Is Feedback All You Need? Leveraging Natural Language Feedback in Goal-Conditioned Reinforcement Learning

Sabrina McCallum[1,2], **Max Taylor-Davies**[1,2], **Stefano V. Albrecht**[1], **Alessandro Suglia**[2]

[1]University of Edinburgh [2]Heriot-Watt University

Abstract

Despite numerous successes, the field of reinforcement learning (RL) remains far from matching the impressive generalisation power of human behaviour learning. One way to help bridge this gap may be to provide RL agents with richer, more human-like feedback expressed in natural language. First, we extend BabyAI to automatically generate language feedback from the environment dynamics and goal condition success. Then, we modify the Decision Transformer architecture to take advantage of this additional signal. We find that training with language feedback either in place of or in addition to the return-to-go or goal descriptions improves agents' generalisation performance, and that agents can benefit from feedback even when this is only available during training, but not at inference.

1 Introduction

Despite significant advances over decades of research, modern AI systems still lag significantly behind the learning abilities of humans. During their development, human infants develop a wide range of adaptive behaviours through an open-ended learning process that is remarkable for both its sample efficiency and generalisation [1]. One likely contributing factor is that, while infant learners do engage in trial-and-error learning, they are also able to draw from a variety of feedback sources beyond their immediate environment. One such source is other humans, who may provide them with additional feedback signals in the form of natural language [2; 3]. Sometimes, this feedback may be akin to a classic reward signal, such as a parent praising (or scolding) their child for doing something right (or wrong). But it can also be richer and more structured, conveying information directly tailored to the learner's current goal, such as explanations of specific observations and events. This may allow the learner to update their prior knowledge and build a more stable and accurate model of the world [4; 5].

While they are able to achieve superhuman performance on some tasks, reinforcement learning (RL) agents typically struggle with both sample efficiency and generalisation, especially in settings where the environment reward is sparse or under-specified [6]. Additionally, there may be environment–task combinations for which no sufficiently expressive Markov reward function exists, or, in the case of tasks specified in language, there may be a mismatch in abstraction between task and reward [7].

To address this, we investigate whether RL agents may be able to learn more generalisable policies when introduced to richer and more human-like feedback signals expressed through natural language. We modify an existing offline RL method to condition on different types of language feedback provided by the environment. Our method includes a procedure to automatically generate this language feedback, and requires no humans in the loop. We find evidence that conditioning on language feedback can boost generalisation performance relative to baselines conditioning only on goal instructions or return, and can match or outperform these baselines when goal instructions or return are not available, even when feedback is not provided at inference time.

37th Conference on Neural Information Processing Systems (NeurIPS 2023).

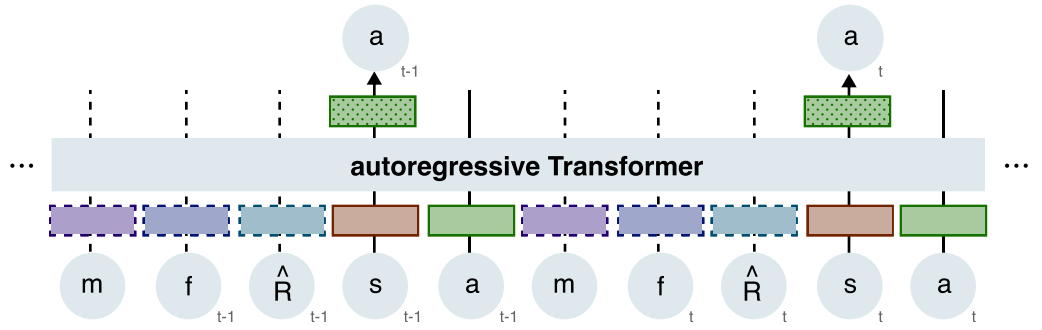

Figure 1: We extend the capabilities of the Decision Transformer (return-conditioned) and Text Decision Transformer (instruction-conditioned), with the option to condition on language feedback.

2 Related work

Reinforcement Learning from Human Feedback (RLHF). A related area of work can be found in RLHF, which describes a methodology for dynamically adapting machine learning models, such as large language models (LLMs) [8; 9; 10; 11] and more recently, vision-and-language models (VLMs) [12; 13], to hard-to-specify goals which are typically linked to human preferences or behaviours. Most similar to our work is Octopus [12], which leverages automatic feedback from the simulator instead of human supervision. Both RLHF and our work use auxiliary feedback information to aid learning where the desired behaviour may be difficult to capture in a simple reward signal. However, while in RLHF, learning from feedback typically occurs as a separate subsequent process, we consider it as part of the main training procedure. Likewise, we do not use the feedback samples to train any explicit reward model, but pass them directly to the core behaviour-learning model.

Feedback in LLM prompts. A growing body of work prompts pretrained LLMs or VLMs to generate plans for a range of robotic manipulation and embodied AI tasks. Some of these studies incorporate a form of language feedback on the generated plan into the prompt. The feedback used is typically automatic and ranges from simple binary task completion feedback [14] to more verbose feedback messages from the environment and execution errors [15; 16; 17; 12]. Concurrent work using feedback for language-to-code generation [18; 19] and chain-of-thought reasoning [20; 21; 22] leverages compiler errors and human feedback, respectively. While the former inspired our notion of feedback and the mechanism used to generate it, we do not make use of pretrained models or in-context learning, and apply feedback specifically in the context of goal-conditioned RL.

3 Proposed method

Architecture. We build upon the Decision Transformer [23], which casts RL as a sequence modelling problem, where behaviour is produced by generating action sequences in an auto-regressive manner and conditioned on the desired return. The Transformer [24] has emerged as the architecture of choice for pre-training LLM's and VLM's, and has been shown to be competitive on offline RL benchmarks thanks to its flexibility w.r.t input encoding, and its ability to condition on previous timesteps over long contexts through the self-attention mechanism. As a variant of the original DT, the Text Decision Transformer (TDT) [25] conditions action generation on language goal instructions. Our architecture (see Figure 1) extends both methods by allowing action generation to be conditioned on return-to-go (RTG), goal instructions, language feedback, or any combination thereof.

Augmenting environments with language feedback. Our approach includes a method to automatically generate low-level language feedback using predefined rules and templates. We conceptualise two feedback types, 'Rule Feedback' and 'Task Feedback', which capture different information about the consequences of the agent's actions, as is illustrated in Figure 2. For our Task Feedback, we decompose high-level goal instructions into granular sub-goals in order to generate feedback on the agent's progress towards the goal. Rule Feedback is provided when an action is executed that violates any of the physical constraints or predicates imposed by the environment. We conceptualise this as a type of corrective feedback extended with a detailed explanation for the failure. For further details on feedback generation, the reader may refer to Appendix A.1.

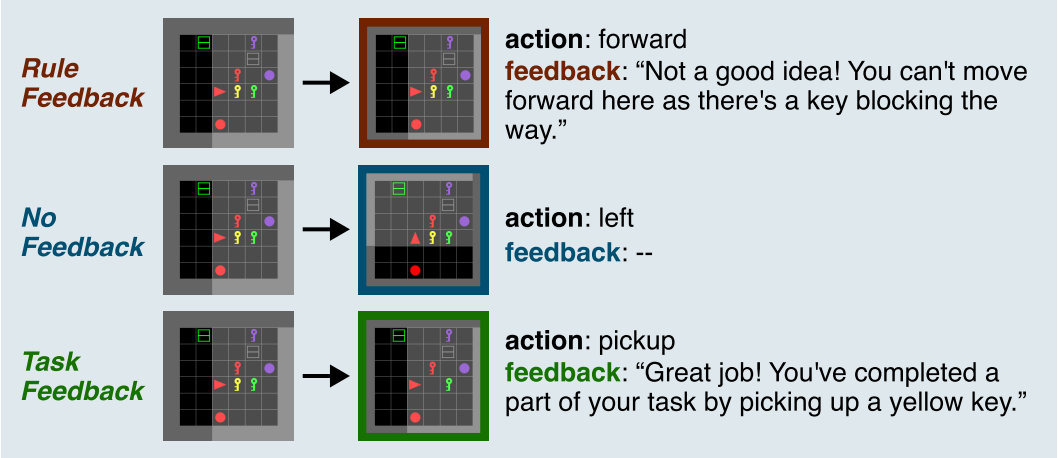

Figure 2: Different actions can result in either Rule Feedback, no feedback, or Task Feedback.

