our current dataset size could lead to overfitting or suboptimal utilization of larger architectures.

Synthetic feedback generation. We generate feedback programmatically using environment internals rather than studying feedback quality or generation mechanisms. This approach enables controlled experimentation without confounding factors from inconsistent or incorrect model-generated feedback while minimising computational overheads. However, future work should explore the development of more sophisticated step-level feedback generation mechanisms involving VLM-based systems.

Cost and practicality of feedback collection. While our results demonstrate the effectiveness of language feedback, practical deployment faces important cost considerations. We focused on systematic feedback generation to isolate the core research question of whether feedback can transform suboptimal data into useful learning signals. However, for many embodied domains, step-level scalar rewards and language feedback will likely require human annotation or sophisticated reward models, with language feedback potentially incurring higher costs than scalar rewards. Future work should systematically evaluate these tradeoffs across different domains and explore efficient feedback collection strategies, particularly for interactive domains where language feedback is inherently more intuitive than scalar scores—such as collaborative robotics requiring natural human-robot communication, interactive tutoring systems where explanatory feedback aids learning, and multi-agent coordination tasks where agents need to understand and communicate about errors and corrections.

Potential Risks and Ethical Considerations

We believe our work on enhancing embodied AI through constructive language feedback holds significant promise for developing more robust and adaptable robotic systems. By enabling agents to learn from a wider range of demonstrations, including suboptimal ones, we move closer to creating AI that can effectively operate in complex and unpredictable real-world environments. However, the ability of AI agents to learn from and act upon language feedback also introduces important ethical considerations and potential risks. As these systems become more sophisticated and integrated into our lives, ensuring their safety, reliability, and fairness becomes paramount. For instance, if language feedback is biased or malicious, an embodied AI agent could learn and perpetuate harmful behaviors or make decisions with unintended negative consequences in real-world scenarios. Aware of this potential issue, we have designed our experimental setup with robustness in mind and defined specific scenarios where we are challenging models with unexpected actions or adversarial language instructions. As shown in our experiments, models trained to predict feedback are more resilient to these perturbations, offering a more compelling solution for future applications.

Due to these reasons, we also decided to focus on a controllable environment where action execution can be simulated without major repercussions on the world and the humans within it. However, we also acknowledge that, due to this reason, the models trained in this setting might be biased towards simulation environments and might not directly generalize to real-world environments.

References

Ademi Adeniji, Amber Xie, Carmelo Sferrazza, Younggyo Seo, Stephen James, and P. Abbeel. 2023. Language reward modulation for pretraining reinforcement learning. *ArXiv*, abs/2308.12270.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683.

M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2012. The arcade learning environment: An evaluation platform for general agents. *ArXiv*, abs/1207.4708.

Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, and Ury Zhilinsky. 2024. $\pi 0$: A vision-language-action flow model for general robot control. *ArXiv*, abs/2410.24164.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the International Conference on Computer Vision (ICCV)*.

Micah Carroll, Orr Paradise, Jessy Lin, Raluca Georgescu, Mingfei Sun, David Bignell, Stephanie Milani, Katja Hofmann, Matthew J. Hausknecht, Anca D. Dragan, and Sam Devlin. 2022. Unimask: Unified inference in sequential decision problems. *ArXiv*, abs/2211.10869.

Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, P. Abbeel, A. Srinivas, and Igor Mordatch. 2021. Decision transformer: Reinforcement learning via sequence modeling. In *Neural Information Processing Systems*.

Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. 2018. Babyai: A platform to study the sample efficiency of grounded language learning. In *International Conference on Learning Representations*.

Javier Chiyah-Garcia, Alessandro Suglia, and Arash Eshghi. 2024. Repairs in a block world: A new benchmark for handling user corrections with multimodal language models. In *Conference on Empirical Methods in Natural Language Processing*.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. 2019. Leveraging procedural generation to benchmark reinforcement learning. In *International Conference on Machine Learning*.

Matt Deitke, Dhruv Batra, Yonatan Bisk, Tommaso Campari, Angel X Chang, Devendra Singh Chaplot, Changan Chen, Claudia Pérez D'Arpino, Kiana Ehsani, Ali Farhadi, et al. 2022. Retrospectives on the embodied ai workshop. *arXiv preprint arXiv:2210.06849*.

Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, P. Abbeel, Abhishek Gupta, and Jacob Andreas. 2023. Guiding pretraining in reinforcement learning with large language models. In *International Conference on Machine Learning*.

Kiana Ehsani, Tanmay Gupta, Rose Hendrix, Jordi Salvador, Luca Weihs, Kuo-Hao Zeng, Kunal Pratap Singh, Yejin Kim, Winson Han, Alvaro Herrasti, et al. 2024. Spoc: Imitating shortest paths in simulation enables effective navigation and manipulation in the real world. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16238–16250.

Desmond Elliott and Ákos Kádár. 2017. Imagination improves multimodal translation. In *International Joint Conference on Natural Language Processing*.

Jonathan Frankle and Michael Carbin. 2018. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv: Learning*.

Xiaofeng Gao, Qiaozi Gao, Ran Gong, Kaixiang Lin, Govind Thattai, and Gaurav S. Sukhatme. 2022. Dialfred: Dialogue-enabled agents for embodied instruction following. *IEEE Robotics and Automation Letters*, 7:10049–10056.

Chengyang Gu, Yuxin Pan, Haotian Bai, Hui Xiong, and Yize Chen. 2024. Seal: Semantic-augmented imitation learning via language model. *ArXiv*, abs/2410.02231.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. 2019. Dream to control: Learning behaviors by latent imagination. *ArXiv*, abs/1912.01603.

Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. 2020. Mastering atari with discrete world models. *ArXiv*, abs/2010.02193.

Danijar Hafner, J. Pavsukonis, Jimmy Ba, and Timothy P. Lillicrap. 2023. Mastering diverse domains through world models. *ArXiv*, abs/2301.04104.

Sukai Huang, Shu-Wei Liu, Nir Lipovetzky, and Trevor Cohn. 2024. The dark side of rich rewards: Understanding and mitigating noise in vlm rewards. *arXiv preprint arXiv:2409.15922*.

Dieuwke Hupkes, Verna Dankers, Mathijs Mul, and Elia Bruni. 2019. Compositionality decomposed: How do neural networks generalise? *J. Artif. Intell. Res.*, 67:757–795.

Dieuwke Hupkes, Mario Giulianelli, Verna Dankers, Mikel Artetxe, Yanai Elazar, Tiago Pimentel, Christos Christodoulopoulos, Karim Lasri, Naomi Saphra, Arabella Sinclair, Dennis Ulmer, Florian Schottmann, Khuyagbaatar Batsuren, Kaiser Sun, Koustuv Sinha, Leila Khalatbari, Maria Ryskina, Rita Frieske, Ryan Cotterell, and Zhijing Jin. 2023. A taxonomy and review of generalization research in nlp. *Nature Machine Intelligence*, 5:1161–1174.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. 2016. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*.

Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. 2023. Vima: robot manipulation with multimodal prompts. In *Proceedings of the 40th International Conference on Machine Learning*, pages 14975–15022.

Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag R. Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. 2024. Openvla: An open-source vision-language-action model. *ArXiv*, abs/2406.09246.

Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktaschel. 2021. A survey of generalisation in deep reinforcement learning. *ArXiv*, abs/2111.09794.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Jessy Lin, Yuqing Du, Olivia Watkins, Danijar Hafner, P. Abbeel, Dan Klein, and Anca D. Dragan. 2023. Learning to model the world with language. *ArXiv*, abs/2308.01399.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning. *ArXiv*, abs/2304.08485.

Cong Lu, Shengran Hu, and Jeff Clune. 2024. Intelligent go-explore: Standing on the shoulders of giant foundation models. *ArXiv*, abs/2405.15143.

Chang Ma, Junlei Zhang, Zhihao Zhu, Cheng Yang, Yujiu Yang, Yaohui Jin, Zhenzhong Lan, Lingpeng Kong, and Junxian He. 2024. Agentboard: An analytical evaluation board of multi-turn llm agents. *ArXiv*, abs/2401.13178.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael H. Bowling. 2017. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *ArXiv*, abs/1709.06009.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Sabrina McCallum, Max Taylor-Davies, Stefano V. Albrecht, and Alessandro Suglia. 2023. Is feedback all you need? leveraging natural language feedback in goal-conditioned reinforcement learning. *ArXiv*, abs/2312.04736.

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*, 7(3):7327–7334.

Meta and Hugging Face. 2024. Llama 2. https://huggingface.co/docs/transformers/en/model_doc/llama2.

So Yeon Min, Hao Zhu, Ruslan Salakhutdinov, and Yonatan Bisk. 2022. Don't copy the teacher: Data and model challenges in embodied dialogue. *ArXiv*, abs/2210.04443.

Nvidia, Johan Bjorck, Fernando Castaneda, Nikita Cherniadev, Xingye Da, Runyu Ding, LinxiJimFan, Yu Fang, Dieter Fox, Fengyuan Hu, Spencer Huang, Joel Jang, Zhenyuan Jiang, Jan Kautz, Kaushil Kundalia, Lawrence Lao, Zhiqi Li, Zongyu Lin, Kevin Lin, Guilin Liu, Edith Llontop, Loic Magne, Ajay Mandlekar, Avnish Narayan, Soroush Nasiriany, Scott Reed, You Liang Tan, Guanzhi Wang, Zu Wang, Jing Wang, Qi Wang, Jiannan Xiang, Yuqi Xie, Yinzhen Xu, Zhen-Teng Xu, Seonghyeon Ye, Zhiding Yu, Ao Zhang, Hao Zhang, Yizhou Zhao, Ruijie Zheng, and Yuke Zhu. 2025. Gr00t n1: An open foundation model for generalist humanoid robots. *ArXiv*, abs/2503.14734.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, P. Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramithu, Gokhan Tur, and Dilek Z. Hakkani-Tür. 2021. Teach: Task-driven embodied agents that chat. In *AAAI Conference on Artificial Intelligence*.

Richard Yuanzhe Pang, Weizhe Yuan, Kyunghyun Cho, He He, Sainbayar Sukhbaatar, and Jason Weston. 2024. Iterative reasoning preference optimization. *arXiv preprint arXiv:2404.19733*.

Amit Parekh, Nikolas Vitsakis, Alessandro Suglia, and Ioannis Konstas. 2024. Investigating the role of instruction variety and task difficulty in robotic manipulation tasks. *ArXiv*, abs/2407.03967.

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

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Conference on Empirical Methods in Natural Language Processing*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2019. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10737–10746.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2020. Alfworld: Aligning text and embodied environments for interactive learning. *ArXiv*, abs/2010.03768.

Mustafa Shukor, Dana Aubakirova, Francesco Capuano, Pepijn Kooijmans, Steven Palma, Adil Zouitine, Michel Aractingi, Caroline Pascal, Martino Russi, Andrés Marafioti, Simon Alibert, Matthieu Cord, Thomas Wolf, and Rémi Cadène. 2025. Smolvla: A vision-language-action model for affordable and efficient robotics. *ArXiv*, abs/2506.01844.

Alessandro Suglia, Antonio Vergari, Ioannis Konstas, Yonatan Bisk, Emanuele Bastianelli, Andrea Vanzo, and Oliver Lemon. 2020. Imagining grounded conceptual representations from perceptual information in situated guessing games. In *International Conference on Computational Linguistics*.

Richard S Sutton. 2018. Reinforcement learning: An introduction. *A Bradford Book*.

Yuval Tassa, Yotam Doron, Alistair Muldal, Tom Erez, Yazhe Li, Diego de Las Casas, David Budden, Abbas Abdolmaleki, Josh Merel, Andrew Lefrancq, Timothy P. Lillicrap, and Martin A. Riedmiller. 2018. Deepmind control suite. *ArXiv*, abs/1801.00690.

Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Pannag R. Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. 2024. Octo: An open-source generalist robot policy. *ArXiv*, abs/2405.12213.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurélien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv*, abs/2302.13971.

Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. 2024. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*.

Ashish Vaswani, Noam M. Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Neural Information Processing Systems*.

H. J. Austin Wang and Karthik Narasimhan. 2021. Grounding language to entities and dynamics for generalization in reinforcement learning. *ArXiv*, abs/2101.07393.

Lucas Willems. 2023. Rl starter files.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. 2018. Vizdoom competitions: Playing doom from pixels. *IEEE Transactions on Games*. IEEE Transactions on Games outstanding paper award 2022.

Jiajun Xi, Yinong He, Jianing Yang, Yinpei Dai, and Joyce Chai. 2024a. Teaching embodied reinforcement learning agents: Informativeness and diversity of language use. *ArXiv*, abs/2410.24218.

Zhiheng Xi, Yiwen Ding, Wenxiang Chen, Boyang Hong, Honglin Guo, Junzhe Wang, Dingwen Yang, Chenyang Liao, Xin Guo, Wei He, Songyang Gao, Luyao Chen, Rui Zheng, Yicheng Zou, Tao Gui, Qi Zhang, Xipeng Qiu, Xuanjing Huang, Zuxuan Wu, and Yu-Gang Jiang. 2024b. Agentgym: Evolving large language model-based agents across diverse environments. *ArXiv*, abs/2406.04151.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. *ArXiv*, abs/2210.03629.

Sriram Yenamandra, Arun Ramachandran, Karmesh Yadav, Austin Wang, Mukul Khanna, Theophile Gervet, Tsung-Yen Yang, Vidhi Jain, Alexander William Clegg, John Turner, et al. 2023. Homerobot: Open-vocabulary mobile manipulation. *arXiv preprint arXiv:2306.11565*.

