Meta-learning from relevant demonstrations can improve compositional generalization

Anonymous Author(s)
Affiliation
Address
email

Abstract

We study the problem of compositional generalization of language-instructed agents in gSCAN. gSCAN is a popular benchmark which requires an agent to generalize to instructions containing novel combinations of words, which are not seen in the training data. We propose to improve the agent's generalization capabilities with an architecture inspired by the Meta-Sequence-to-Sequence learning approach [28]. The agent receives as a context a few examples of pairs of instructions and action trajectories in a given instance of the environment (a support set) and it is tasked to predict an action sequence for a query instruction for the same environment instance. The context is generated by an oracle and the instructions come from the same distribution as seen in the training data. In each training episode, we also shuffle the indices of the attributes of the observed environment states and the words of the instructions to make the agent figure out the relations between the attributes and the words from the context. Our predictive model has the standard transformer architecture. We show that the proposed architecture can significantly improve the generalization capabilities of the agent on one of the most difficult gSCAN splits: the "adverb-to-verb" Split H.

1 Introduction

We want autonomous agents to have the same compositional understanding of language that humans do [10; 46]. Without this understanding, the sample complexity required to train them for a wide range of compositions of instructions would be very high [44; 25]. Naturally, such compositional generalization has received interest from both the language and reinforcement learning communities. "Compositional generalization" can be divided into several different sub-skills, for example being able to reason about object properties compositionally, composing sub-instructions into a sequence or generating action sequences according to novel instructions made up of familiar components [29]. In this work, we examine this latter challenge in more detail by focusing on Split H of gSCAN [39], otherwise known as the "adverb-to-verb" split.

gSCAN[1] is a testing environment for language-grounded agents, consisting of a 6-by-6 grid-world where each episode has a unique combination of objects, an initial agent position and some language instruction. The instructions follow a template of "`action` a `size? color? object adverb`", where `?` indicates that a token is optional. Certain combinations of instructions and object combinations are not found in the training set. A *success* happens when the agent exactly matches the target actions for an episode. A more detailed description of gSCAN is found in Appendix A.

Test Split H contains only instructions following the template "`pull` a `size? color? object while spinning`". This requires the agent to walk towards the target object and pull it the required number

[1] `https://github.com/LauraRuis/GroundedScan`, MIT License

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

of times, while at the same time performing actions `LTURN(4)` after each `WALK` and `PULL`. These instructions from Test Split H, nor their corresponding action sequences are found in the training data. The nature of the distributuon shift is shown in Appendix B. Solving the problem requires the agent to generate the unseen action trajectory based on a compositional understanding of the instructions.

We hypothesize that a promising approach is Meta Sequence-to-Sequence Learning (meta-seq2seq) [28]. We think the reason why this approach works well is the permutations applied to the supports and target instructions, which ensures that an agent does not overfit to particular sequences of symbols in the output space and instead forces meta-learning to determine what the true output actions should be for a given episode. Extending this approach to language grounding environments was flagged as a possible future work direction in [39]. In this work we propose to do exactly that.

Our contributions are: **first**, we describe an extension of meta-seq2seq with state-relevant supports, **second** we report promising success rate performance on gSCAN Split H, **third** we explain different ways to generate the supports and how this affects performance, and **fourth** we motivate how this approach aligns with intuitions about human compositional problem solving.

2 Related Work

Compositional Generalization There is a long line of work on the challenge of compositional generalization in deep learning. Initial works show that sequence models such as RNNs cannot solve these problems well [46; 30]. Datasets such as SCAN [29], COGS [26] and 0gendata [15] serve as benchmarks to measure progress. Since then, various approaches have been proposed to improve compositional generalization performance, including data augmentation [1; 42; 19; 37], problem-specific inductive biases [5; 40; 20; 50; 45], increased data diversity [34; 1], transfer learning [52] and neural module networks [2; 38]. These approaches can perform very well, but usually require prior assumptions about underlying data. In computer vision and multimodal domains, the Transformer architecture has been shown to solve some compositional generalization tasks [47; 24; 8]. Transformer's success on token-level tasks is also promising but still limited [35; 3; 36; 37]. Meta-learning [11; 49; 32] and group equivariance [16] have also shown promise on such problems.

Grounded Environments Many language grounding environments exist, such as BabyAI [9], ALRED [43], VizDoom [7] and SILG [51]. gSCAN and its derivatives [39; 48] specifically focus on task compositional generalization in an interactive world. The various splits of gSCAN are still not completely solved. Various approaches including linguistic-assisted attention [27], graph networks [14], neural module networks [22; 38], data augmentation [41] and Transformers [36] have been proposed. Splits D and H remain challenging to solve with a general approach.

In-context and model-based learning We take inspiration from a long line of work on in-context meta-learning, starting with RL^2 [13] for RNNs and more recently the extension to transformers with TrML [31]. Also related to this work is the idea of retrieval for in-context learning [18; 4] and proposing goals and planning in an imagination of the world [33; 6; 21; 12].

3 Method

Our proposed architecture (Figure 1) extends [29] to embodied environments. The critical aspects of the architecture remain the same. For a given episode with the initial state S, there is a query instruction I^Q and n support instructions $I_1, ..., I_n$ with their corresponding action sequences $A_1, ..., A_n$ (targets). The support instructions and targets are encoded vectors using transformer encoders (the output is taken from a specially reserved position). Instruction-task attention between each token in I^Q and each encoded support instruction $I_1, ..., I_n$ provides mixing weights for each encoded support targets in $A_1, ..., A_n$. Critically, we make random permutations of word/symbol assignments for **both** the instruction and the support commands. The permutations make the task unsolvable without the supports, forcing the model to make use of them and also not to overfit to particular symbol sequences. The query targets A^Q are decoded autoregressively using a transformer decoder with causal masking. Appendix C contains on the finer details of training and the model.