4 Experiments

To test our approach, we extend BabyAI [26], a suite of 2D gridworld environments which facilitates the training of agents on goal-oriented tasks specified in language. Levels in BabyAI increase in difficulty, and range from simple single rooms to complicated mazes with many objects. Harder levels involve multiple sub-goals, long horizons and complex compositional instructions. All levels in BabyAI pose challenges associated with sparse rewards. We select a subset of the levels including single rooms and mazes, and use a random policy to generate offline training datasets for each level using partial image observations. By default, BabyAI environments include high-level language goal instructions, or 'missions', and we augment this with language feedback as described in Section 3.

We compare the relative performance of different variants of our feedback-conditioned model against baselines which condition only on RTG or only on mission, and investigate a) whether conditioning on language feedback in addition to RTG or mission boosts the generalisation performance of the baselines, b) whether we can exceed, or at least match, the generalisation performance of the baselines by relying solely on language feedback, and c) how providing feedback not only during training but additionally at inference impacts generalisation performance. As our Task Feedback corresponds to goal conditions, we use goal-conditioned success rate as the evaluation metric, and devise an evaluation protocol for compositional generalisation for both IID and OOD scenarios. For implementation details and all results, see Appendices A.2 and A.3, respectively.

Exp 1: Combining RTG with Task Feedback boosts performance on single-room levels. Since Rule Feedback seems to negatively impact performance on maze levels when combined with the RTG, we hypothesise that while this dense feedback discourages unnecessary behaviour that has no effect on the environment, thereby reducing the risk of the environment timing out in easier levels with lower max steps, it is less helpful in harder maze levels, where the agent may try to complete the task for longer, and potentially distracts from the goal-relevant behaviour encouraged by the RTG.

Exp 2: Combining mission with Rule Feedback boosts performance on maze levels. In addition, we observe that combining the mission with only the Task Feedback can be detrimental for performance. Unlike Rule Feedback, which is explicitly linked to the effect of the previous action, Task Feedback encapsulates the cumulative effect of the full sequence of actions since the previous goal condition, and we hypothesise that when presented with the mission and the Task Feedback together, the model has to learn to simultaneously leverage two long-horizon signals at the same time.

Exp 2: Replacing RTG with feedback improves OOD performance. We find that performance improvements for both OOD and IID scenarios are more significant the lower the performance of the RTG-only baseline is, as well as when the relative goal location is OOD, which may be an indication that language feedback captures transferable, higher-level information where behaviour learned on the basis of a numerical reward is perhaps too specific to the training configurations and tasks.

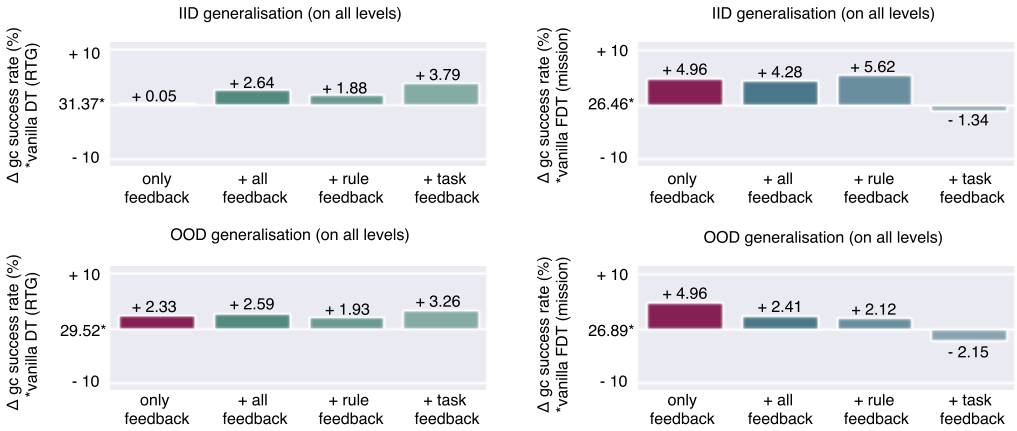

Figure 3: Difference in performance between the return-only (DT) and mission-only (TDT) baselines and our proposed variants that condition with feedback.

Exp 4: Replacing mission with feedback improves performance. In the case of OOD performance, we observe the most significant improvements when the performance of the baseline is poor and the model seemingly fails to learn a generalisable mapping to task-relevant behaviour from the mission alone, as well as for test environments where the model has to generalise to unseen relative goal locations or fixed goal objects, both of which are specified in the final part of the mission.

Exp 5: Feedback at inference is only useful when performance is otherwise poor. Apart from the feedback-only variant, the performance change with feedback at inference appears to be inversely proportional to the performance without feedback at inference. While this behaviour merits further investigation, we hypothesise that the distribution of feedback encountered at inference is unseen in training episodes which negatively affects the agent's performance.

5 Limitations and future work

While we limit ourselves to demonstrating the potential of language feedback for goal-conditioned RL on a single algorithm and learning environment, we are optimistic that the underlying idea can be transferred successfully to other RL algorithms and simulation environments beyond 2D gridworlds—we aim to explore this in future work. Replacing templates with more diverse language, e.g. generated with LLM prompting, may provide the additional flexibility and scalability required for transfer to more diverse task, action and observation spaces, without having to rely on humans in the loop. As our feedback is generated automatically and does not rely on human annotators, it should translate directly to the online setting in the scope of simulated learning environments. It remains to be seen if pre-training agents in this fashion would be sufficient for successful sim-to-real transfer; however, we believe that providing free-form language feedback to robots deployed in the real world would be intuitive for human collaborators. Exploring the implications of this approach to Human-Robot-Interaction is an interesting and promising avenue for future research.

6 Conclusion

We investigate the potential of using automatically generated language feedback to train agents in sparse-reward environments with language-specified goals. We find evidence that conditioning on such feedback in addition to goal instructions or desired return can yield significant improvements in generalisation to environments that are IID or OOD with respect to various factors. Within the BabyAI environment suite, feedback seems to provide a useful alternative or complementary signal to desired return in easy levels, and to goal instructions in harder levels. Additionally, we establish that language feedback can potentially serve as an alternative condition when goal instructions or desired return are not available, and that in some cases, feedback is only effective when it is also provided at inference time.

References

[1] J. R. Saffran, R. N. Aslin, and E. L. Newport, "Statistical learning by 8-month-old infants," *Science*, vol. 274, no. 5294, pp. 1926–1928, 1996. [Online]. Available: http://www.jstor.org/stable/2891705

[2] J. A. C. Hattie and H. S. Timperley, "The power of feedback," *Review of Educational Research*, vol. 77, pp. 112 – 81, 2007. [Online]. Available: https://api.semanticscholar.org/CorpusID:82532100

[3] L. E. Schulz, "The origins of inquiry: inductive inference and exploration in early childhood," *Trends in Cognitive Sciences*, vol. 16, pp. 382–389, 2012. [Online]. Available: https://api.semanticscholar.org/CorpusID:17346427

[4] T. Miller, "Explanation in artificial intelligence: Insights from the social sciences," *Artificial intelligence*, vol. 267, pp. 1–38, 2019.