Alex Zhang, Khanh Nguyen, Jens Tuyls, Albert Lin, and Karthik Narasimhan. 2024. Language-guided world models: A model-based approach to ai control. *ArXiv*, abs/2402.01695.

Artem Zholus, Alexey Skrynnik, Shrestha Mohanty, Zoya Volovikova, Julia Kiseleva, Artur Szlam, Marc-Alexandre Côté, and Aleksandr I. Panov. 2022. Iglu gridworld: Simple and fast environment for embodied dialog agents. *ArXiv*, abs/2206.00142.

Victor Zhong, Dipendra Misra, Xingdi Yuan, and Marc-Alexandre Cot'e. 2024. Policy improvement using language feedback models. *ArXiv*, abs/2402.07876.

A Further background on related work

Language-guided embodied AI An exception to the customary approach of creating embodied AI datasets from planner-generated trajectories is TEACh (Padmakumar et al., 2021), which was collected from pairs of human annotators collaborating to complete tasks in the simulated environment. A proportion of TEACh trajectories contain inefficiencies and corrections, typically resulting from miscommunication between the (Min et al., 2022). However, any such occurrences of suboptimal behaviour in TEACh are incidental rather than systematic, and the dataset does not include multiple example trajectories for the same task instance.

Language as feedback For the purpose of our study, we consider language feedback from an oracle that is external to the agent itself, and therefore does not directly apply to cases where agents self-improve through an internal process of self-reflection (Madaan et al., 2024; Yao et al., 2022).

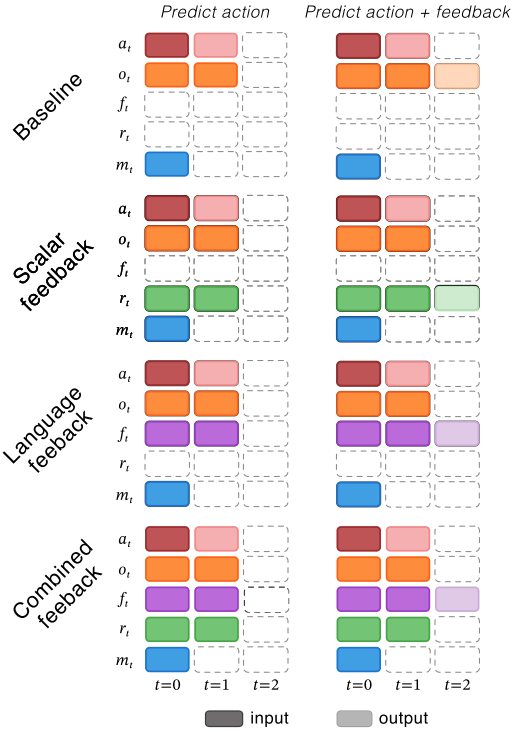
Compositional generalisation As has been argued in Kirk et al. (2021), many of the categories of compositional generalisation originally introduced by Hupkes et al. (2019) for language are applicable for agents in interactive environments; we focus on two of the five categories: *Systematicity* and *Productivity* and test to what extent learning from multiple solutions and suboptimal behaviour can translate into increased Systematicity and Productivity in agents that have access to different feedback signals. For details on the remaining three categories, we refer the reader to the original definitions in Hupkes et al. (2019). In the context of embodied agents and interactive environments, Systematicity pertains to the ability to systematically recombine known components and rules to novel combinations (Hupkes et al., 2019; Kirk et al., 2021). Kirk et al. (2021) refer to this as *combinatorial interpolation*, as the agent needs to interpolate to values of environment parameters which it has seen independently but not in combination. Productivity, which Hupkes et al. (2019) define as the ability of models to generate to output sequences that exceed the length of the sequences seen during training, is loosely equivalent to *extrapolation* in (Kirk et al., 2021), according to which the values for a single or multiple environment parameters fall outside the ranges seen during training; as the resulting environments tend to be more complex, agents are typically required to generate longer trajectories than those present in the input. In the case of language-guided embodied tasks, we can define compositionality with respect to the language of the mission instructions or the complexity of the environment.

Parameter	Value
vocab_size	*1*
hidden_size	*768*
intermediate_size	*3072*
num_hidden_layers	*768*
num_attention_heads	*12*
num_key_value_heads	*12*
hidden_act	SiLU
max_position_embeddings	2048
initializer_range	0.02
rms_norm_eps	1e-6
use_cache	True
pad_token_id	None
bos_token_id	1
eos_token_id	2
pretraining_tp	1
tie_word_embeddings	False
rope_theta	10000.0
rope_scaling	None
attention_bias	False
attention_dropout	0.0
mlp_bias	False
head_dim	None

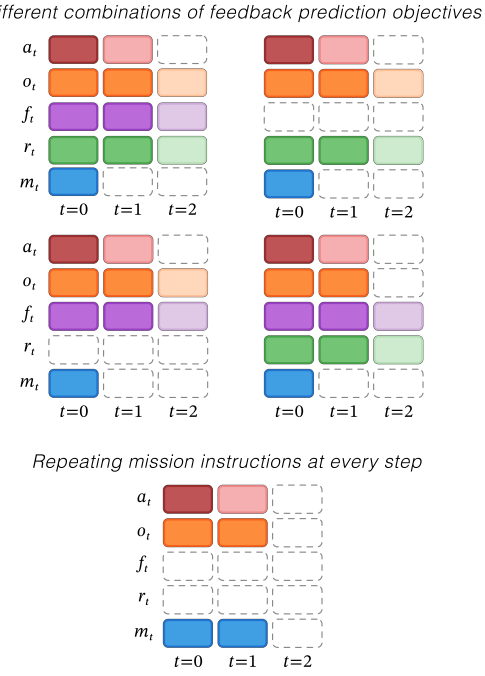
Table B.1: Configuration of our Llama-style backbone. Most values are taken from the huggingface implementation (Meta and Hugging Face, 2024). Values we override are highlighted in italics.

B Method details

Transformer backbone. Unlike the Decision Transformer, which uses a scaled-down GPT-2 model(Radford et al., 2019) at its core, we adapt architecture from Llama2 (Touvron et al., 2023) to serve as our Transformer backbone, which allows us to leverage innovations such as the RoPE (Rotary Position Embedding) positional encodings, Grouped-Query Attention (GQA), and longer context lengths to facilitate learning from entire trajectories, rather than sub-trajectories of limited context lengths, as well as more efficient training. Specifically, we use the Flash Attention implementation of Llama from Huggingface. Note that we do not use the original Llama2 configuration or weights; instead we use the model dimensions cor-



(a) Input and output tokens of the models used in our experiments. For instance, models using *scalar feedback* condition action generation on instructions and returns-to-go, and optionally predict rewards of the next time step.



(b) Further possible models supported by our flexible token masking scheme.

