Modular Networks for Compositional Instruction Following

Rodolfo Corona Daniel Fried Coline Devin Dan Klein Trevor Darrell
UC Berkeley
{rcorona,dfried,coline,klein,trevordarrell}@berkeley.edu

Abstract

Standard architectures used in instruction following often struggle on novel compositions of *subgoals* (e.g. navigating to landmarks or picking up objects) observed during training. We propose a modular architecture for following natural language instructions that describe sequences of diverse subgoals. In our approach, subgoal modules each carry out natural language instructions for a specific subgoal type. A sequence of modules to execute is chosen by learning to segment the instructions and predicting a subgoal type for each segment. When compared to standard, non-modular sequence-to-sequence approaches on ALFRED (Shridhar et al., 2020), a challenging instruction following benchmark, we find that modularization improves generalization to novel subgoal compositions, as well as to environments unseen in training.

1 Introduction

Work on grounded instruction following (MacMahon et al., 2006; Vogel and Jurafsky, 2010; Tellex et al., 2011; Chen and Mooney, 2011; Artzi and Zettlemoyer, 2013) has recently been driven by sequence-to-sequence models (Mei et al., 2016; Hermann et al., 2017), which allow end-to-end grounding of linguistically-rich instructions into equally-rich visual contexts (Misra et al., 2018; Anderson et al., 2018; Chen et al., 2019). These sequence-to-sequence models are *monolithic*: they consist of a single network structure which is applied identically to every example in the dataset.

Monolithic instruction following models typically perform well when evaluated on test data from the same distribution seen during training. However, they often struggle in *compositional generalization*: composing atomic parts, such as actions or goals, where the parts are seen in training but their compositions are not (Lake and Baroni, 2018; Ruis et al., 2020; Hill et al., 2020).

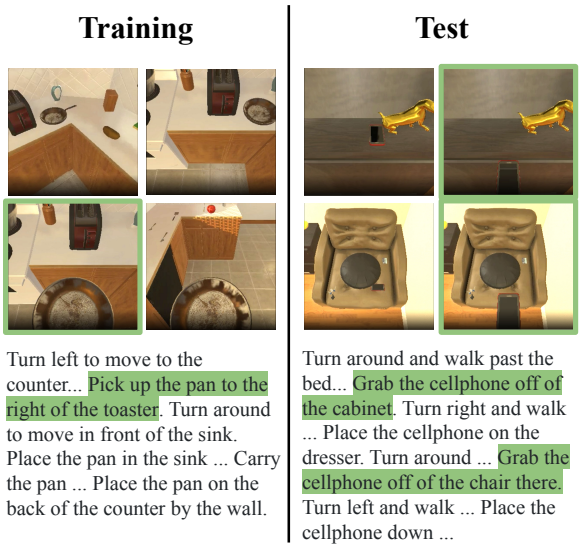

Figure 1: At evaluation time, an instruction following agent may need to generalize both to novel chains of subgoals encountered during training as well as to completely new environments. In the generalization condition above, the agent must generalize to *multiple* pickup actions (in green) at test time, whereas only single ones were seen at training, as well as to a new house. We propose a modular architecture to handle these cases.

In this work, we improve compositional generalization in instruction following with *modular networks*, which have been successful in non-embodied language grounding tasks (Andreas et al., 2016; Hu et al., 2017; Cirik et al., 2018; Yu et al., 2018; Mao et al., 2019; Han et al., 2019) and in following synthetic instructions or symbolic policy descriptions (Andreas et al., 2017; Oh et al., 2017; Das et al., 2018). Modular networks split the decision making process into a set of neural modules. Modules are each specialized for some function, composed into a structure specific to each example, and trained jointly to complete the task.

Prior work has found that modular networks often perform well in compositional generalization