[5] T. Lombrozo, "The structure and function of explanations," *Trends in cognitive sciences*, vol. 10, no. 10, pp. 464–470, 2006.

[6] J. M. C. Ocana, R. Capobianco, and D. Nardi, "An overview of environmental features that impact deep reinforcement learning in sparse-reward domains," *Journal of Artificial Intelligence Research*, vol. 76, pp. 1181–1218, 2023.

[7] D. Abel, W. Dabney, A. Harutyunyan, M. K. Ho, M. L. Littman, D. Precup, and S. Singh, "On the expressivity of markov reward," *ArXiv*, vol. abs/2111.00876, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:240354171

[8] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe, "Training language models to follow instructions with human feedback," *ArXiv*, vol. abs/2203.02155, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246426909

[9] J. Scheurer, J. A. Campos, J. S. Chan, A. Chen, K. Cho, and E. Perez, "Training language models with language feedback," 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:248965479

[10] N. Stiennon, L. Ouyang, J. Wu, D. M. Ziegler, R. J. Lowe, C. Voss, A. Radford, D. Amodei, and P. Christiano, "Learning to summarize from human feedback," *ArXiv*, vol. abs/2009.01325, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221665105

[11] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *ArXiv*, vol. abs/1909.08593, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:202660943

[12] J. Yang, Y. Dong, S. Liu, B. Li, Z. Wang, C. Jiang, H. Tan, J. Kang, Y. Zhang, K. Zhou, and Z. Liu, "Octopus: Embodied vision-language programmer from environmental feedback," 2023.

[13] Z. Sun, S. Shen, S. Cao, H. Liu, C. Li, Y. Shen, C. Gan, L. Gui, Y.-X. Wang, Y. Yang, K. Keutzer, and T. Darrell, "Aligning large multimodal models with factually augmented rlhf," *ArXiv*, vol. abs/2309.14525, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:262824780

[14] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. R. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, N. Brown, T. Jackson, L. Luu, S. Levine, K. Hausman, and B. Ichter, "Inner monologue: Embodied reasoning through planning with language models," in *Conference on Robot Learning*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:250451569

[15] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. D. Reid, and N. Sünderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable task planning," *ArXiv*, vol. abs/2307.06135, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:259837542

[16] M. Skreta, N. Yoshikawa, S. Arellano-Rubach, Z. Ji, L. B. Kristensen, K. Darvish, A. Aspuru-Guzik, F. Shkurti, and A. Garg, "Errors are useful prompts: Instruction guided task programming with verifier-assisted iterative prompting," *ArXiv*, vol. abs/2303.14100, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257757298

[17] G. Wang, Y. Xie, Y. Jiang, A. Mandlekar, C. Xiao, Y. Zhu, L. J. Fan, and A. Anandkumar, "Voyager: An open-ended embodied agent with large language models," *ArXiv*, vol. abs/2305.16291, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:258887849

[18] X. Chen, M. Lin, N. Schärli, and D. Zhou, "Teaching large language models to self-debug," 2023.

[19] A. Ni, S. Iyer, D. R. Radev, V. Stoyanov, W. tau Yih, S. I. Wang, and X. V. Lin, "Lever: Learning to verify language-to-code generation with execution," *ArXiv*, vol. abs/2302.08468, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:256900680

[20] S. An, Z. Ma, Z. Lin, N. Zheng, J.-G. Lou, and W. Chen, "Learning from mistakes makes llm better reasoner," *ArXiv*, vol. abs/2310.20689, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:264813981

[21] B. Peng, M. Galley, P. He, H. Cheng, Y. Xie, Y. Hu, Q. Huang, L. Lidén, Z. Yu, W. Chen, and J. Gao, "Check your facts and try again: Improving large language models with external knowledge and automated feedback," *ArXiv*, vol. abs/2302.12813, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:257205781

[22] A. Madaan, N. Tandon, P. Clark, and Y. Yang, "Memory-assisted prompt editing to improve gpt-3 after deployment," in *Conference on Empirical Methods in Natural Language Processing*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253236860

[23] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and I. Mordatch, "Decision transformer: Reinforcement learning via sequence modeling," in *Neural Information Processing Systems*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235294299

[24] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Neural Information Processing Systems*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:13756489

[25] A. L. Putterman, K. Lu, I. Mordatch, and P. Abbeel, "Pretraining for language conditioned imitation with transformers," 2022. [Online]. Available: https://openreview.net/forum?id=eCPCn25gat

[26] M. Chevalier-Boisvert, D. Bahdanau, S. Lahlou, L. Willems, C. Saharia, T. H. Nguyen, and Y. Bengio, "Babyai: A platform to study the sample efficiency of grounded language learning," in *International Conference on Learning Representations*, 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:59536625

[27] J. Lin, Y. Du, O. Watkins, D. Hafner, P. Abbeel, D. Klein, and A. D. Dragan, "Learning to model the world with language," *ArXiv*, vol. abs/2308.01399, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:260438420

[28] L. Ruis, J. Andreas, M. Baroni, D. Bouchacourt, and B. M. Lake, "A benchmark for systematic generalization in grounded language understanding," *ArXiv*, vol. abs/2003.05161, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:212658007

[29] T. Cao, J. Wang, Y. Zhang, and S. Manivasagam, "Zero-shot compositional policy learning via language grounding," 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:258180211

[30] C. Heinze-Deml and D. Bouchacourt, "Think before you act: A simple baseline for compositional generalization," *ArXiv*, vol. abs/2009.13962, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:221995908

[31] Z. Wu, E. Kreiss, D. C. Ong, and C. Potts, "Reascan: Compositional reasoning in language grounding," *ArXiv*, vol. abs/2109.08994, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:237572003

[32] A. Sikarwar, A. Patel, and N. Goyal, "When can transformers ground and compose: Insights from compositional generalization benchmarks," *ArXiv*, vol. abs/2210.12786, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:253097723

[33] M. Aghzal, E. Plaku, and Z. Yao, "Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning," *ArXiv*, vol. abs/2310.03249, 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:263671594

[34] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533

[35] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Conference on Empirical Methods in Natural Language Processing*, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:201646309

A Appendix

A.1 Datasets

Since we train our models offline, we create datasets containing trajectories generated by a random policy for eight levels in BabyAI [26] which are listed in Table 1. For the single-room levels, we generate 10 trajectories for 128 environment instantiations each, and 10 times as many for the more complex maze levels. Trajectories are composed of sequences of mission string, partial image observation, action, reward and feedback string. Note that BabyAI only provides a sparse, terminal reward between 0 and 1 at the end of the episode, and no intermediate rewards. We make the training datasets used in our experiments available on https://www.dropbox.com/sh/b0bff46d4s230hr/AADJE4xu_Aliqd_mAv3kUdSda?dl=0.

We augment BabyAI to generate 'Task Feedback' and 'Rule Feedback'. Both types of feedback are rule-based and deterministic, so that the same rule will trigger the same string every time (see Tables 2- 3). The procedure is illustrated in Figure 4. BabyAI missions consist of one or multiple high-level actions (`go to`, `open`, `pick up`, or `put next`), each paired with goal objects, which are specified using color, type and location descriptors. Apart from the simple `goto`, these actions involve multiple steps; we decompose them into their component steps, which serve as sub-goals (or goal conditions) for the purpose of generating feedback and calculating the goal-condition success rate used to compare model performance. Note that we do not use these sub-goals as such as a learning signal. To illustrate this, the mission "put a yellow ball next to the green box" would be decomposed into four sub-goals: `go to a yellow ball`, `pick up a yellow ball`, `go next to the green box`, and `put a yellow ball next to the green box`.