Figure B.1: We use a flexible token masking scheme inspired by Carroll et al. (2022) to achieve different configurations from the same base model to mask unused tokens, where m_i=instructions, f_i=language feedback, r_i=returns-to-go/rewards, o_i=observations, a_i=actions.

responding to those use for the Base size of Octo (Team et al., 2024), a pretrained Transformer-based robot policy, and randomly initialise weights. We give an overview of the configuration of the Llama-style backbone in Table B.1

Flexible token masking. Similar to Carroll et al. (2022), we devise a flexible masking scheme to facilitate conditioning on different additional tokens while keeping the model architecture constant and reusing the same datasets for different feedback options. Supported additional tokens range from one-off language instructions, to scalar rewards, language feedback, and their combination. Tokens are either masked out for the whole sequence if irrelevant for a given test case or only for those time steps they were not provided for. Figure B.1a illustrates the masking schemes used for our models used in this study, while Figure B.1b highlights additional use cases of our masking schemes. For instance, we can accommodate arbitrary combinations of feedback prediction objectives, and optionally repeat the mission instructions at every step

Token embeddings. While our design is deliberately modular and supports a range of image and text encoder options, we opt for sentence-level text embeddings from a fully frozen Sentence-BERT (Reimers and Gurevych, 2019) corresponding to the pre-trained general purpose model `all-mpnet-base-v2`, for the mission instructions and language feedback (where applicable) to allow us to condense long text inputs into compact vector representations corresponding to a single token, and a custom CNN image encoder (see Table B.2) proposed for pixel-based RL experiments in MiniGrid and BabyAI by (Willems, 2023), which we train concurrently with the policy, and which condenses the image input into one token. Compressing the multi-modal input modalities into a single token per timestep most closely aligns with the input into the Transformer backbone is constructed in Decision Transformers. We project language embeddings using a projection layer similar to LlaVa (Liu et al., 2023), where $f(\mathbf{X}_t)$ is the text encoder, and $\mathbf{W}$ is the projection matrix which maps the text features into the target embedding space with the dimensionality of the Llama-style backbone (Equation (1)). The image embeddings are projected using a simple projection layer in keeping with the implementation of the CNN used for BabyAI.

$$\mathbf{H}_t = \mathbf{W} \cdot \mathbf{Z}_t, \text{ with } \mathbf{Z}_t = f(\mathbf{X}_t) \quad (1)$$

Block	Operation	In Ch.	Out Ch.	Kernel
Conv Block 1	Conv2D	3	16	(2, 2)
	ReLU	16	16	–
Pooling	MaxPool2D	16	16	(2, 2)
Conv Block 2	Conv2D	16	32	(2, 2)
	ReLU	32	32	–
Conv Block 3	Conv2D	32	64	(2, 2)
	ReLU	64	64	–
Output	Flatten	64	–	–

Table B.2: Architecture of the custom CNN used to embed image observations adopted from (Willems, 2023).

We embed the discrete actions and timesteps using Embedding layers. After adding the timestep embeddings to the token-specific embeddings, the token embeddings for each timestep are interleaved to form flattened sequences, $(m_1, r_1, f_1, o_1, a_1, \ldots, m_T, r_T, f_T, o_T, a_T)$, where T is equal to the number of steps of the trajectory. The interleaved token embeddings are then normalised and passed as input to the Llama-style backbone, along with the corresponding attention mask, where we mask out unused tokens. Note that, unlike in the original Decision Transformer, we leverage an additional level of position embeddings for each token in the flattened sequence in the Llama-style backbone.

Prediction heads and losses. We follow previous work on Decision Transformers (Chen et al., 2021) and learn to predict actions in the environment as a next token prediction task by minimising cross-entropy loss of the actions. We consider this as our main loss function for the training of the agent's policy. We consider several auxiliary losses aimed at predicting property tokens at the next timesteps. We define auxiliary losses for a range of property tokens including *image tokens*, *reward tokens*, and *feedback tokens*. Our losses are inspired by the latent imagination method by Elliott and Kádár (2017). For each loss, we leverage the hidden state $\mathbf{h}_t$ associated with the timestep t of our Transformer backbone to predict the embedding of the property token at the next timestep $\mathbf{p}_{t+1}$ via a dedicated prediction head P. We report below the details of each loss.

Action token prediction. For the action prediction, we treat this as a classification task over the discrete action space. We assume the target to be the action a_{t+1}. We define a prediction head P_a as a linear layer that takes the hidden state $\mathbf{h}_t$ and outputs unnormalized logits over the action vocabulary. We use the cross-entropy loss as follows:

$$\mathcal{L}_a = -\sum_{t=1}^{T} \log(p(a_t|\mathbf{h}_{t-1})) \tag{2}$$

Feedback token prediction. For the feedback prediction, we assume that our embedding representation for the feedback is the Sentence-BERT representation $\mathbf{f}_{t+1}$ associated with the feedback in position $t+1$. We define a prediction head P_f as a linear layer with GELU activations. We use the MSE loss as follows:

$$\mathcal{L}_f = \text{MSE}(P_f(\mathbf{h}_t), \mathbf{f}_{t+1}) \tag{3}$$

Reward token prediction. For the reward prediction, we assume the target to be the reward G_{t+1} associated with the reward in position $t+1$ (see Appendix E for details). We define a prediction head P_r as a linear layer with SIGMOID activation. We use the MSE loss as follows:

$$\mathcal{L}_r = \text{MSE}(P_r(\mathbf{h}_t), G_{t+1}) \tag{4}$$

Image token prediction. For the image prediction, we follow a similar definition to Elliott and Kádár (2017), but instead of the MSE loss, we calculate the loss based on the Cosine similarity between the image embedding the $\mathbf{i}_{t+1}$ produced by the CNN encoder for the timestep $t+1$, and the image embeddings predicted by the prediction head consisting of a linear layer P_i with RELU activation. We use a Cosine loss as follows:

$$\mathcal{L}_i = 1 - \cos(P_i(\mathbf{h}_t), \mathbf{i}_{t+1}) \tag{5}$$

In our experiments, we found this loss did not improve performance consistently compared to the baseline (see Appendix G). We leave for future work exploring more specific training regimes which can better leverage this loss following work in self-supervised representation learning (Caron et al., 2021; Jaderberg et al., 2016).

Weighted average loss. Due to the difference in magnitude between the different losses, we learn via SGD a loss-specific weight to balance the contribution of each auxiliary loss. We compute the final loss as a weighted average of the different losses used by a certain model configuration.

Name	Type	Params
Embed returns	*Linear*	*1.5K*
Embed images	Custom CNN	10.5K
Embed actions	Embedding	5.4K
Embed timesteps	Embedding	444K
Project image embeddings	Linear ReLU	30.7M
Project language embedding	Linear Linear GELU	1.2M
Normalise embeddings	LayerNorm	1.5K
Backbone	LlamaModel	113M
Predict actions	Linear	4.6K
Predict feedback emb.	*Linear* *GELU*	*590K*
Predict image emb.	*Linear* *ReLU*	*590K*
Predict rewards	*Linear* *Sigmoid*	*769*

Table B.3: Model architecture, for a total parameter count of ∼146M. Optional modules in italics. Note that we pass only the instructions through the language embedding projection layers for models that don't use language feedback. Details on the CNN and the Llama backbone are provided separately.