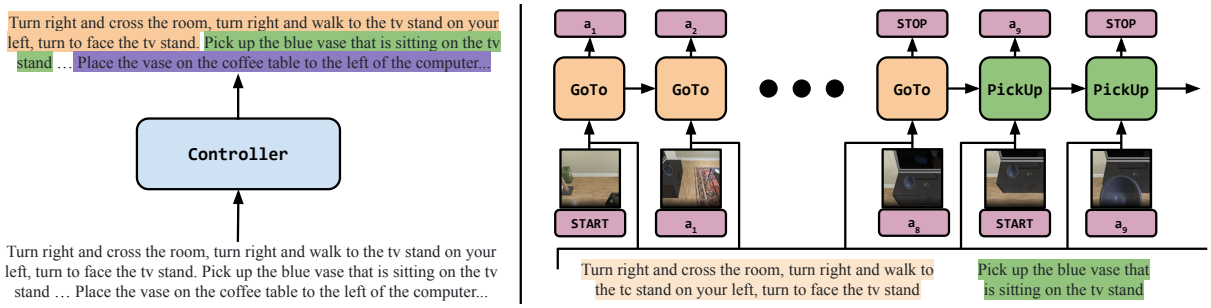

Figure 2: Our modular approach first uses a *controller* (left) trained with supervised learning to segment a given instruction and label segments with subgoal types (*e.g.* GOTO, PICKUP) to execute. These subgoal types are used to chain together *modules* (right) to carry out instructions in the environment. Each module is a separately-parameterized sequence-to-sequence model that conditions on an attended representation of the instruction sequence, the visual observations, and the action taken at the previous timestep. Modules pass recurrent hidden states to each other.

because of their composable structure (Devin et al., 2017; Andreas et al., 2017; Bahdanau et al., 2019; Purushwalkam et al., 2019), and that they can generalize to new environments or domains through module specialization (Hu et al., 2019; Blukis et al., 2020). However, all this work has either focused on grounding tasks without a temporal component or used a network structure which is not predicted from language.

We propose a modular architecture for embodied vision-and-language instruction following[1], and find that this architecture improves generalization on unseen compositions of *subgoals* (such as navigation, picking up objects, cleaning them, etc.). We define separate sequence-to-sequence modules per type of subgoal. These modules are strung together to execute complex high-level tasks. We train a controller to predict a sequence of subgoal types from language instructions, which determines the order in which to execute the modules.

We evaluate models on the ALFRED dataset (Shridhar et al., 2020), an instruction-following benchmark containing a diverse set of household tasks. We focus on compositional generalization: carrying out instructions describing novel high-level tasks, containing novel compositions of actions (see Figure 1 for an example). We find that our modular model improves performance on average across subgoal types when compared to a standard, monolithic sequence-to-sequence architecture. Additionally, we find improved generalization to environments not seen in training.

[1]Code and dataset splits may be found at github.com/rcorona/modular_compositional_alfred.

2 Modular Instruction Following Networks

We focus on following instructions in embodied tasks involving navigation and complex object interactions, as shown in Figure 2.

In training, each set of *full instructions* (*e.g.* "Turn right and cross the room ... Place the vase on the coffee table to the left of the computer.") is paired with a *demonstration* of image observations and actions. In training, we further assume that the full instruction is segmented into *subgoal instructions*, and each subgoal instruction is labeled with one of a small number (in our work, 8) of *subgoal types* , *e.g.* ["Walk to the coffee maker.": GOTO], ["Pick up the dirty mug...": PICKUP], . . . , and paired with the corresponding segment of the demonstration.

During evaluation, the agent is given only full instructions (which are unsegmented and unlabeled), and must predict a sequence of actions to carry out the instructions, conditioning on the image observations it receives.

Our modular architecture for compositional instruction following consists of a *high-level controller* (Figure 2, left), and *modules* for each subgoal type (Figure 2, right). The high-level controller chooses modules to execute in sequence based on the natural language instructions, and each chosen module executes until it outputs a STOP action. The modules all share the same sequence-to-sequence architecture, which is the same as the monolithic architecture. We initialize each module's parameters with parameters from the monolithic model, and then fine-tune the parameters of each module to specialize for its subgoal.

2.1 Instruction-Based Controller

Our instruction-based controller is trained to segment a full instruction into sub-instructions and predict the subgoal type for each sub-instruction. We use a linear chain CRF (Lafferty et al., 2001) that conditions on a bidirectional-LSTM encoding of the full instruction and predicts tags for each word, which determine the segmentation and sequence of subgoal types. This model is based on standard neural segmentation and labelling models (Huang et al., 2015; Lample et al., 2016).

We train the controller on the ground-truth instruction segmentations and subgoal sequence labels, and in evaluation use the model to predict segmentations and their associated subgoal sequences (Figure 2, top left). This predicted sequence of subgoals determines the order to execute the modules (Figure 2, right). We use a BIO chunking scheme to jointly segment the instruction and predict a subgoal label for each segment.

Formally, for a full instruction of length N, the controller defines a distribution over subgoal tags $s_{1:N}$ for each word given the instruction $x_{1:N}$ as

$$p(s_{1:N} \mid x_{1:N}) \propto \exp \sum_{n=1}^{N} \left(U_{s_n} + B_{s_{n-1},s_n}\right)$$

The subgoal tag scores U_{s_n} for word n are given by a linear projection of bidirectional LSTM features for the word at position n. The tag transition scores B_{s_{n-1},s_n} are learned scalar parameters.