Note that unlike the correction feedback in [27], our Rule Feedback does not rely on heuristic representations of rules outside of the dynamics of the simulator, and we avoid the use of instruction language in the feedback to clearly separate this from the goal instructions. In a similar vein to recent work in the space of LLMs for code generation [12] and planning [15], which leverages execution errors for actions as automatic feedback, we exploit BabyAI's internal action validation as the simulation environment does not return execution errors or similar system messages as such.

We evaluate generalisation on environments seeded with held out seeds both for in-distribution (IID) as well as out-of-distribution (OOD) scenarios. Our approach is inspired by work on compositional generalisation in grid worlds, such as gSCAN [28] and related benchmarks [29; 30; 31; 32; 33]. The attributes that we control for and their corresponding held-out values are described in Table 4.

We publish the code used to generate datasets and feedback, and to determine training, IID and OOD seeds on https://github.com/maxtaylordavies/feedback-DT.

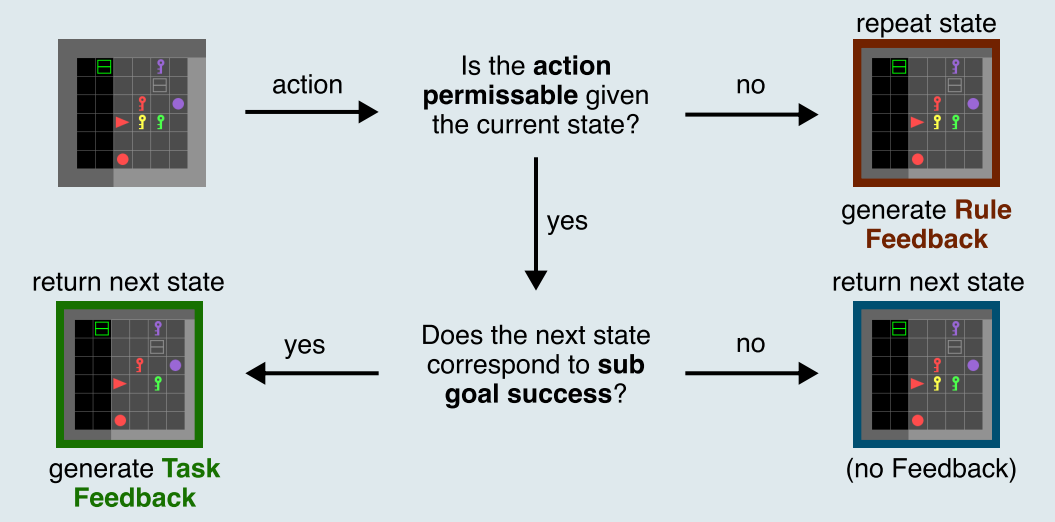

Figure 4: The feedback generation process for BabyAI environments.

Table 1: BabyAI levels used in our experiments. The top four levels are single rooms, the bottom four mazes. GC's = Goal Conditions. Unlike GoToLocal, GoToObj has no distractor objects. The code used to generate the datasets can be found on https://github.com/maxtaylordavies/feedback-DT.

Level	Mission space	GC's	Steps$_{max}$	Episodes
GoToObj	`go to {the/a} {col} {type}`	1	64	1,280
GoToLocal	`go to {the/a} {col} {type}`	1	64	1,280
PickupLoc	`pick up {the/a} {col} {type} {loc}`	2	64	1,280
PutNextLocal	`put {the/a} {col1} {type1} next to {the/a} {col2} {type2}`	4	128	1,280
Pickup	`pick up {the/a} {col} {type}`	2	576	12,800
PutNext	`put {the/a} {col1} {type1} next to {the/a} {col2} {type2}`	4	1,152	12,800
Synth	`go to {the/a} {col} {type}`, `pick up {the/a} {col} {type}`, `open {the/a} {col} door`, `put {the/a} {col1} {type1} next to {the/a} {col2} {type2}`	1-4	1,152	12,800
SynthLoc	Same as Synth, but with `loc` language	1-4	1,152	12,800

Table 2: Feedback templates and their corresponding rules, which check whether the effect of an action corresponds to the current goal condition. Unlike the other instruction types, GoNextTo is not a default BabyAI instruction type and used exclusively as a sub-goal for PutNext instructions.

Instruction type	Action	Goal condition	Feedback template
`GoTo`	`forward` `left` `right`	Goal(FrontCell)	"Fantastic! You've completed {a part of }your task by going to {goal object description}."
`GoNextTo`	`forward` `left` `right`	NextTo(FrontCell, Goal)	"That's right! You've completed {a part of }your task by going next to goal object description."
`Open`	`toggle`	Goal(FrontCell) $\wedge$ OpenDoor(FrontCell)	"That's correct! You've completed {a part of }your task by opening {goal door description}."
`Pickup`	`pickup`	Carrying(Object) $\wedge$ Goal(Object)	"Great job! You've completed {a part of }your task by picking up {goal object description}."
`PutNext`	`drop`	NextTo(FrontCell, FixedGoal) $\wedge$ MoveGoal(FrontCell)	"That's right! You've completed {a part of }your task by going next to goal object description."

Table 3: Feedback templates and their corresponding rules, which check whether an action has violated any of the pre-conditions for the action to have an effect on the environment.

Action	Condition	Feedback template
`forward`	Wall(FrontCell)	"Not a good idea! You can't move forward while you're facing the wall."
`forward`	Object(FrontCell) $\wedge$ $\neg$Door(FrontCell)	"Not a good idea! You can't move forward here as there's a {object} blocking the way."
`forward`	Door(FrontCell) $\wedge$ Closed(FrontCell)	"Not a good idea! You can't move forward here as the door in front of you is closed."
`forward`	Door(FrontCell) $\wedge$ Locked(FrontCell)	"Not a good idea! You can't move forward here as the door in front of you is locked."
`pickup`	Wall(FrontCell)	"Not a good idea! You can't pick up the wall."
`pickup`	Empty(FrontCell)	"Not a good idea! There's nothing in front of you, and you can't pick up empty space."
`pickup`	Door(FrontCell)	"Not a good idea! You can't pick up doors."
`pickup`	Object(Carrying)	"Not a good idea! You can't pick up another object while you're already carrying one."
`drop`	Wall(FrontCell)	"Don't do that! You can't drop an object while you're facing the wall."
`drop`	Object(FrontCell) $\wedge$ $\neg$Door(FrontCell)	"Don't do that! You can't drop an object on top of another object, and there's already a {object type} in front of you."
`drop`	Door(FrontCell)	"Don't do that! You can't drop an object while you're facing a door."
`drop`	Empty(Carrying)	"Don't do that! You're not carrying any object so dropping has no effect."
`toggle`	Wall(FrontCell)	"That won't work here. You can't open the wall."
`toggle`	Object(FrontCell) $\wedge$ $\neg$Box(Object)	"That won't work here. You can't open {object type}s."
`toggle`	Empty(FrontCell)	"That won't work. There's nothing in front of you, and you can't open empty space."
`toggle`	Door(FrontCell) $\wedge$ Locked(FrontCell) $\wedge$ $\neg$Key(Carrying)	"That won't work here. You can't open a locked door without a key of the same color as the door, and you're not carrying any key."
`toggle`	Door(FrontCell) $\wedge$ Locked(FrontCell) $\wedge$ Key(Carrying) $\wedge$ $\neg$SameCol(Carrying, FrontCell)	"That won't work here. You can't open a locked door without a key of the same color as the door. You're carrying a {key color} key, but the door in front of you is {door color}."