The key extension to meta-seq2seq is how the support instructions and targets are generated. As noted in [39; 32], these cannot be sampled randomly from the data as they are likely not relevant to the

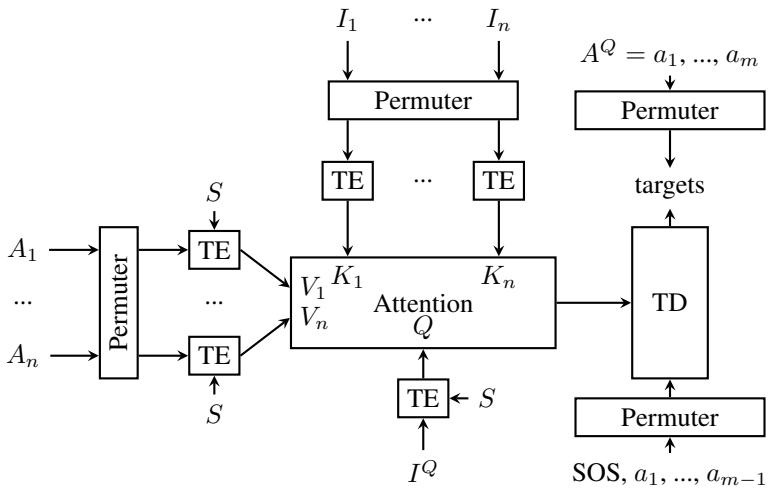


Figure 1: A schematic showing our proposed method. An oracle function, generates relevant in-distribution support instructions sequences $I_1, ..., I_n$, which contain parts of the query instruction, but not the query itself. The action generator, which could be an oracle function or an existing in-distribution model, generates the corresponding support actions sequences $A_1, ..., A_n$ for those instructions in the environment. See Appendix G for an example of how the oracle generates support instructions and actions given a query. The support instructions and actions are embedded using a Transformer encoder (TE) along with the state S. Attention is between each word in I^Q and each encoded support instruction $I_1, ..., I_n$. The output of this attention is the input sequence for a Transformer Decoder (TD), which predicts the output sequence in terms of the permuted action indices for I^Q using causal masking. The agent must use the supports to solve the query instruction as the word and action indices are re-permuted on every episode.

current state. In this work, we use an oracle templating function to generate instructions and actions pertaining to the same object in the same environment, but with different in-distribution verb and adverb combinations. The oracle does not generate an example of the query instruction, instructions only seen in Split H. See Appendix G for a detailed description and examples.

4 Experiments

We ran experiments to compare success rate performance of our approach against a transformer [36] and other baselines, shown in Table 1. Both the transformer and the adapted meta-seq2seq approach above have a similar parameter budget, effective batch size, learning rate, and training iteration count. Runs are over seeds 0-9 and the bottom quantile of seeds by Split A performance are excluded. In Ours(o, A) we measure validation performance on all the other splits by Split A performance, since the other splits are in-principle unseen. Ours(o) is our model at the best Split H performance. Permutation of word and action indices happens at training time but not at test time.

Table 1: Our approach compared to other recent works on the gSCAN dataset. Numbers are mean success rate over 10 seeds ± standard deviation. Additional comparisons can be found Appendix F.

	ViT	seq2seq	GECA	Modular	Role-guided	Transformer	Ours(o, A)	Ours(o)
Reference	Qiu et al. [36]	Ruis et al. [39]	Ruis et al. [39]	Ruis and Lake [38]	Kuo et al. [27]	Ours	Ours	Ours
Learning Rate	0.0015	0.001	0.001	0.001	0.001	0.0001	0.0001	0.0001
Batch Size	128	200	200	200	200	4096	4096	4096
Steps	114.96K	200K	200K	73K	150K	35K	35K	35K
#params	3M					13.2M	13.2M	13.2M
A	99.95 ± 0.02	97.15 ± 0.46	87.6 ± 1.19	96.34 ± 0.28	97.32	**1.0 ± 0.0**	0.96 ± 0.0	0.95 ± 0.01
B	**99.90 ± 0.06**	30.05 ± 26.76	34.92 ± 39.30	59.66 ± 23.76	95.35	0.91 ± 0.15	0.97 ± 0.01	0.96 ± 0.01
C	**99.25 ± 0.91**	29.79 ± 17.70	78.77 ± 6.63	32.09 ± 9.79	80.16	0.86 ± 0.19	0.97 ± 0.01	0.97 ± 0.01
D	0.0 ± 0.0	0 ± 0	0.00 ± 0.00	0.0 ± 0.0	5.73	0.0 ± 0.0	0.34 ± 0.05	**0.35 ± 0.05**
E	**99.02 ± 1.16**	37.25 ± 2.85	33.19 ± 3.69	49.34 ± 11.60	75.19	0.79 ± 0.26	0.98 ± 0.01	0.97 ± 0.01
F	99.98 ± 0.01	94.16 ± 1.25	85.99 ± 0.85	94.16 ± 1.25	98.63	**1.0 ± 0.0**	0.97 ± 0.01	0.97 ± 0.01
H	22.16 ± 0.01	19.04 ± 4.08	11.83 ± 0.31	76.84 ± 26.94	21.95	0.08 ± 0.09	0.8 ± 0.05	**0.86 ± 0.02**

Table 2: Different types of oracle behaviour. Numbers are successes ± standard deviation over top 5 of 10 seeds by Split A performance, due to some seeds taking much longer to converge than others. Ablations measured at 28,000 iterations at the best validation checkpoint on Split A.