In training, we supervise $s_{1:N}$ using the segmentation of the instruction $x_{1:N}$ into K subgoal instructions and the subgoal label for each instruction. To predict subgoals for a full instruction in evaluation, we obtain $\arg\max_{s_{1:N}} p(s_{1:N} \mid x_{1:N})$ using Viterbi decoding, which provides a segmentation into sub-instructions and a subgoal label for each sub-instruction.

The controller obtains 96% exact match accuracy on subgoal sequences on validation data.

2.2 Module Architecture

Our modularized architecture may be seen in Figure 2, right. The architecture consists of 8 independent modules, one for each of the 8 subgoals in the domain (*e.g.* GoTo, PickUp). For each module, we use the same architecture as Shridhar et al. (2020)'s monolithic model. This is a sequence-to-sequence model composed of an LSTM decoder taking as input an attended embedding of the natural language instruction, pretrained ResNet-18 (He et al., 2016) features of the image observations, and the previous action's embedding. Hidden states are passed between the modules' LSTM decoders at subgoal transitions (Figure 2, right).

At each time step, each module M^i computes its hidden state based on the last time step's action a_{t-1}, the current time step's observed image features o_t, an attended language embedding $\hat{x}_t^i$, and the previous hidden state h_{t-1}^i:

$$e_t^i = [a_{t-1}; o_t; \hat{x}_t^i]$$
$$h_t^i = \text{LSTM}_i(e_t^i, h_{t-1}^i)$$

Each module's attended language embedding $\hat{x}_t^i$ is produced using its own attention mechanism over embeddings $X = x_{1:N}$ of the language instruction, which are produced by a bidirectional LSTM encoder:

$$z_t^i = (W_x^i h_{t-1}^i)^\top X$$
$$\alpha_t^i = \text{Softmax}(z_t^i)$$
$$\hat{x}_t^i = (\alpha_t^i)^\top X$$

Finally, the action a_t and object interaction mask m_t are predicted from h_t^i and e_t^i with a linear layer and a deconvolution network respectively. More details about this architecture can be found in Shridhar et al. (2020). Both the action and mask decoders, well as the language encoder, are shared across modules.[2]

Our use of subgoal modules is similar to the hierarchical policy approaches of Andreas et al. (2017), Oh et al. (2017), and Das et al. (2018). However, in those approaches, the input to each module is symbolic (*e.g.* FIND[KITCHEN]). In contrast, all modules in our work condition directly on natural language.

2.3 Training

We first pre-train the monolithic model by maximizing the likelihood of the ground-truth trajectories in the training data (Shridhar et al., 2020). We train for up to 20 epochs using the Adam optimizer (Kingma and Ba, 2014) with early stopping on validation data (see Appendix A.1 for hyperparameters). We use this monolithic model to initialize the parameters of each of the modules, which have identical architecture to the monolithic model, and

[2]The modules' instruction encoder is separate from the controller's encoder (Sec. 2.1), as we found it possible to achieve high performance on the subgoal prediction task using a smaller encoder than the one used by the modules.

fine-tune them using the same training and early stopping procedure on the same validation data,[3] allowing the monolithic model's parameters to specialize for each module. Each module predicts only the actions for its segment of each trajectory; however, modules are jointly fine-tuned, passing hidden states (and gradients) from module to module.

3 Generalization Evaluation

We evaluate models on out-of-domain generalization in two conditions (see below) using the ALFRED benchmark (Shridhar et al., 2020), comparing our modular approach to their non-modular sequence-to-sequence model. ALFRED is implemented in AI2-THOR 2.0 (Kolve et al., 2017), which contains a set of simulated environments with realistic indoor scene renderings and object interactions.

The dataset contains approximately 25K expert instruction-trajectory pairs, comprised of 3 instructions for each of 8K unique trajectories. The instructions include both a high level instruction and a sequence of low level instructions. In our experiments, we do not use the high level instructions, which Shridhar et al. (2020) found to produce comparable results when evaluated on generalization to unseen environments with these architectures.

Figure 1 shows two example trajectories and their associated instructions. Trajectories are composed (see Sec. 2) of sequences of eight different types of subgoals: navigation (GoTo) and a variety of object interactions (*e.g.* PickUp, Clean, Heat). Each subgoal's subtrajectory is composed of a sequence of low-level discrete actions which specify commands for navigation or object interactions (which are accompanied by image segmentations to choose the object to interact with).