Table 4: Attributes and held out values used to evaluate OOD generalisation in our experiments. *Fixed goal object and relative goal location are only applicable to levels which include PutNext instructions, and those with location language (Loc), respectively.

Out-of-domain attribute	Held out value
Goal object	yellow box
Fixed goal object*	blue ball
Relative goal location*	on your right
Room size	< 8 x 8 tiles
Agent starting location in room	bottom left (in room) / bottom right (in maze)

A.2 Model Architecture and Training

For the decoder, we adapt the Huggingface implementation of the Decision Transfomer by [23] based on GPT2 [34], while the state encoder conforms to the state encoder used for BabyAI in [26]. Details are provided in Table 5. As in the original implementation, actions and rewards are encoded linearly, and we use absolute positional embeddings. We extend the original model with sentence-level embeddings for the mission and language feedback, for which we downsample 768-dimensional embeddings from a frozen SentenceBERT model [35]. All embeddings are 128-dimensional, and language embeddings are provided one sentence per timestep.

We keep most of the default values for training parameters listed in Table 6 as used by [23] for their model corresponding to the architecture described above. We train for up to 10 epochs with early stopping based on goal-condition success on the training seeds, and using batches of 64 samples consisting of sub-episodes with context length 64. In the case of single-room levels with simple actions (`goto` and `pickup`), this corresponds to the full episode for unsuccessful episodes. Note that we use random starting points within episodes from which to sample sub-episodes and that consequently, even sub-episodes sampled from long (unsuccessful or maze-level) episodes may contain fewer than 64 steps. The value used for early stopping patience is based on level complexity, whereby we double the patience value after first two levels and again after the first four levels. The target RTG for inference is 1, which is the maximum achievable in BabyAI. While for variants using the mission, we provide the mission for both training and inference, for those variants using feedback, we ablate whether the model has access to the actual feedback at inference time or whether it is given a constant, randomly sampled placeholder. The project code is available on https://github.com/maxtaylordavies/feedback-DT.

Table 5: Architecture hyperparameters

(a) Decoder

Parameter	Value
N layers	3
N attention heads	1
Hidden dimension	128
Dropout	0.1
Non-linearity	ReLU

(b) State encoder

Parameter	Value
N convolutions	3
Channels	16, 32, 64
Filter sizes	2x2, 2x2, 2x2
Strides	1, 1, 1
Non-linearity	ReLU

Table 6: Training hyperparameters

(a) Optimisation

Parameter	Value
Optimiser	AdamW
β_1, β_2	0.9, 0.99
Weight decay	$1 \times 10^{-}4$
Learning rate	$5 \times 10^{-}4$
Scheduler	Linear
Warm-up ratio	0.1

(b) Other

Parameter	Value
Max gradient norm	0.25
Max epochs	10
Early stopping	
- patience (val steps)	8/16/32
- threshold	0.01
Batch size	64

A.3 Further results

A.3.1 Performance by level

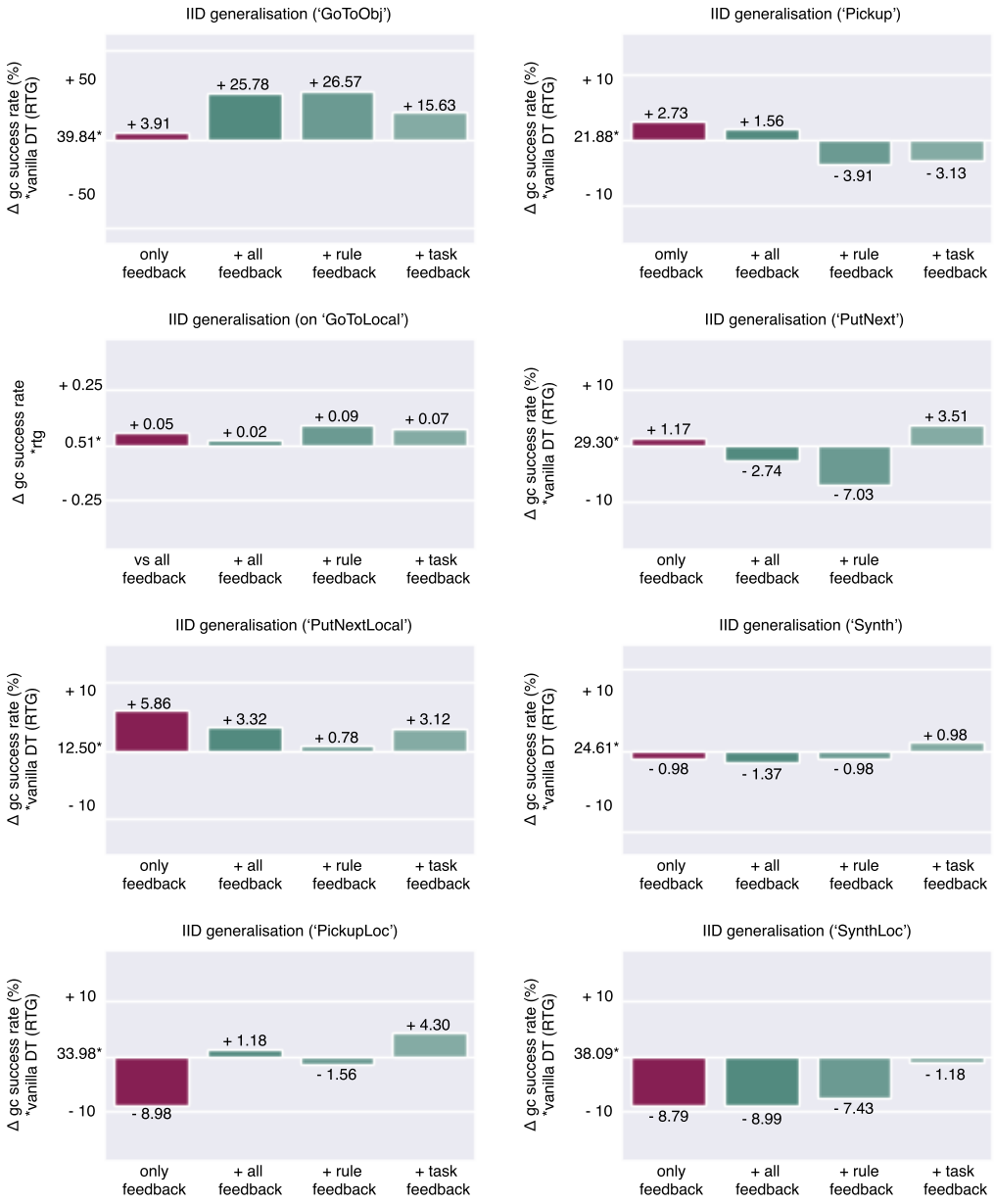

Figure 5: IID generalisation performance by level for the proposed variants conditioning on return and/or feedback compared against the return-only (vanilla DT) baseline.