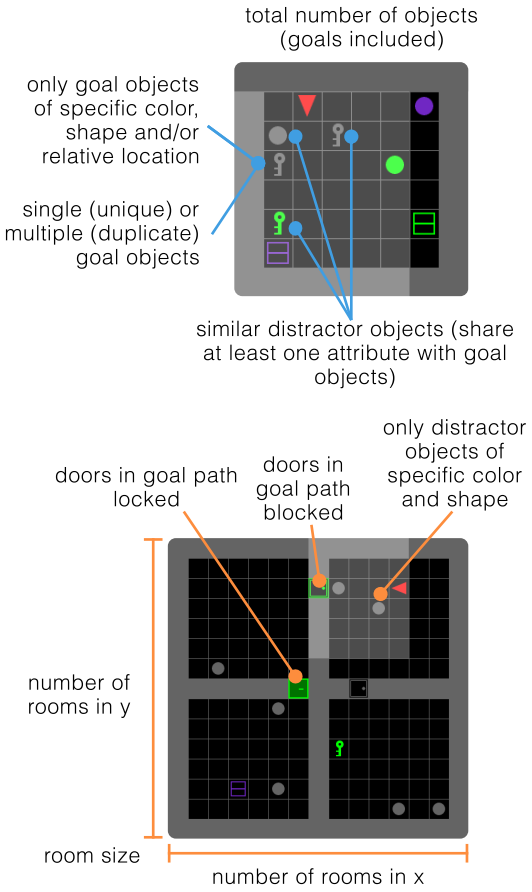


Figure D.1: An illustration of selected controllable parameters in our version of BabyAI.

C PPO baseline

The PPO baseline is trained on the equivalent amount of data as our multi-task IL models but only on single tasks (e.g. Pickup *or* PutNext). This corresponds to ~9K, ~18K and ~36K trajectories, and evaluated only on the equivalent evaluation task. We train five PPO models corresponding to the same global seeds used for our other models, and average their performance. Note that Chevalier-Boisvert et al. (2018) find that solving the comparable `PickupLoc` and `PutNext` tasks from the original BabyAI suite `using symbolic observations` requires ~1.4-1.6M and ~2.2-2.7M training episodes, respectively. Our findings show that the generalisation performance of the PPO baseline when trained on only up to 36K trajectories, corresponding to less than ~2.5% of data, is marginally below random performance. We use the hyper-parameters and acrhitecture corresponding to the implementation from the GitHub repository published for Chevalier-Boisvert et al. (2018)[7], but use the pixel-based observations and the corresponding image encoder.

D Environment details

BabyAI (Chevalier-Boisvert et al., 2018) builds on top of MiniGrid to procedurally generate so-called *missions* in abstract 2D grid worlds, where a mission is defined as the combination of a set of mission instructions and initial environment state (Chevalier-Boisvert et al., 2018).[8] The discrete action space consists of navigation and manipulation actions (`left`, `right`, `forward`, `pickup`, `drop`, `toggle`) and an optional `done` action. As we do not require the agent to indicate when it has finished a task, we constrain the action space to exclude the done action, which reduces the number of possible actions to $|X| = 6$. By default, BabyAI environments are partially observable, and observations are symbolic representations of a top-down view of the grid. BabyAI approximates an egocentric agent perspective by masking out tiles in the grid that are currently not in the agent's field of view because they are hidden behind walls or behind the agent.

[7] github.com/mila-iqia/babyai

[8] We use a similar definition to describe the more generic term of task instance.

Since we desire models to learn from pixels, we use an RGB image wrapper to map partial observations into pixel space.

The 19 original levels in BabyAI were designed to be used as a multi-task benchmarking suite and for curriculum learning, and evaluates agents on a range of skills. Besides understanding action language and goal object attributes, agents need to be able to navigate single rooms and multi-room mazes, unblock the path to doors, and guess to unlock doors during navigation. Additionally, some levels test the agent's understanding of spatial or temporal language: the agent may have to identify goal objects as specified by a location descriptor (e.g., 'pick up the blue key on your left'), or to understand composite instructions, whereby multiple instruction clauses are chained together in a sequence that may or may not require the instructions to be executed in a specific order (e.g., 'pick up the green ball after you pick up the purple box').

As BabyAI was not designed with compositional generalisation in mind, it does not expose the means to control environment parameters such that $p_{train} \neq p_{test}$. Additionally, due to the mechanism by which instructions and objects are sampled to procedurally generate missions, not all missions for a given level will test the skill introduced by the level. From initial experimentation, we find that understanding location language and composite instructions, as well as unblocking or unlocking doors, and even maze navigation are not strictly required for all missions of those levels meant to test the respective skill. While we assume that this is a deliberate design choice allowing a phasing in of skills for curriculum learning, it limits the level of control that can be exerted over the parameters of the environment further, and hinders the systematic evaluation of different skills.

As Kirk et al. (2021) argue, environments should be controllable and possess a structured parameter space to constitute a suitable test bed for most forms of compositional generalisation. As is the case with BabyAI, controllability is typically not supported for environments that facilitate procedural generation. Previous work attempting to use BabyAI for compositional generalisation (McCallum et al., 2023) achieves different training and test distributions by generating a large number of BabyAI missions and filtering for those with the desired goal object attributes. This approach scales poorly to larger datasets and beyond one or two parameters. We devise a more scalable approach and build on top of the BabyAI-MiniGrid ecosystem to develop BABYAI=XGEN, a version of BabyAI which gives users full control over a comprehensive range of environment parameters by simply passing a `config` object with the desired configuration when instantiating an environment with `gym.make`. We provide an exhaustive list of controllable parameters in Table D.1, and illustrate a selection of parameters in Figure D.1. As BabyAI is now integrated into Gymnasium (Towers et al., 2024), we will release BABYAI-XGEN following Apache 2.0 License.