3.1 Generalization Conditions

The ALFRED dataset was constructed to test generalization to novel instructions and unseen environments. However, all evaluation trajectories in the dataset correspond to sequences of subgoals that are seen during training. For example, some training and evaluation instances might both correspond to the underlying subgoal sequence GoTo, PickUp, GoTo, Put, but differ in their low-level actions, their language descriptions, and possibly also the environments they are carried out in.

[3] Additionally, we append a special STOP action to the end of each module's action sequence so that it can predict when to give control back to the high-level controller.

Novel Tasks. We evaluate models' ability to generalize to different high-level tasks (compositions of subgoals) than seen in training. The dataset contains seven different task types, such as *Pick & Place*, as described in Appendix B.1. We hold out two task types and evaluate models on their ability to generalize to them: *Pick Two & Place* and *Stack & Place*. These tasks are chosen because they contain subgoal types that are all individually seen in training, but typically in different sequences.

We create generalization splits *pick-2-seen* and *pick-2-unseen* by filtering the *seen* and *unseen* splits below to contain only *Pick Two & Place* tasks, and remove all *Pick Two & Place* tasks from the training data. We create splits *stack-seen* and *stack-unseen* for *Stack & Place* similarly.

Novel Instructions and Environments This is the standard condition defined in the original ALFRED dataset. There are two held-out validation sets: *seen*, which tests generalization to novel instructions and trajectories but through environments seen during training, and *unseen*, which tests generalization to novel environments: rooms with new layouts, object appearances, and furnishings.

3.2 Results

We compare our modular architecture with the monolithic baseline, averaging performance over models trained from 3 random seeds. For each generalization condition, we measure success rates over full trajectories as well as over each subgoal type independently. Due to the challenging nature of the domain, subgoal evaluation provides finer-grained comparisons than full trajectories.

We use the same evaluation methods and metrics as in Shridhar et al. (2020). Success rates are weighted by path lengths to penalize successful trajectories which are longer than the ground-truth demonstration trajectory. To evaluate full trajectories, we measure path completion: the portion of subgoals completed within the full trajectories. To evaluate the subgoals independently, we advance the model along the expert trajectory up until the point where a given subgoal begins (to maintain a history of actions and observations), then require the model to carry out the subgoal from that point.

We also report results from Shridhar et al. (2020) and Singh et al. (2020). We note that the approach of Singh et al. (2020) obtains higher performance on full trajectories than the system of Shridhar et al. (2020) (which we base our approach on)

	Model	*Clean*	*Cool*	*Goto*	*Heat*	*Pickup*	*Put*	*Slice*	*Toggle*	*Avg.*
seen	S+	82	87	49	85	32	80	23	97	67
	MOCA	79	87	54	84	53	62	51	93	70
	Mono.	82	**88***	**52****	**82****	37	**81**	34	**98**	**69***
	Mod.	82	86	43	80	**41****	80	**37**	97	68
unseen	S+	21	94	21	88	20	51	14	54	45
	MOCA	71	38	32	86	44	39	55	11	47
	Mono.	28	90	**23****	**89****	25	48	25	42	46
	Mod.	**67****	**94***	14	85	**28****	**55**	**39***	**73***	**57****

(a) Standard (novel environments) validation splits

	Model	*Goto*	*Pickup*	*Put*	*Avg.*
seen	Mono.	**18***	21	48	29
	Mod.	14	**35***	**63***	**37***
unseen	Mono.	14	12	14	13
	Mod.	14	**25****	**33****	**24****

(b) Pick-2 task splits

	Model	*Goto*	*Pickup*	*Put*	*Avg.*
seen	Mono.	**28***	15	54	32
	Mod.	21	**27***	**58***	**35***
unseen	Mono.	**19**	7	25	17
	Mod.	14	**16***	**28**	**19***

(c) Stack task splits

Table 1: Path weighted subgoal success percentages, by subgoal type, on the various generalization splits, and averaged across subgoal types (*Avg.*). We compare the performance of the monolithic (*Mono.*) model to our modular model (*Mod.*). The modular model generalizes better on average to unseen environments (standard-unseen) and to both seen and unseen environments for two held-out task types: *Pick-2* and *Stack*. Bolded numbers show the best model between Mono and Modular, with * and ** denoting differences that are statistically significant at the $p < 0.15$ and $p < 0.05$ levels, respectively, by a one-tailed t-test. S+ gives results from Shridhar et al. (2020) and MOCA from Singh et al. (2020).

primarily by introducing a modular object interaction architecture (shared across all subgoals) and a pre-trained object segmentation model. These techniques could also be incorporated into our approach, which uses modular components for individual subgoal types.