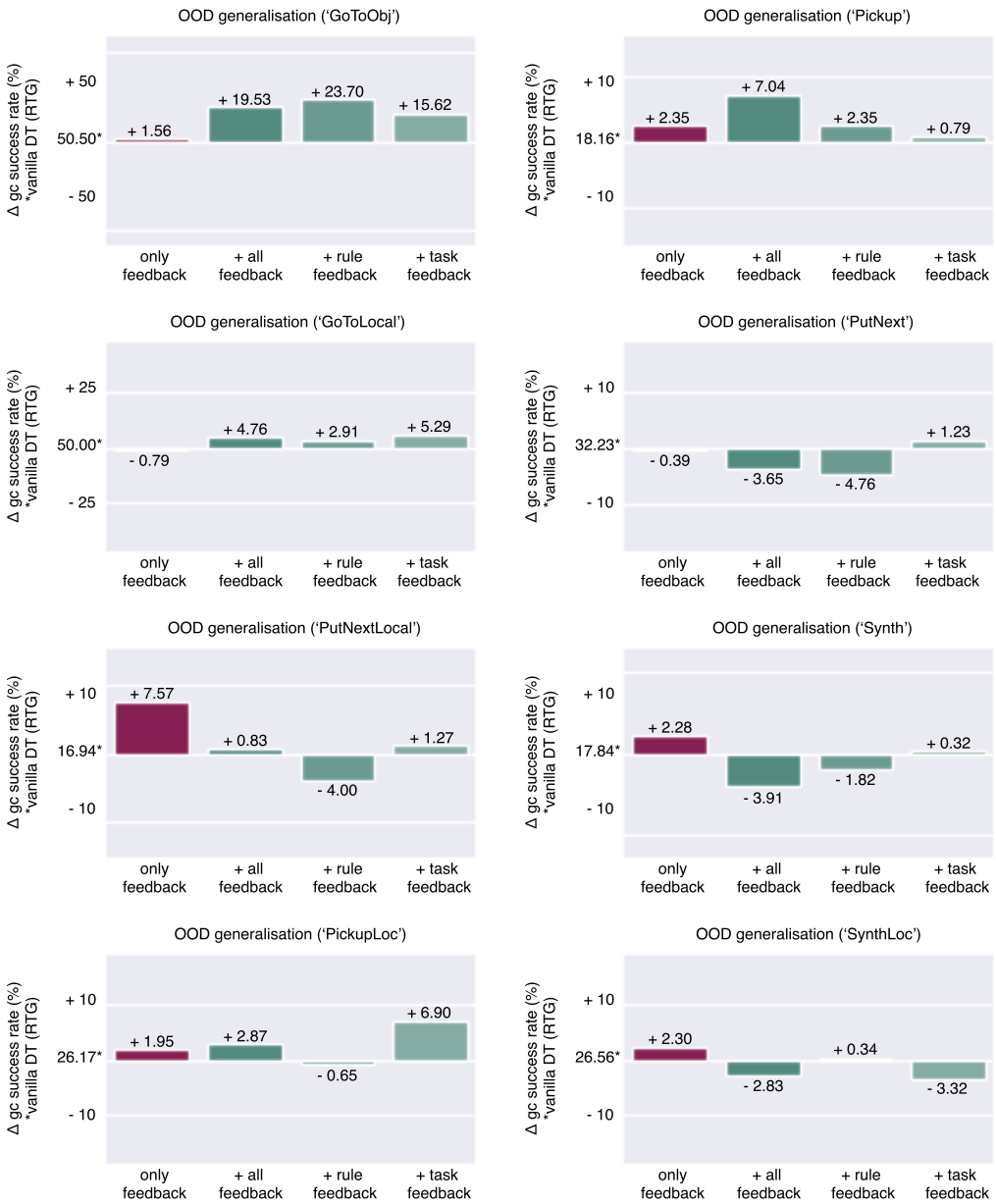

Figure 6: OOD generalisation performance by level for the proposed variants conditioning on return and/or feedback compared against the return-only (vanilla DT) baseline.

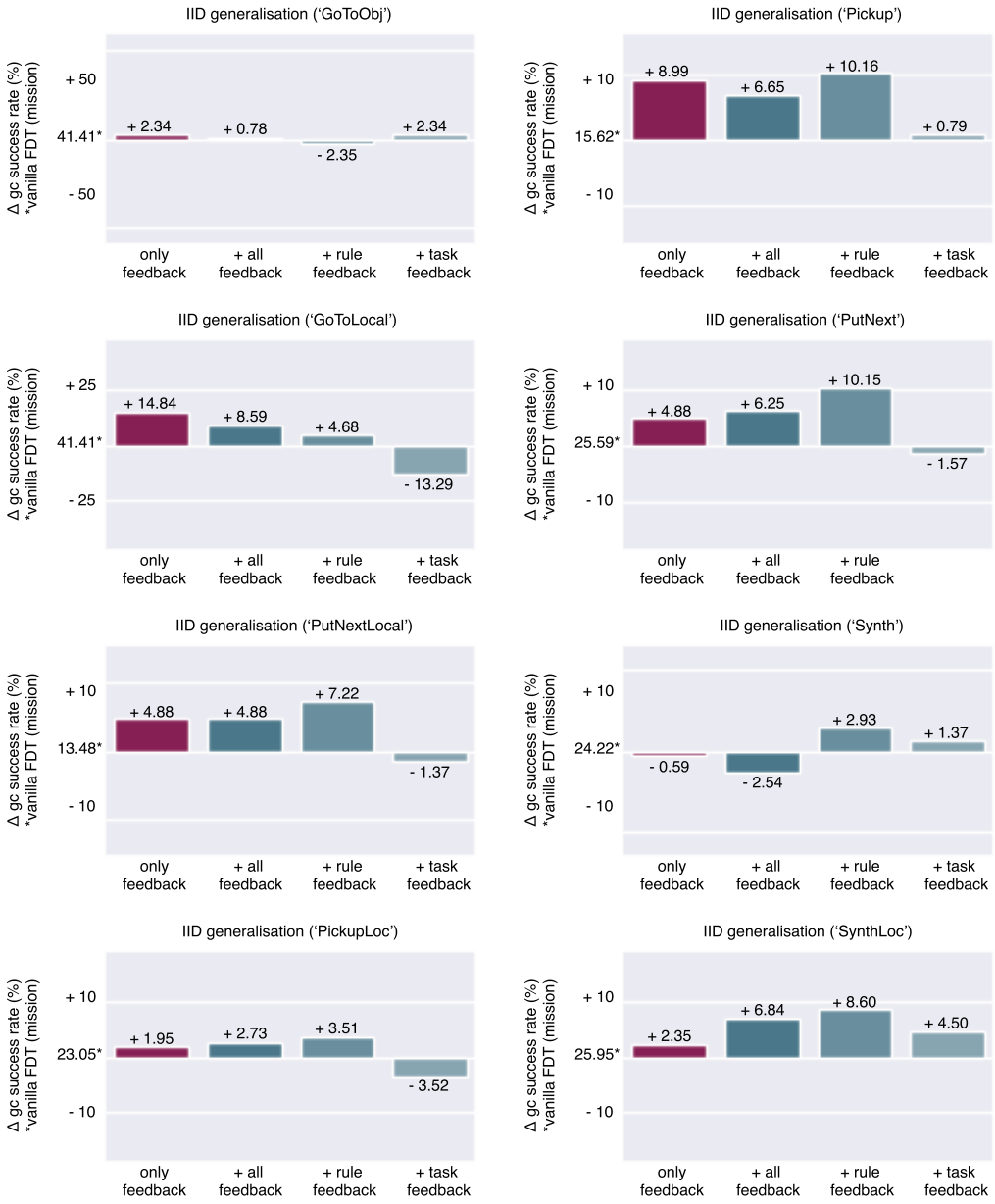

Figure 7: IID generalisation performance by level for the proposed variants conditioning on mission and/or feedback compared against the mission-only (vanilla TDT) baseline.