	Ours(o, A)	No permutations	Transformer Actions	Distractors	Random Instructions	Sample Environments
A	0.96 ± 0.0	0.97 ± 0.0	0.94 ± 0.01	0.78 ± 0.12	0.27 ± 0.01	0.23 ± 0.02
B	0.96 ± 0.01	0.98 ± 0.0	0.95 ± 0.01	0.82 ± 0.1	0.27 ± 0.02	0.16 ± 0.03
C	0.97 ± 0.01	0.98 ± 0.0	0.96 ± 0.01	0.82 ± 0.09	0.27 ± 0.0	0.01 ± 0.0
D	0.35 ± 0.06	0.03 ± 0.03	0.0 ± 0.0	0.17 ± 0.1	0.02 ± 0.02	0.0 ± 0.0
E	0.98 ± 0.01	0.98 ± 0.0	0.97 ± 0.0	0.83 ± 0.12	0.28 ± 0.02	0.04 ± 0.02
F	0.97 ± 0.01	0.99 ± 0.0	0.96 ± 0.01	0.79 ± 0.13	0.23 ± 0.02	0.34 ± 0.04
H	0.81 ± 0.03	0.17 ± 0.07	0.76 ± 0.03	0.49 ± 0.17	0.0 ± 0.0	0.08 ± 0.01

We achieve very strong performance on Split H, comparable to the recent work of Ruis and Lake [38]. An analysis of the remaining failure cases is provided in Appendix H. Note that for splits B-F, the oracle demonstrates related instructions to the corresponding to an out-of-distribution object or location, so those results are not exactly comparable to prior work, but they do demonstrate that a few-shot compositional meta-learning paradigm can extend to other splits. We also see that a Transformer at the same scale starts to have degraded performance on out-of-distribution splits, indicating that it continues to overfit to the in-distribution set as the model scales up.

We also studied different kinds of oracle function behaviour, shown in Table 2. Without permutations, performance on Splits D and H are comparable to a Transformer, confirming the result in [28]. Ideally we also want to be robust to irrelevant distractor supports if we are to replace the oracle in the future with a generative model that generates a superset of the relevant instructions. When 3 out of 8 instructions are distractors, performance is worse, but this is mainly due to higher variance in convergence rate between seeds. As we expect, generating random and likely irrelevant instructions or sampling different environment layouts for relevant instructions causes performance on all splits to drop significantly. Since our ultimate aim is to replace the oracle with a generative model in future work, we also tested a scenario the action supports come from a baseline Transformer which receives the current state and the support instructions generated by the oracle function. In this case the Transformer was trained on the gSCAN training set and had a 95% autoregressive generation success rate on Split A generates actions for instructions generated by the oracle. In this scenario, we can still get decent performance on all splits except Split D.

5 Discussion and Conclusion

The extension of meta-seq2seq we present in this paper has promising results in the challenging Split H of the gSCAN benchmark. We believe its architecture is also simple and intuitive to understand through the lens of how humans might think about compositional problems. When faced with an unfamiliar instruction (I^Q), the agent thinks of similar instructions ($I_1, ..., I_n$) and their solutions ($A_1, ..., A_n$) in the same environment. The agent then thinks about how those solutions can be composed in light of the current instruction.

At present, an important limitation of this work is that an oracle function generates $I_1, ..., I_n$ and $A_1, ..., A_n$ for a given I^Q. We hope to extend this work by replacing the oracle with a generative model, which generates in-distribution query instructions and their corresponding actions with reference to the generative distribution of the training data. There is also a slight degradation of performance on the other splits, including the in-distribution Split A, compared to a Transformer. In light of the limitations, we suggest that applications use this work in conjunction with a baseline, for example by distillation [23] or by using this approach as a sort of "system-2" fallback for when a "system-1" model is uncertain about its inputs [17].

Generalization to unseen instruction compositions remains a challenging problem. Our hypothesis was that meta-seq2seq is a promising general approach and could be extended to grounded language scenarios by an agent which generates a relevant meta-learning context. Our preliminary results show that such an extension has promise and is an area for future work.

References

[1] Jacob Andreas. Good-enough compositional data augmentation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7556–7566. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.676. URL https://doi.org/10.18653/v1/2020.acl-main.676.

[2] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48, 2016. doi: 10.1109/CVPR.2016.12.

[3] Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 7096–7116. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.576. URL https://doi.org/10.18653/v1/2020.emnlp-main.576.

[4] Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens. *CoRR*, abs/2112.04426, 2021. URL https://arxiv.org/abs/2112.04426.

[5] Ayush K. Chakravarthy, Jacob Labe Russin, and Randall C. O'Reilly. Systematicity emerges in transformers when abstract grammatical roles guide attention. In Daphne Ippolito, Liunian Harold Li, Maria Leonor Pacheco, Danqi Chen, and Nianwen Xue, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop, NAACL-HLT 2022, Hybrid Event / Seattle, WA, USA, July 10-15, 2022*, pages 1–8. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.naacl-srw.1.

[6] Elliot Chane-Sane, Cordelia Schmid, and Ivan Laptev. Goal-conditioned reinforcement learning with imagined subgoals. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 1430–1440. PMLR, 2021. URL http://proceedings.mlr.press/v139/chane-sane21a.html.

[7] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding. In Sheila A. McIlraith and Kilian Q. Weinberger, editors, *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 2819–2826. AAAI Press, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17425.

[8] Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: universal image-text representation learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XXX*, volume 12375 of *Lecture Notes in Computer Science*, pages 104–120. Springer, 2020. doi: 10.1007/978-3-030-58577-8_7. URL https://doi.org/10.1007/978-3-030-58577-8_7.

[9] Maxime Chevalier-Boisvert, Dzmitry Bahdanau, Salem Lahlou, Lucas Willems, Chitwan Saharia, Thien Huu Nguyen, and Yoshua Bengio. BabyAI: A platform to study the sample efficiency of grounded language learning. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=rJeXCo0cYX.