E Dataset details

For the suboptimal trajectories, we use $p_1 = 0.5$ and $p_2 = 0.75$ to replace planner actions with random actions and achieve different degrees of suboptimality (see Equation (6). While we refer to the planner generated trajectories as *optimal*, we acknowledge that BFS does not guarantee optimality; however, BFS is efficient and there is no exploration.

$$A_t = \begin{cases} a, & \text{with prob. } 1-\rho, \\ x \sim \mathcal{U}(\mathcal{A}), & \text{with prob. } \rho \end{cases} \tag{6}$$

We construct a range of multi-task datasets with an equal number of trajectories per task, where the number of training tasks ranges from three for the experiments for compositionality in language, to nine for the experiments testing compositionality in the environment. We list the of the training and evaluation tasks, along with details on the configurations used to instantiate the tasks, in Table E.1. As (Kirk et al., 2021) note, a model that has seen a wide range of possible values for a parameter during training will be able to perform better on unseen values at test time than a model whose training data contained only one possible value for the parameter. We therefore ensure that we include examples of multiple possible parameter values in the training data. Likewise, we expose models to missions requiring skills of various degrees of difficulty, whereby harder skills build on top of harder skills (`Goto` < `Pickup` < `PutNext`). For mazes, we assume a similar relationship between single rooms and mazes (where single room < maze), and open, closed and locked doors (where open doors < closed doors < locked doors).

Parameter	Description
`room_size`	The number of cells in a room
`num_rows`	How many rooms a maze should have (in the y dimension)
`num_cols`	How many rooms the maze should have (in the x dimension)
`remove_walls`	Whether to combine multiple maze rooms by removing walls
`num_objs`	The total number of objects in a given room or maze (including goals and distractors/obstacles)
`duplicate_goals`	Whether to include multiple duplicate instances of the goal object, or only one unique goal object; affects article usage in instructions (e.g., "pickup the blue key" vs "pickup a blue key")
`dists_unique`	Whether all distractor objects should be unique
`dists_include_similar`	Whether to include distractor objects that share attributes with goal objects; adds a distractor for each specified goal object attribute (color, shape, location)
`avoid_overlapping_goals`	Whether to avoid sampling goal objects with overlapping attributes
`only_objs`	Permissible object attributes as lists of `(color, shape)` combinations for distractors and goals
`only_locations`	Similar to `only_objects` but for lists of `location`
`exclude_objects`	Disallowed object attributes as lists of `(color, shape)` combinations
`exclude_locations`	Similar to `exclude_objects` but for lists of `location`
`distinguish_by`	Whether to refer to goal objects by color, shape and/or location in instructions; `False` means never use that attribute, `True` enables random usage unless specified in `strict`
`action_kinds`	The action language available in mission instructions: can be one or multiple of `goto`, `open`, `pickup`, or `putnext`
`instr_kinds`	Whether to generate single instructions (`action`) or sequential instructions (`and`, `before`, `after`), or combinations
`seq_complexity`	Number of single instructions in sequences: `low` (two), `medium` (three), `high`, or combinations
`multiple_locations`	Controls location language for multiple goal objects: `True`, `'strictly'`, or `False`
`unblocking`	Controls path unblocking requirements: `False`, `True`, or `'strictly'` (maze-only)
`explicit_unlocking`	Whether goal doors need unlocking as additional goal condition (maze-only)
`implicit_unlocking`	Similar to `unblocking`, but for unlocking doors along goal path
`all_doors_open`	Whether all maze doors start open (maze-only)
`goal_room_same_as_start`	Controls goal object placement relative to start room: `False`, `True`, or `'strictly'` (maze-only)
`verify`	Optional verification settings: `'use_done_action'` or `'strict'` mode

Table D.1: Configuration parameters for maze generation and mission instructions

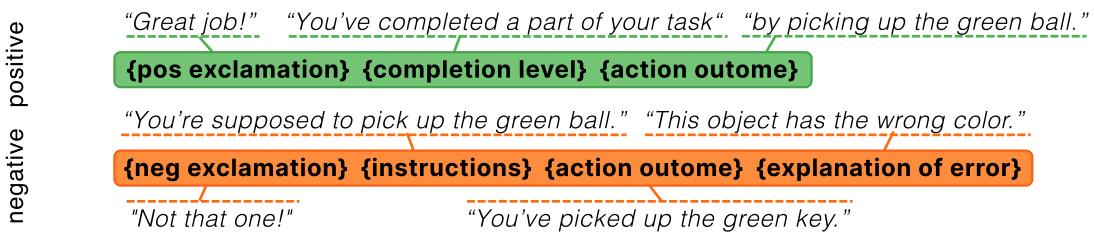


Figure F.1: We extend the task feedback oracle in McCallum et al. (2023) and provide positive and negative task feedback with templates. We sample exclamations from predefined options, the rest is specific to mission and action.

Compositionality in Instructions		
Train Datasets	Test Datasets	Important Configurations
Systematicity		
GoToColorType	GoToColorTypeUnseen	Train: Excludes yellow balls from goals and distractors
PickupColorType	PickupColorTypeUnseen	Test: Yellow balls as goals only, with similar distractors maintained
PutNextColorType	PutNextColorTypeUnseen1 PutNextColorTypeUnseen2	
Productivity (color)		
GoToOnlyColor	GoToOnlyColorUnseen	Train: Excludes yellow objects from goals and distractors
PickupOnlyColor	PickupOnlyColorUnseen	Test: Yellow objects as goals only, with similar distractors maintained
PutNextOnlyColor	PutNextOnlyColorUnseen1 PutNextOnlyColorUnseen2	
Productivity (location)		
GoToLoc	GoToLocUnseen	Train: Excludes 'left' location from goals
PickupLoc	PickupLocUnseen	Test: Only includes 'left' location for goals
PutNextLoc	PutNextLocUnseen PutNextLocUnseen2	
Productivity + Systematicity (seq)		
PickupEasySeqAnd	PickupMediumSeqBefore	Train: Only obstacles as distractors, with low sequence complexity
PickupEasySeqBefore	PickupMediumSeqAfter	Test: Only obstacles as distractors, with medium sequence complexity
PickupEasySeqAfter		
Compositionality in Environment		
Productivity		
Pickup	PickupN16	Base: Only includes obstacles as distractors
PickupMaze1X2	PickupMaze2X3	Regular: Goal in different room, all doors open
PickupMaze1X2Blocked	PickupMaze2X3Blocked	Blocked: Goal in different room, requires moving obstacles
PickupMaze1X2DoorsClosed	PickupMaze2X3DoorsClosed	Closed: Goal in different room, doors initially closed
PickupMaze1X2DoorsLocked	PickupMaze2X3DoorsLocked	Locked: Goal in different room, requires finding and using keys
PickupMaze2X2	PickupMaze3X3	
PickupMaze2X2Blocked	PickupMaze3X3Blocked	
PickupMaze2X2DoorsClosed	PickupMaze3X3DoorsClosed	
PickupMaze2X2DoorsLocked	PickupMaze3X3DoorsLocked	

Table E.1: Overview of environment configurations used to generate our multitask training datasets and test environments. Note that for our robustness experiments (see §5.2), we use the training configurations but instantiate the test environments with unseen seeds.

F Feedback details

To provide granular language feedback and denser scalar rewards for all conceivable tasks in our custom BabyAI, we extend the rule-based feedback oracles proposed in McCallum et al. (2023).

Task feedback. In line with (Xi et al., 2024a), we increase the informativeness of the language feedback compared with McCallum et al. (2023) and construct the feedback string according to an updated template, increasing diversity by sampling from a range of predefined options when populating the generic parts of the feedback template shown in Figure F.1.