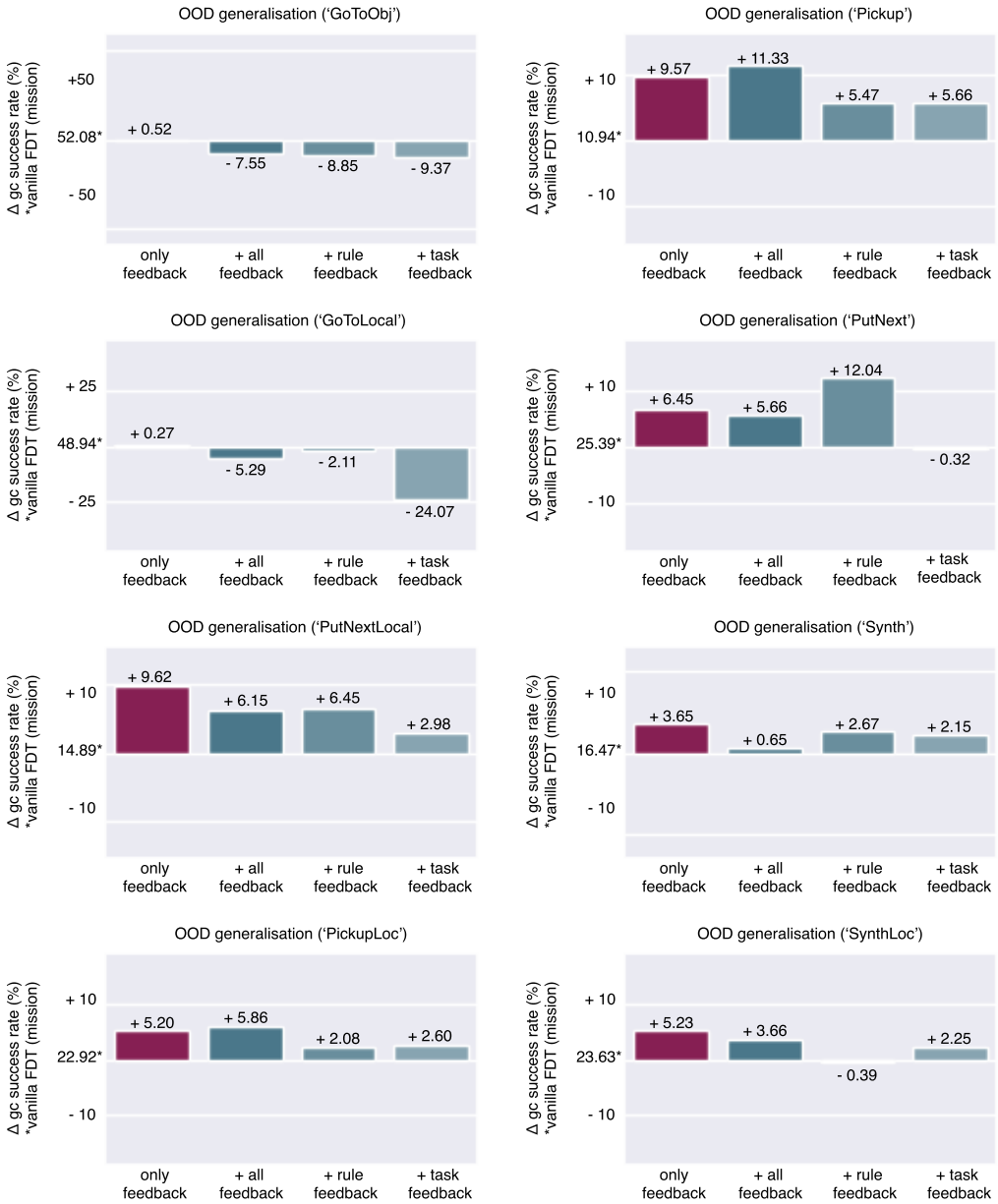

Figure 8: OOD generalisation performance by level for the proposed variants conditioning on mission and/or feedback compared against the mission-only (vanilla TDT) baseline.

A.3.2 OOD performance by OOD type

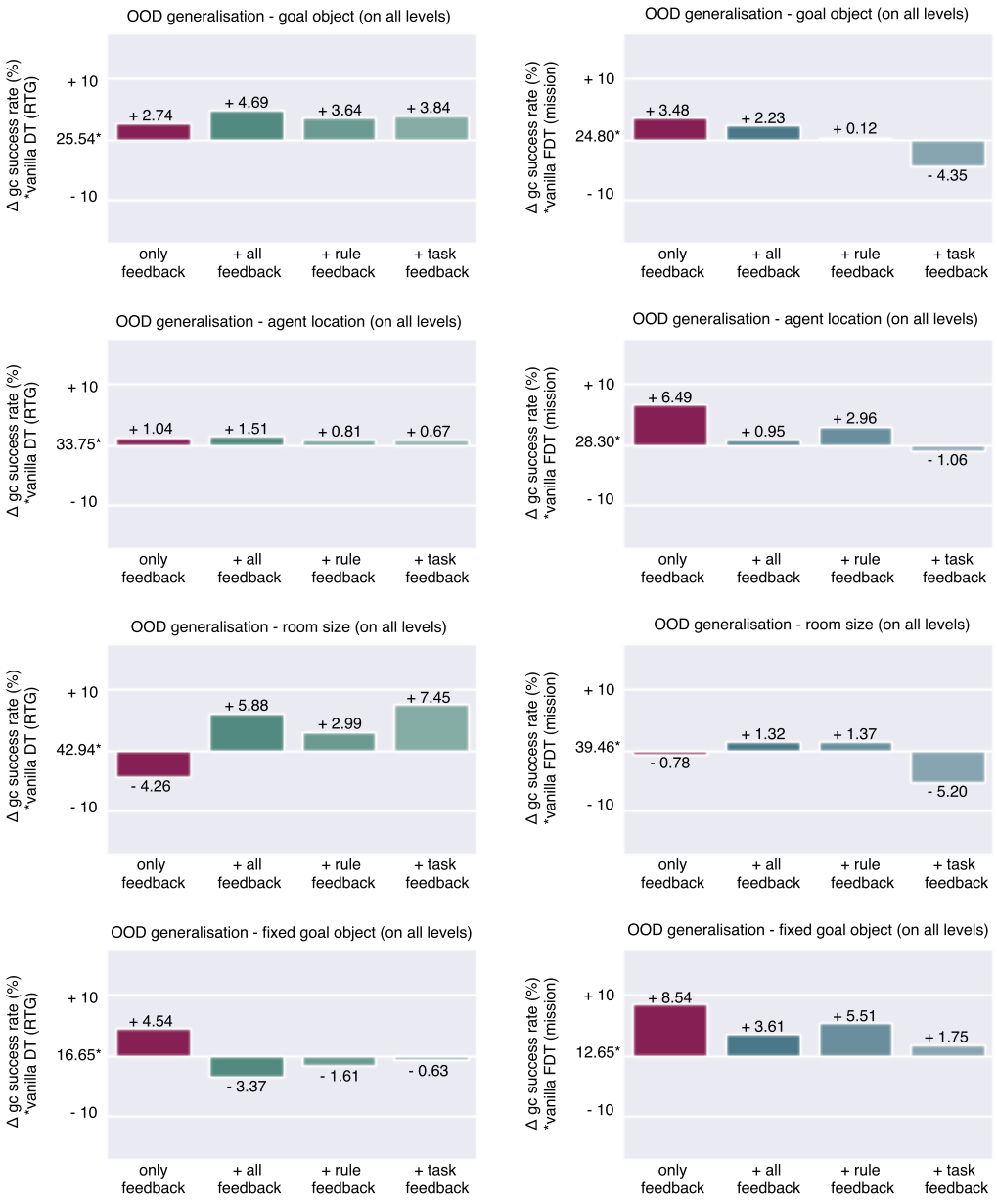

Figure 9: OOD generalisation performance across all eight levels for the proposed variants conditioning on return and/or feedback and mission and/or feedback compared against the return-only (vanilla DT) and mission-only (vanilla TDT) baselines, respectively, w.r.t. agent starting location, goal object color and type, room size and fixed goal object color and type.