[10] Noam Chomsky. *Syntactic Structures*. Mouton and Co., The Hague, 1957.

[11] Henry Conklin, Bailin Wang, Kenny Smith, and Ivan Titov. Meta-learning to compositionally generalize. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3322–3335. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.acl-long.258. URL https://doi.org/10.18653/v1/2021.acl-long.258.

[12] Andreea Deac, Petar Velickovic, Ognjen Milinkovic, Pierre-Luc Bacon, Jian Tang, and Mladen Nikolic. Neural algorithmic reasoners are implicit planners. *CoRR*, abs/2110.05442, 2021. URL https://arxiv.org/abs/2110.05442.

[13] Yan Duan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *CoRR*, abs/1611.02779, 2016. URL http://arxiv.org/abs/1611.02779.

[14] Tong Gao, Qi Huang, and Raymond J. Mooney. Systematic generalization on gSCAN with language conditioned embedding. In Kam-Fai Wong, Kevin Knight, and Hua Wu, editors, *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing, AACL/IJCNLP 2020, Suzhou, China, December 4-7, 2020*, pages 491–503. Association for Computational Linguistics, 2020. URL https://aclanthology.org/2020.aacl-main.49/.

[15] Atticus Geiger, Ignacio Cases, Lauri Karttunen, and Christopher Potts. Posing fair generalization tasks for natural language inference. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 4484–4494. Association for Computational Linguistics, 2019. doi: 10.18653/v1/D19-1456. URL https://doi.org/10.18653/v1/D19-1456.

[16] Jonathan Gordon, David Lopez-Paz, Marco Baroni, and Diane Bouchacourt. Permutation equivariant models for compositional generalization in language. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=SylVNerFvr.

[17] Anirudh Goyal and Yoshua Bengio. Inductive biases for deep learning of higher-level cognition. *CoRR*, abs/2011.15091, 2020. URL https://arxiv.org/abs/2011.15091.

[18] Anirudh Goyal, Abram L. Friesen, Andrea Banino, Theophane Weber, Nan Rosemary Ke, Adrià Puigdomènech Badia, Arthur Guez, Mehdi Mirza, Ksenia Konyushkova, Michal Valko, Simon Osindero, Timothy P. Lillicrap, Nicolas Heess, and Charles Blundell. Retrieval-augmented reinforcement learning. *CoRR*, abs/2202.08417, 2022. URL https://arxiv.org/abs/2202.08417.

[19] Demi Guo, Yoon Kim, and Alexander M. Rush. Sequence-level mixed sample data augmentation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5547–5552. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.447. URL https://doi.org/10.18653/v1/2020.emnlp-main.447.

[20] Yinuo Guo, Zeqi Lin, Jian-Guang Lou, and Dongmei Zhang. Hierarchical poset decoding for compositional generalization in language. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/4d7e0d72898ae7ea3593eb5ebf20c744-Abstract.html.

[21] Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

[22] Christina Heinze-Deml and Diane Bouchacourt. Think before you act: A simple baseline for compositional generalization. *arXiv:2009.13962*, 2009.13962, 2020. URL https://arxiv.org/abs/2009.13962.

[23] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531, 2015. URL http://arxiv.org/abs/1503.02531.

[24] Drew A. Hudson and Larry Zitnick. Generative adversarial transformers. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pages 4487–4499. PMLR, 2021. URL http://proceedings.mlr.press/v139/hudson21a.html.

[25] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-Z: zero-shot task generalization with robotic imitation learning. In Aleksandra Faust, David Hsu, and Gerhard Neumann, editors, *5th Annual Conference on Robot Learning, 8-11 November 2021, London, UK*, pages 991–1002, 2021. URL https://proceedings.mlr.press/v164/jang22a.html.

[26] Najoung Kim and Tal Linzen. COGS: A compositional generalization challenge based on semantic interpretation. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9087–9105. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.731. URL https://doi.org/10.18653/v1/2020.emnlp-main.731.

[27] Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Compositional networks enable systematic generalization for grounded language understanding. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 216–226. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-emnlp.21. URL https://doi.org/10.18653/v1/2021.findings-emnlp.21.

[28] Brenden M. Lake. Compositional generalization through meta sequence-to-sequence learning. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 9788–9798, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/f4d0e2e7fc057a58f7ca4a391f01940a-Abstract.html.

[29] Brenden M. Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 2879–2888. PMLR, 2018. URL http://proceedings.mlr.press/v80/lake18a.html.

[30] João Loula, Marco Baroni, and Brenden M. Lake. Rearranging the familiar: Testing compositional generalization in recurrent networks. In Tal Linzen, Grzegorz Chrupala, and Afra Alishahi, editors, *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 108–114. Association for Computational Linguistics, 2018. doi: 10.18653/v1/w18-5413. URL https://doi.org/10.18653/v1/w18-5413.

[31] Luckeciano C. Melo. Transformers are meta-reinforcement learners. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International*

Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA, volume 162 of *Proceedings of Machine Learning Research*, pages 15340–15359. PMLR, 2022. URL https://proceedings.mlr.press/v162/melo22a.html.

[32] Eric Mitchell, Chelsea Finn, and Christopher D. Manning. Challenges of acquiring compositional inductive biases via meta-learning. In Isabelle Guyon, Jan N. van Rijn, Sébastien Treguer, and Joaquin Vanschoren, editors, *AAAI Workshop on Meta-Learning and MetaDL Challenge, MetaDL@AAAI 2021, virtual, February 9, 2021*, volume 140 of *Proceedings of Machine Learning Research*, pages 138–148. PMLR, 2021. URL https://proceedings.mlr.press/v140/mitchell21a.html.