Affordance feedback. We adopt the *rule feedback* oracle from McCallum et al. (2023) without any changes, but will refer to such language feedback, which is triggered when agents execute actions that have no visible effect on the environment due to its affordances, as *affordance feedback*. Such 'failed' actions include, for instance, the agent trying to move forward through a closed door, trying to open a wall, or attempting to put an object on top of another object (which is not permitted in BabyAI). A number of examples of task feedback and affordance feedback are provided in the appendix.

Shaped rewards. We leverage the success and action failure detection mechanisms in the feedback oracles to return intermediate scalar rewards. These shaped rewards replace the default reward function in BabyAI, which only provides a sparse reward signal in the form of terminal, binary rewards as shown in Equation (7), where γ is the discount factor, n the number of steps the agent took, and N_T the step budget for a given task in BabyAI.

$$R(T) = \begin{cases} 1 - \gamma \cdot \frac{n}{N_T} & \text{if task success} \\ 0 & \text{otherwise} \end{cases} \tag{7}$$

. This was informed by initial experiments showed only marginal differences between the IL case and return-to-go conditioning with the original binary rewards (see Table F.1), as the returns-to-go are identical for every timestep except the last. Furthermore, they are identical for all training trajectories, since we are not including trajectories for failed episodes. According to the revised reward function, agents receive a fraction of the terminal reward based on the total number of subgoals in the task each time they complete a subgoal; for each failed action, we assign a small negative reward. The negative action failure rewards can be combined with a binary terminal reward, or the dense subgoal rewards.

Model	Success Rate (%)
NONE	12.7
SCALAR (BINARY)	13.7
SCALAR (SHAPED)	69.5

Table F.1: Success rate (%) when using no feedback (NONE), sparse returns-to-go (SCALAR (BINARY)) and dense returns to go (SCALAR (SHAPED))

Note that the target return during testing is set to the terminal reward ($G = 1.0$).

$$R(G) = \begin{cases} \frac{1}{|G|} & \text{if subgoal achieved} \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$R(F) = \begin{cases} -0.01 & \text{if failed action} \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$R(T) = \begin{cases} 1 & \text{if task success} \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

G Experimental framework

Baselines and variants. Our flexible architecture as described in §3 serves as the basis for all our baselines and variants; we simply unmask additional token modalities for *scalar feedback* (returns-to-go) and/or *language feedback* and optionally predict additional tokens, specifically the scalar rewards and language feedback. For all model types, we initialise five models using different random seeds. We use the same hyperparameters across models in the same model family, that is, models that use the same feedback signal; note that preliminary hyperparamter-focused experiments, show that slightly different batch sizes and learning rates are best for the baseline, the variant using language feedback, and the variants using scalar rewards. The hyperparameter choices can be found in Table G.1. Note that rather than sub-trajectories of a given context length, all models use the full trajectory as context; trajectories are padded to the length of the longest trajectory in the batch. Despite trying different model sizes following Team et al. (2024), we report results of the BASE size for all our model variants.

Hyperparameter	Value
Optimizer	AdamW
Weight Decay	0.0001
Gradient Clip Threshold	0.25
Learning Rate	1e-5*
Batch Size	32*
Max training epochs	20
Early stopping min delta	0.01
Early stopping monitor	action loss (val)
Early stopping mode	min
Early stopping patience	2 epochs
Total trajectories seen	33,684 **
Examples per optimizer step	128

Table G.1: Hyperparameters using during model training. *For the variant with the language feedback (LANGUAGE), we use lr=2.5e-5, and for the variants using scalar feedback (SCALAR and COMBINED), we use lr=5e-5 and a batch size of 64. **For the experiments testing compositionatily in the environment, models will have seen 110,052 trajectories. Note that the length of the trajectories varies quite significantly, and we refrain from providing an estimate for the number of transitions seen.

Computing infrastructure and computational budget. We trained and evaluated all models using up to six NVIDIA A40 GPUs with 48GB, with runs taking around 3-8 hours when training on multi-task datasets with three tasks, and approximately 25 hours when training on multi-task datasets spanning nine tasks. Evaluation runs took between 1.5 and 7.5 hours, depending on the complexity and number of evaluation tasks, as well as the quality of the obtained models[9]. In a total, the required compute budget for this work given the hardware used is 2,000 GPU hours.

H Further experimental details and results

All results are averaged across 128 missions per evaluation task and 5 different model seeds. Note that we use designated evaluation configurations for our compositional generalisation experiments; for all other experiments, we instantiate the environments according to the training configurations but with unseen seeds to ensure that agents have to generalise to unseen task instances, while disentangling the results from skills required for compositional generalisation.

[9]Models with poor performance took longer to evaluate than models with fair or good performance, as performance translates into the number of evaluation episodes that lasted for the maximum number allowed steps

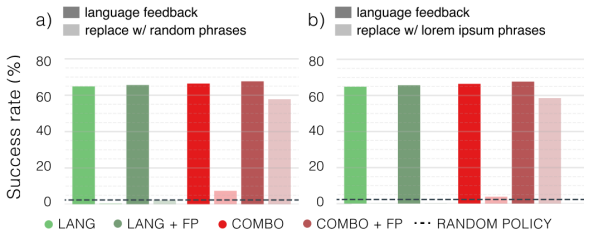


Figure H.1: Comparison of success rates under additional adversarial language feedback evaluation settings. We average performance on in-distribution Pickup and PutNext tasks. From left to right: a) replace feedback with random sentences, d) replace feedback with lorem ipsum sentences. ST=suboptimal trajectories, FP=feedback prediction

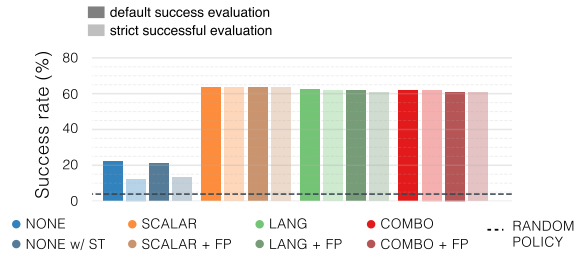


Figure H.2: Comparison of success rates when interactions with non-goal objects are allowed vs. when they lead to early termination (task failure). We average performance on in-distribution Pickup and PutNext tasks. Note this refers to the inference scnenario. ST=suboptimal trajectories, FP=feedback prediction.

External perturbations. We adapt *sticky actions* from Machado et al. (2017), who introduced this setting to introduce non-deterministic behaviour in the Atari Learning Environment (Bellemare et al., 2012). At every time step, instead of the agent's current action, the environment executes the agent's previous action again with a probability defined by stickiness parameter ζ. Concretely, at time step t the agent chooses an action a and takes a step in the environment with action A_t. The full equation is given in the appendix.

$$A_t = \begin{cases} a, \text{ with prob. } 1-\zeta, \\ a_{t-1}, \text{ with prob. } \zeta. \end{cases} \tag{11}$$

11 According to this, sticky actions may be executed for k consecutive time steps with a probability of $(1-\zeta)^k\zeta$.