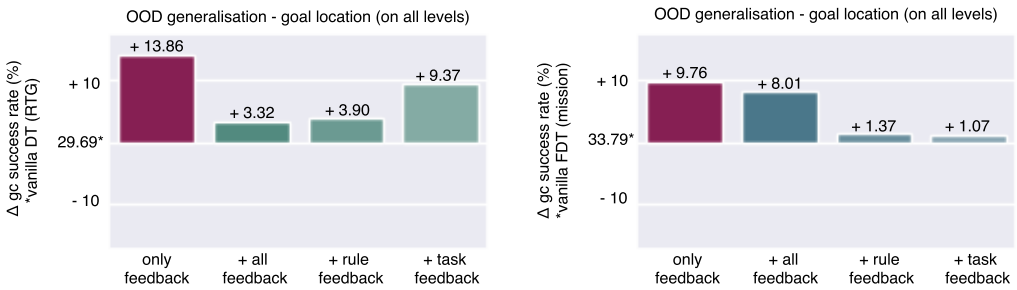

Figure 10: OOD generalisation performance for the proposed variants conditioning on return and/or feedback and mission and/or feedback compared against the return-only (vanilla DT) and mission-only (vanilla TDT) baselines, respectively, w.r.t. relative goal location across all levels.

A.3.3 Impact of feedback at inference

Table 7: Performance of the feedback-only variant against the mission-only and RTG-only baselines, and change in performance when using feedback at inference. *OOD performance averaged across OOD types.

Baseline	Level	IID performance (%)		OOD performance* (%)	
		Delta (vs baseline)	Change (at inference)	Delta (vs baseline)	Change (at inference)
mission	All levels	+4.96	-2.34	+4.96	-4.99
	GoToObj	+2.34	-18.75	-0.52	-28.38
	GoToLocal	+14.84	-0.78	+0.27	+6.35
	PutNextLocal	+4.88	-0.39	+9.62	-6.00
	PickupLoc	+1.95	+10.94	+5.20	-2.47
	Pickup	+8.99	-5.47	+9.57	-0.39
	PutNext	+4.88	-2.35	+6.45	-4.11
	Synth	-0.59	-2.54	+3.65	-0.98
	SynthLoc	+2.35	+0.58	+5.23	-2.69
RTG	All levels	+0.05	-2.34	+2.33	-4.99
	GoToObj	+3.91	-18.75	+1.56	-28.38
	GoToLocal	+5.47	-0.78	-0.79	+6.35
	PutNextLocal	+5.86	-0.39	+7.57	-6.00
	PickupLoc	-8.98	+10.94	+1.95	-2.47
	Pickup	+2.73	-5.47	+2.35	-0.39
	PutNext	+1.17	-2.35	-0.39	-4.11
	Synth	-0.98	-2.54	+2.28	-0.98
	SynthLoc	-8.79	+0.58	+2.30	-2.69

Table 8: Performance of the variants with all feedback in addition to mission/RTG against the respective baselines, and change in performance when using feedback at inference. *OOD performance averaged across OOD types.

Baseline	Level	IID performance (%)		OOD performance* (%)	
		Delta (vs baseline)	Change (at inference)	Delta (vs baseline)	Change (at inference)
mission	All levels	+4.28	-7.20	+2.41	-6.63
	GoToObj	+0.78	-10.94	-7.55	-15.88
	GoToLocal	+8.59	-18.75	-5.29	-17.99
	PutNextLocal	+4.88	-4.10	+6.15	-0.63
	PickupLoc	+2.73	-3.51	+5.86	-6.90
	PutNext	+6.25	-3.52	+5.66	-3.58
	Pickup	+6.65	-11.33	+11.33	-7.04
	Synth	-2.54	+0.98	+0.65	-0.32
	SynthLoc	+6.84	-6.45	+3.66	-3.95
RTG	All levels	+2.64	-1.15	+2.59	-4.07
	GoToObj	+25.78	-9.37	+19.53	-14.32
	GoToLocal	+2.34	+5.47	+4.76	-3.97
	PutNextLocal	+3.32	-7.62	+0.83	-6.69
	PickupLoc	+1.18	-0.78	+2.87	-5.73
	Pickup	+1.56	-6.64	+7.04	-7.23
	PutNext	-2.74	+0.20	-3.65	-1.04
	Synth	-1.37	+5.47	-3.91	+4.10
	SynthLoc	-8.99	+4.10	-2.83	+0.54

Table 9: Performance of the variants with Rule Feedback in addition to mission/RTG compared against the respective baselines, and change in performance when using feedback at inference. *OOD performance averaged across OOD types.

Baseline	Level	IID performance (%)		OOD performance* (%)	
		Delta (vs baseline)	Change (at inference)	Delta (vs baseline)	Change (at inference)
mission	All levels	+5.62	-9.99	+2.12	-6.70
	GoToObj	-2.35	-22.65	-8.85	-17.19
	GoToLocal	+4.68	-8.59	-2.11	-17.99
	PutNextLocal	+7.22	-6.25	+6.45	-3.32
	PickupLoc	+3.51	-4.68	+2.08	+1.04
	Pickup	+10.16	-10.16	+5.47	-0.39
	PutNext	+10.15	-17.38	+12.04	-12.89
	Synth	+2.93	-4.88	+2.67	-4.56
	SynthLoc	+8.60	-5.28	-0.39	+0.20
RTG	All levels	+1.88	-0.22	+1.93	-2.34
	GoToObj	+26.57	-11.72	+23.70	-20.31
	GoToLocal	+8.60	0.00	+2.91	-1.06
	PutNextLocal	+0.78	-4.88	-4.00	-2.05
	PickupLoc	-1.56	-3.12	-0.65	-1.82
	Pickup	-3.91	-3.13	+2.35	+1.37
	PutNext	-7.03	+8.59	-4.76	+4.76
	Synth	-0.98	+5.08	-1.82	+4.94
	SynthLoc	-7.43	+7.43	+0.34	-3.12

Table 10: Performance of the variants with Task Feedback in addition to mission/RTG compared against the respective baselines, and change in performance when using feedback at inference. *OOD performance averaged across OOD types.

Baseline	Level	IID performance (%)		OOD performance* (%)	
		Delta (vs baseline)	Change (at inference)	Delta (vs baseline)	Change (at inference)
mission	All levels	-1.34	+4.23	-2.15	+3.79
	GoToObj	+2.34	+3.13	-9.37	+10.41
	GoToLocal	-13.29	+18.76	-24.07	+21.96
	PutNextLocal	-1.37	+0.59	+2.98	-1.17
	PickupLoc	-3.52	+7.42	+2.60	+1.30
	PutNext	-1.57	+3.71	-0.32	+1.69
	Pickup	+0.79	+5.07	+5.66	+4.30
	Synth	+1.37	-1.76	+2.15	-2.93
	SynthLoc	+4.50	-3.13	+2.25	-1.42
RTG	All levels	+3.79	-3.59	+3.26	-4.21
	GoToObj	+15.63	-14.06	+15.62	-20.57
	GoToLocal	+7.03	-1.56	+5.29	-3.17
	PutNextLocal	+3.12	-5.85	+1.27	-6.74
	PickupLoc	+4.30	+0.39	+6.90	+1.44
	Pickup	-3.13	-1.17	+0.79	+2.73
	PutNext	+3.51	-3.90	+1.23	-6.05
	Synth	+0.98	-0.59	+0.32	-1.82
	SynthLoc	-1.18	-1.95	-3.32	+1.71