[33] Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In Samy Bengio, Hanna M. Wallach, Hugo Larochelle, Kristen Grauman, Nicolò Cesa-Bianchi, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada*, pages 9209–9220, 2018. URL https://proceedings.neurips.cc/paper/2018/hash/7ec69dd44416c46745f6edd947b470cd-Abstract.html.

[34] Arkil Patel, Satwik Bhattamishra, Phil Blunsom, and Navin Goyal. Revisiting the compositional generalization abilities of neural sequence models. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 424–434. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.acl-short.46.

[35] Alethea Power, Yuri Burda, Harrison Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *CoRR*, abs/2201.02177, 2022. URL https://arxiv.org/abs/2201.02177.

[36] Linlu Qiu, Hexiang Hu, Bowen Zhang, Peter Shaw, and Fei Sha. Systematic generalization on gSCAN: What is nearly solved and what is next? In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 2180–2188. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.emnlp-main.166. URL https://doi.org/10.18653/v1/2021.emnlp-main.166.

[37] Linlu Qiu, Peter Shaw, Panupong Pasupat, Pawel Krzysztof Nowak, Tal Linzen, Fei Sha, and Kristina Toutanova. Improving compositional generalization with latent structure and data augmentation. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 4341–4362. Association for Computational Linguistics, 2022. URL https://aclanthology.org/2022.naacl-main.323.

[38] Laura Ruis and Brenden M. Lake. Improving systematic generalization through modularity and augmentation. *arXiv:2202.10745*, 2022. URL https://arxiv.org/abs/2202.10745.

[39] Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M. Lake. A benchmark for systematic generalization in grounded language understanding. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, Virtual Event*, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/e5a90182cc81e12ab5e72d66e0b46fe3-Abstract.html.

[40] Jacob L. Russin, Jason Jo, Randall C. O'Reilly, and Yoshua Bengio. Systematicity in a recurrent neural network by factorizing syntax and semantics. In Stephanie Denison, Michael Mack, Yang Xu, and Blair C. Armstrong, editors, *Proceedings of the 42th Annual Meeting of the Cognitive Science Society - Developing a Mind: Learning in Humans, Animals, and Machines, CogSci 2020, virtual, July 29 - August 1, 2020*. cognitivesciencesociety.org, 2020. URL https://cogsci.mindmodeling.org/2020/papers/0027/index.html.

[41] Matthew Setzler, Scott Howland, and Lauren A. Phillips. Recursive decoding: A situated cognition approach to compositional generation in grounded language understanding. *CoRR*, abs/2201.11766, 2022. URL https://arxiv.org/abs/2201.11766.

[42] Haoyue Shi, Karen Livescu, and Kevin Gimpel. Substructure substitution: Structured data augmentation for NLP. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 3494–3508. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-acl.307. URL https://doi.org/10.18653/v1/2021.findings-acl.307.

[43] Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 10737–10746. Computer Vision Foundation / IEEE, 2020. doi: 10.1109/CVPR42600.2020.01075. URL https://openaccess.thecvf.com/content_CVPR_2020/html/Shridhar_ALFRED_A_Benchmark_for_Interpreting_Grounded_Instructions_for_Everyday_Tasks_CVPR_2020_paper.html.

[44] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, pages 9767–9779, 2021. URL http://proceedings.mlr.press/v139/sodhani21a.html.

[45] Sam Spilsbury and Alexander Ilin. Compositional generalization in grounded language learning via induced model sparsity. In Daphne Ippolito, Liunian Harold Li, Maria Leonor Pacheco, Danqi Chen, and Nianwen Xue, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Student Research Workshop, NAACL-HLT 2022, Hybrid Event / Seattle, WA, USA, July 10-15, 2022*, pages 143–155. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.naacl-srw.19. URL https://doi.org/10.18653/v1/2022.naacl-srw.19.

[46] Josh Tenenbaum. Building machines that learn and think like people. In Elisabeth André, Sven Koenig, Mehdi Dastani, and Gita Sukthankar, editors, *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS 2018, Stockholm, Sweden, July 10-15, 2018*, page 5. International Foundation for Autonomous Agents and Multiagent Systems Richland, SC, USA / ACM, 2018. URL http://dl.acm.org/citation.cfm?id=3237389.

[47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

[48] Zhengxuan Wu, Elisa Kreiss, Desmond C. Ong, and Christopher Potts. Reascan: Compositional reasoning in language grounding. *CoRR*, abs/2109.08994, 2021. URL https://arxiv.org/abs/2109.08994.

[49] Jingfeng Yang, Haoming Jiang, Qingyu Yin, Danqing Zhang, Bing Yin, and Diyi Yang. Seqzero: Few-shot compositional semantic parsing with sequential prompts and zero-shot models. *CoRR*, abs/2205.07381, 2022. doi: 10.48550/arXiv.2205.07381. URL https://doi.org/10.48550/arXiv.2205.07381.

[50] Pengcheng Yin, Hao Fang, Graham Neubig, Adam Pauls, Emmanouil Antonios Platanios, Yu Su, Sam Thomson, and Jacob Andreas. Compositional generalization for neural semantic parsing via span-level supervised attention. In Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, editors, *Proceedings of the 2021 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021, pages 2810–2823. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.225. URL https://doi.org/10.18653/v1/2021.naacl-main.225.

[51] Victor Zhong, Austin W. Hanjie, Sida I. Wang, Karthik Narasimhan, and Luke Zettlemoyer. SILG: the multi-domain symbolic interactive language grounding benchmark. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan, editors, *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 21505–21519, 2021. URL https://proceedings.neurips.cc/paper/2021/hash/b3e3e393c77e35a4a3f3cbd1e429b5dc-Abstract.html.