Adversarial feedback. We test in total three adversarial scenarios, whereby we: 1) provide random English sentences at random timesteps, or

replace language feedback with 2) random Lorem Ipsum sentences, or 3) random English sentences. Note that for English sentences, we avoid sampling words which occur in the actual language feedback. We consider case 1), the results for which are reported in §5.2, the most adversarial scenario, as feedback is neither semantically relevant nor timely. A similar setting to 2) and 3) has been explored in Parekh et al. (2024), who perturb instructions into *Gobbledygook Words*. Instead, we explicitly choose adversarial feedback that is similar to the actual feedback. We show cases 2) and 3) in Figure H.1, and can identify no noticeable difference for the baseline and the models trained on both language feedback and scalar feedback between the two cases, and only minimal difference for the models trained only on language feedback. We note that the performance of the most robust model is slightly worse when feedback is missing (see Figure 4d), compared with the adversarial cases.

Solution Efficiency. We calculate our oracle-path normalised path lengths as shown in Equation Equation (12), where L_i^* is the number of steps in the expert demonstration for a given task instance, and $\hat{L}_i$ the number of steps it took the model to complete the task, averaged across all successful episodes. This is inspired by the path-weighted success rate in Shridhar et al. (2019), which combines the overall success rate with the efficiency of successful paths. We choose to disentangle these two aspects of success and provide them as separate metrics, specifically success rate and partial success rate across all evaluation episodes, and oracle-path normalised path lengths only for successful episodes.

$$p_o n = \frac{1}{N} \sum_{i \in \mathcal{S}}^{N} \frac{L_i^*}{\hat{L}_i} \tag{12}$$

Strict Task Success. As an additional setting, we test whether models trained with language feedback learn behaviours that see them unnecessarily interacting with distractor objects to elicit feedback, which may be undesirable during deployment and results in less efficient solutions. Using the *strict* evaluation setting, which terminates episodes early as failed when the agent interacts with (picks up) distractor objects. Figure H.2 shows that when evaluating models in *strict* mode, performance does not deteriorate noticeable for models trained with access to feedback signals, while the performance of the baseline drops to 70% of the original performance when trained only on optimal trajectories, and 65% when trained on both optimal and suboptimal trajectories, which indicates that the baseline exhibits more inefficient trial-and-error behaviour at inference.

Model	Success Rate (%)
NONE	14.1
NONE (FP)	14.4

Table H.1: Success rate (%) of the baseline trained without any additional feedback (NONE) signal with and the auxiliary task of predicting the next image (FP).

Image prediction loss. We find that models that can predict the embedding of the next image observation, which could be considered *visual feedback*, do not tend to outperform models that only predict the next action (see Table H.1). We hypothesize that this is in part due to the abstract nature of the grid world and the birds eye view, where there is little variation in the observations, with the background and objects being represented as solid colour blocks. Since objects take up at most one cell and in combination with the discrete actions, object interactions will never change more than 2 cells (or 3% of the visible pixels assuming an 8x8 room) when the agent is moved forward by one cell, or 1 cell when an object disappears or reappears as a result of a pickup or drop action, and can be as minimal as a few pixels when the triangle of the agent is rotated 90 degrees as a result of a left or right navigation action. Conversely, due to the partially observable nature of the environment and the discrete nature of the action space, navigation actions, particularly turning left or right, may result in a considerable part of the environment abruptly changing from visible to obscured or vice versa, whereby the cells that are not visible to the agent will simply be masked out. A more realistic observation space with a first person view and more fine-grained actions may be a better fit for testing the potential of leveraging the next observation as visual feedback similar to human learning.

Model scale. To validate our architectural choices, we conduct experiments examining how model capacity affects performance across different feedback modalities. We train models of varying sizes—tiny (~10M), small (~30M), and base (~90M) parameters—while keeping training data constant. The results, shown in Table H.2,

Feedback	Tiny (~10M)	Small (~30M)	Base (~90M)
NONE	11.9%	14.5%	12.7%
SCALAR	64.7%	68.4%	69.6%
LANG	60.9%	67.6%	67.7%
COMBO	67.6%	69.1%	70.4%

Table H.2: Success rate (%) across different model sizes, showing plateauing beyond a certain capacity threshold.

demonstrate that performance plateaus beyond a certain capacity threshold across all feedback conditions. These findings indicate that BabyAI-XGen presents fundamental challenges that cannot be addressed through increased model capacity alone. The performance bottleneck appears to stem from the compositional generalization requirements rather than insufficient model capacity. Beyond the tiny-to-small transition, additional parameters provide diminishing returns, suggesting that the core challenge lies in learning compositional reasoning patterns rather than requiring larger representational capacity. This behavior aligns with the lottery ticket hypothesis (Frankle and Carbin, 2018), which demonstrates that overparameterized networks contain sparse subnetworks achieving comparable performance, indicating that much additional parameter space remains unused for the specific task requirements. In our compositional generalization setting, the fundamental difficulty lies in developing reasoning capabilities for novel task combinations rather than memorizing complex patterns that would benefit from increased capacity. Our approach prioritizes isolating the fundamental mechanisms of learning from suboptimal trajectories with feedback. While pretrained language models represent an important direction for future work—particularly regarding how pretraining might qualitatively affect feedback processing—our current architecture provides sufficient capacity to demonstrate the effectiveness of our approach while maintaining experimental clarity and avoiding confounding factors from pretrained knowledge.

I Reproducibility

To meet the high reproducibility standards in the ML research community, a fully reproducible training and evaluation framework is available via github.com/sabraaap/fossil. This includes implementations for all experiments carried out. Models were trained using PyTorch (Ansel et al., 2024) and Lightning (Falcon and The PyTorch Lightning Team, 2024), and dependencies were tracked using Poetry. Experiments were managed via Hydra configuration files (Yadan, 2019), and all configurations, commands, hyperparameters, and seeds used are available and signposted clearly in the provided codebase.

Environment. We register BabyAI-XGen, our controllable environment based on BabyAI developed for generalisation research, as a Gymnasium environment and provide several predefined configuration objects to reproduce not only the environments used to generate our datasets and evaluate our models, but also blueprints for versions of the existing BabyAI level suite. This can be accessed at github.com/sabraaap/fossil.

Training data. All training data has been made available via Hugging Face Datasets repositories huggingface.co/fossil-eai/datasets. In addition, training data can be regenerated using the dataset generation scripts which is available in the provided codebase at github.com/sabraaap/fossil.

Model checkpoints. All model checkpoints are provided on the Hugging Face Hub at huggingface.co/fossil-eai/models to facilitate further experiments and explorations.