[52] Wang Zhu, Peter Shaw, Tal Linzen, and Fei Sha. Learning to generalize compositionally by transferring across semantic parsing tasks. *CoRR*, abs/2111.05013, 2021. URL https://arxiv.org/abs/2111.05013.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [N/A]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Details on the gSCAN environment

The objects are one of a circle, cylinder or triangle, can be five different sizes and come in the colors red, blue, purple and yellow. The environment is encoded as a grid where each cell is a bag of words, similar to [9].

Available instruction action words are `push`, `pull` and `walk to` and available instruction adverbs are `while spinning`, `while zigzagging`, `hesitantly` and `cautiously`. Actions which the agent must product are in the set of `WALK`, `STAY`, `LTURN`, `RTURN`, `PUSH` and `PULL`.

Split A is an in-distribution validation set split, containing instructions, target objects and target locations which can be found in the training set.

Splits B to F are out-of-distribution sets where the target object has an unseen description made up of combinations of familiar terms (for example red square in split C) or is at a location not seen before during training (for example, southwest of the agent in split D).

Split G is a "meta-learning" split, where an example of "cautiously" is seen only k times during training. We don't consider this split in our work, though initial experiments showed that our approach does not solve this task when the number of related examples is small. There is a good discussion in Ruis and Lake [38] about a data-augmentation method which can help to solve this split.

B Conditional Probability Distribution of gSCAN splits

Figure 2: One-step ahead conditional distributions in various gSCAN splits for targets. Shown is $P(a_t|a_{t-1})$ where a_t is down the rows and a_{t-1} is along the columns). Bolded are conditional probabilities notably different from the training split.

(a) Training split

	PULL	PUSH	STAY	LTURN	RTURN	WALK
PULL	0.52	0.00	0.33	0.00	0.00	0.15
PUSH	0.00	0.27	0.26	0.26	0.00	0.21
STAY	0.19	0.05	0.00	0.00	0.00	0.75
LTURN	0.00	0.01	0.01	0.73	0.00	0.24
RTURN	0.00	0.00	0.18	0.14	0.00	0.68
WALK	0.00	0.00	0.20	0.35	0.17	0.28

(b) Split A

	PULL	PUSH	STAY	LTURN	RTURN	WALK
PULL	0.52	0.00	0.32	0.00	0.00	0.16
PUSH	0.00	0.28	0.25	0.27	0.00	0.21
STAY	0.25	0.07	0.00	0.00	0.00	0.68
LTURN	0.00	0.02	0.02	0.73	0.00	0.23
RTURN	0.00	0.00	0.20	0.13	0.00	0.67
WALK	0.00	0.00	0.19	0.33	0.19	0.29

(c) Split H

	PULL	PUSH	STAY	LTURN	RTURN	WALK
PULL	0.00	0.00	0.00	**1.00**	0.00	0.00
PUSH	0.00	0.00	0.00	0.00	0.00	0.00
STAY	0.00	0.00	0.00	0.00	0.00	0.00
LTURN	**0.08**	0.00	0.00	0.78	0.00	0.14
RTURN	0.00	0.00	0.00	1.00	0.00	0.00
WALK	0.00	0.00	0.00	0.89	0.11	0.00

(d) Split D

	PULL	PUSH	STAY	LTURN	RTURN	WALK
PULL	0.52	0.00	0.33	0.00	0.00	0.15
PUSH	0.00	0.27	0.26	0.27	0.00	0.20
STAY	0.16	0.05	0.00	0.00	0.00	0.79
LTURN	0.00	0.01	0.04	0.69	0.00	0.26
RTURN	0.00	0.00	0.00	0.00	0.00	**1.00**
WALK	0.00	0.00	0.18	0.53	0.07	0.22

As shown in Figure 2, there are significant conditional distribution shifts between the various gSCAN splits. In the training split, both $P(\texttt{LTURN}|\texttt{PULL}) = 0$ and $P(\texttt{PULL}|\texttt{LTURN}) = 0$. On the contrary, $P(\texttt{PULL}|\texttt{LTURN}) = 1$ in Split H and $P(\texttt{LTURN}|\texttt{PULL}) > 0$. In Split D, there is a much higher probability of making a right turn after walking, which corresponds to the agent going southwest (since navigation in gSCAN goes first in the vertical direction, then in the horizontal direction). While these one-step-ahead conditional probabilities don't capture the full picture, since there are other variables such as other previous actions as well as the instruction, they do give a snapshot of the nature of the distribution shift problem in splits D and H.

C Training and Model Details

For the both the meta-seq2seq model and Transformer model we use a hidden size of 128 units and fully-connected layer size of 512 units. During training, dropout is not used and weight decay is set to 10^{-3} with the AdamW optimizer. Learning rate warmup is used up to step 5000 to a peak learning rate of 10^{-5}, then decayed on a log-linear schedule from steps 5000 to 50000 to 10^{-6}. Beta values are left at their defaults, $\beta_1 = 0.9$ and $\beta_2 = 0.999$. Gradients norms are clipped at 0.2 to improve

training stability. We use 16-bit precision during training and make use of gradient accumulation in order to simulate large batch sizes where memory is limited.

The Transformer baseline we use is different and more vanilla than the ViT inspired architecture used in [36]. The Transformer follows an encoder-decoder structure, where the state and instruction are jointly encoded and the actions are decoded with causal masking. Each cell in the state is encoded as a bag-of-words and concatenated together, such that different components correspond to different properties of the cell. Learnable 2D positional encodings are appended to each cell. The cells are flattened and concatenated with the position-encoded inputs to form the encoder sequence.

For the meta-seq2seq model, there are Transformer encoders for both instructions and meta-learning action supports. The Transformer encoders for the instructions are tied for both the query instruction and the key support instructions. The Transformer encoder for the instruction encodes the state (as above) and decodes the instruction without any causal masking. An inducing point is appended to the end of the instruction such that the instruction can be represented as a single vector for the purposes of attention computation. Attention is taken between each word in the query instruction and each encoded inducing point for each of the support instructions, then multiplied with the encoded inducing points corresponding to each of the support action sequences. The output sequence of actions is then decoded using a Transformer decoder with causal masking.

D Computational Resource Usage and Reproducibility Requirements

Experiments were run on our internal GPU cluster. Running a meta-learning experiment to 30,000 iterations takes about 3 days on a NVIDIA Tesla V100 GPU. For 7 different experiment runs with 10 seeds each, the total compute time is about 210 GPU-days, though in practice the experiments can be run in parallel.

The batch size (4096) we use is quite large and does not fit in GPU memory for a consumer grade GPU. In order to achieve these batch sizes, we use gradient accumulation, so the batch size for each backward step might be 256, but then gradients are averaged over 16 steps to make an effective batch size of 4096. This trades training time for optimization stability.

E Code and Datasets

To assist the reader in understanding our work we provide a copy of our PyTorch code for the models, code used to generate figures and tables in this work and experimental code at this URL: `https://neurips-larel-meta-learning-gscan-generalize-submission.s3.eu-north-1.amazonaws.com/submission.zip`

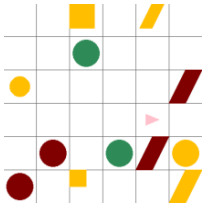
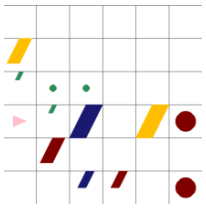
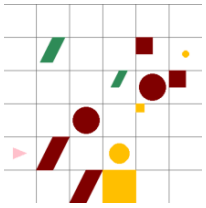
F Additional Comparisons

Table 3: Additional related work comparions.

	FiLM	RelNet	LCGN	Planning	RD Random/RL	Transformer	Ours(o, A)	Ours(o)
Reference	[36]	[36]	[14]	[22]	[41]	Ours	Ours	Ours
Learning Rate	0.0015	0.0015	0.0001	0.001		0.0001	0.0001	0.0001
Batch Size	128	128	64	200	4096	512	4096	4096
Steps	115K	115K	575K	368K	35K		35K	35K
#params				535K		13.2M	13.2M	13.2M
A	98.83 ± 0.32	97.38 ± 0.33	98.6 ± 0.9	94.19 ± 0.71	98.39 ± 0.17	1.0 ± 0.0	0.96 ± 0.0	0.95 ± 0.01
B	94.04 ± 7.41	49.44 ± 8.19	99.08 ± 0.69	87.31 ± 4.38	62.19 ± 24.08	0.91 ± 0.15	0.97 ± 0.01	0.96 ± 0.01
C	60.12 ± 8.81	19.92 ± 9.84	80.31 ± 24.51	81.07 ± 10.12	56.52 ± 29.70	0.86 ± 0.19	0.97 ± 0.01	0.97 ± 0.01
D	0.00 ± 0.00	0.00 ± 0.00	0.16 ± 0.12		0.2762 ± 0.0537	0.0 ± 0.0	0.34 ± 0.05	0.35 ± 0.05
E	31.64 ± 1.04	42.17 ± 6.22	87.32 ± 27.38	52.8 ± 9.96	53.89 ± 5.39	0.97 ± 0.01	0.97 ± 0.007	0.85 ± 0.20
F	86.45 ± 6.67	96.59 ± 0.94	99.33 ± 0.46		95.74 ± 0.75	0.79 ± 0.26	0.98 ± 0.01	0.97 ± 0.01
H	11.71 ± 2.34	18.26 ± 1.24	33.6 ± 20.81		21.95 ± 0.03	0.08 ± 0.09	0.8 ± 0.05	0.86 ± 0.02

G Oracle Function

Table 4: Examples of what supports the oracle function generates for a given query instruction and environment. The first two examples are from the training Split And the last example is from Split H. Note that we never generate the same instruction as the query instruction in the supports, and we also never generate any Split H instruction in the supports. Also note that in some cases, the environment makes pushing or pulling an object impossible, even though it is in the instruction, see the second row for an example of this.

Environment	Query Instruction	Target Actions	Supports	
	walk to a small circle hesitantly	LTURN(2) WALK STAY WALK STAY WALK STAY WALK STAY RTURN WALK STAY	push a small circle hesitantly	LTURN LTURN WALK STAY WALK STAY WALK STAY WALK STAY RTURN WALK STAY PUSH STAY PUSH STAY PUSH STAY PUSH STAY
			pull a small circle hesitantly	LTURN LTURN WALK STAY WALK STAY WALK STAY WALK STAY RTURN WALK STAY pull STAY pull STAY pull STAY pull STAY
			walk to a small circle while spinning	LTURN(4) LTURN LTURN WALK LTURN(4) WALK LTURN(4) WALK LTURN(4) WALK LTURN(4) RTURN WALK
			walk to a small circle while zigzagging	LTURN LTURN WALK RTURN WALK LTURN WALK WALK WALK
			walk to a small circle	LTURN LTURN WALK WALK WALK WALK RTURN WALK
	push a red small cylinder while zigzagging	WALK RTURN WALK LTURN WALK RTURN WALK LTURN WALK PUSH	walk to a red small cylinder while zigzagging	WALK RTURN WALK LTURN WALK RTURN WALK LTURN WALK
			pull a red small cylinder while zigzagging	WALK RTURN WALK LTURN WALK RTURN WALK LTURN WALK
			push a red small cylinder while spinning	LTURN(4) WALK LTURN(4) WALK LTURN(4) WALK LTURN(4) RTURN WALK LTURN(4) WALK
			push a red small cylinder hesitantly	WALK STAY WALK STAY WALK STAY RTURN WALK STAY WALK STAY
			push a red small cylinder	WALK WALK WALK RTURN WALK WALK
	pull a green big cylinder while spinning	LTURN(4) WALK LTURN(4) LTURN WALK LTURN(4) WALK LTURN(4) WALK LTURN(4) PULL LTURN(4) PULL LTURN(4) PULL LTURN(4) PULL	walk to a green big cylinder while spinning	LTURN(4) WALK LTURN(4) LTURN WALK LTURN(4) WALK LTURN(4) WALK
			push a green big cylinder while spinning	LTURN(4) WALK LTURN(4) LTURN WALK LTURN(4) WALK LTURN(4) WALK LTURN(4) PUSH LTURN(4) PUSH
			pull a green big cylinder while zigzagging	WALK LTURN WALK WALK WALK PULL PULL PULL PULL
			pull a green big cylinder hesitantly	WALK STAY LTURN WALK STAY WALK STAY WALK STAY PULL STAY PULL STAY PULL STAY PULL STAY
			pull a green big cylinder	WALK LTURN WALK WALK WALK PULL PULL PULL PULL

The oracle function generates relevant instrunctions by the use of a templating mechanism, which replaces verbs and adverbs in the sentence with other verbs and adverbs, such that the whole combination is still in distribution, but not the same as the query instruction. The rules of the system are:

- Replace “pull" with “push" and “walk to"

- Replace “walk to" with “push" and “pull" (but not if “while spinning" is the adverb)
- Replace “push" with “walk to" and “pull" (but not if “while spinning" is the adverb)
- Replace “while zigzagging" with “hesitantly", nothing and “while spinning" (but not if “push" is the verb)
- Replace “hesitantly" with “while zigzagging", nothing and “while spinning" (but not if “push" is the verb)
- Replace “while spinning" with “hesitantly", “while zigzagging" and nothing

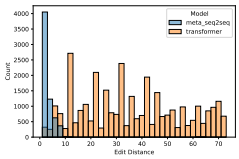
Three examples of what the oracle function generates for a given query instruction and environment can be found in Table 4.

H Failure case analysis

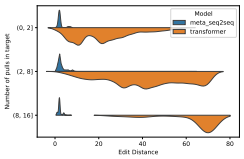
Figure 3: Failure case analysis. In (a) we classify failure cases for the meta-seq2seq model. Note that multiple failures can happen in a single example, so percentages do not add to 100. In (b) and (c), we show the edit distance frequency distribution, as well as edit distance as a function of the number of PULL instructions in the target sequence. Models that generalize poorly will have a larger edit distance for more complex target instructions

Failure reason	% of all failure cases
Did not turn	78.82%
Spurious Pull	33.1%
Missed Pull	8.09%
Other reason	0.058%

(a) Failure case classification



(b) Edit-distance freq. dist.



(c) Pulls vs Edit Distance dist.

We also studied the remaining failure cases on Split H for the best-case version of our model, to see where the remaining challenges are. To do this, we ran the model autoregressively on its instructions and supports without any causally-masked teacher forcing. We observed four types of failure:

- **Did not turn** The agent “missed" a turn instruction when generating an instruction path that requires one, eg, because the target object is not in the same row as the agent. In this case, WALK WALK is generated as opposed to LTURN WALK or RTURN WALK.
- **Spurious Pull** The agent generates a PULL instruction where it should not generate one.
- **Missed Pull** The agent does not generate a PULL instruction where it should generate one.
- **Other reason** The failure is more complex or multi-faceted than can be attributed to the above reasons.

The majority of failures can be attributed to “**Did not turn**". To compute whether the agent is erroneously picking WALK but is uncertain, we compute the mean entropy of the prediction logits in all cases where there is a failure to turn. A high amount of uncertainty should correspond to 0.5. The mean value of the entropy plus standard deviation in this case is 0.22 ± 0.24. This indicates that the agent is somewhat certain about its decision, but there are cases where it is completely certain (in error) or quite uncertain, where further training may improve the situation.

There are also a few cases where the agent generates a PULL instruction or does not generate one when it is expected to (“**Spurious Pull**" and “**Missed Pull**"). We hypothesized that this may be because of an asymmetry between the actions seen for “push while spinning" in the context and “pull while spinning" in the target (the number of PUSH or PULL actions can differ depending the location of the target object and its surroundings), but we didn’t see any concrete relationship here.

We also analyzed the edit distances between the target sequence and the predicted sequences from the model. Because of a lack of teacher-forcing, we expect that the number of exact matches will be about the same, but it is possible that one error could cause compounding errors if the transformer decoder inputs effectively become out of distribution. In practice we found that the edit distances follow a power law. In the majority of cases, only a small number (1 to 3) errors are made throughout

the whole sequence, and only in a small number of cases do we make a large number of errors. Since the length of the novel sequences the model must generate can be quite large, this indicates that the errors made by the model don't come down to a lack of generalization due to poor fitting. We would expect the edit distances to be larger if generalization was the issue. The following example may be illustrative. Say for example the target is "pull while spinning" on an object that needs to be pulled three times. The target target sequence would need to contain `LTURN(4) PULL LTURN(4) PULL LTURN(4) PULL`. A model that failed to generalize would generate in the best case "push while spinning" with targets ending in `LTURN(4) PUSH LTURN(4) PUSH LTURN(4) PUSH` (edit distance of 3), or "pull", with target ending in `PULL PULL PULL` (edit distance of 12). But we see that an order of magnitude more cases have a shorter edit distance in Figure 3, so the issue is not fundamentally down to a lack of generalization.