Novel Tasks. Table 1 shows for each split the success rates on subgoals appearing in at least 50 validation examples. The modular outperforms the monolithic model on both seen and unseen splits (Tables 1b and 1c). Full trajectory results for novel task generalization are shown in Table 2. In the double generalization condition (unseen environments for the held-out pick-2 and stack tasks) on full trajectories, neither model completes subgoals successfully. Overall, we find that modularity helps across most generalization conditions.

Generalization to novel environments. We also compare models on generalization to unseen environments. In the independent subgoal evaluation, the monolithic and modular models perform equally on average in the standard-seen split (Table 1a, top). However, in the standard-unseen split (Table 1a, bottom), our modular model outperforms the baseline substantially, with an average success rate of 57% compared to the monolithic model's 46%. (On subgoal types not shown, the modular model still outperforms the monolithic, by margins up to 16%.) In the full trajectory results (Table 2) we see comparable performance between the monolithic and modular models on unseen environments.

Model	Standard seen	Standard unseen	Pick-2 seen	Stack seen
S+	9.4 (5.7)	7.4 (4.7)	—	—
MOCA	28.5 (22.3)	13.4 (8.3)	—	—
Mono.	**10.9 (7.0)**	**7.1** (4.9)	1.3 (1.6)	1.3 (0.3)
Mod.	9.1 (6.6)	7.0 **(5.5)**	**1.5** (1.6)	**1.5 (0.4)**

Table 2: We compare performance of the monolithic and modular models on full trajectories, reporting the percentages of subgoals correctly completed. Numbers in parentheses weight these percentages by path length. S+ gives results from Shridhar et al. (2020), and MOCA from Singh et al. (2020).

4 Conclusions

We introduced a novel modular architecture for grounded instruction following where each module is a sequence-to-sequence model conditioned on natural language instructions. With the ALFRED dataset as a testbed, we showed that our modular model achieves better out-of-domain generalization, generalizing better at the subgoal level to novel task compositions and unseen environments than the monolithic model used in prior work. All of the module types in our model currently use separate parameterizations but identical architectures; future work might leverage the modularity of our approach by using specialized architectures, training procedures, or loss functions for each subgoal type. Furthermore, unsupervised methods for jointly segmenting instructions and trajectories without requiring labeled subgoal labels and alignments would be a valuable addition to our framework.

Acknowledgments

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship Program under Grant No. DGE 1752814, a Ford Foundation fellowship to the first author, a Google PhD fellowship to the second author, and by DARPA through the XAI program and the LwLL program.

References

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 166–175.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 39–48.

Yoav Artzi and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics (TACL)*, 1(1):49–62.

Dzmitry Bahdanau, Shikhar Murty, Michael Noukhovitch, Thien Huu Nguyen, Harm de Vries, and Aaron Courville. 2019. Systematic generalization: what is required and can it be learned? In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Valts Blukis, Yannick Terme, Eyvind Niklasson, Ross A. Knepper, and Yoav Artzi. 2020. Learning to map natural language instructions to physical quadcopter control using simulated flight. volume 100 of *Proceedings of Machine Learning Research*, pages 1415–1438. PMLR.

David L. Chen and Raymond J. Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Howard Chen, Alane Shur, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Volkan Cirik, Taylor Berg-Kirkpatrick, and Louis-Phillippe Morency. 2018. Using syntax to ground referring expressions in natural images. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. 2018. Neural modular control for embodied question answering. In *Proceedings of the Conference on Robot Learning (CoRL)*.

Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. 2017. Learning modular neural network policies for multi-task and multi-robot transfer. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176.

Chi Han, Jiayuan Mao, Chuang Gan, Josh Tenenbaum, and Jiajun Wu. 2019. Visual concept-metaconcept learning. In *Advances in Neural Information Processing Systems (NIPS)*, pages 5002–5013.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778.

Karl Moritz Hermann, Felix Hill, Simon Green, Fumin Wang, Ryan Faulkner, Hubert Soyer, David Szepesvari, Wojciech Marian Czarnecki, Max Jaderberg, Denis Teplyashin, Marcus Wainwright, Chris Apps, Demis Hassabis, and Phil Blunsom. 2017. Grounded language learning in a simulated 3d world. *CoRR*, abs/1706.06551.

Felix Hill, Andrew Lampinen, Rosalia Schneider, Stephen Clark, Matthew Botvinick, James L McClelland, and Adam Santoro. 2020. Environmental drivers of systematicity and generalization in a situated agent. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. 2017. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 804–813.

Ronghang Hu, Daniel Fried, Anna Rohrbach, Dan Klein, Trevor Darrell, and Kate Saenko. 2019. Are you looking? grounding to multiple modalities in vision-and-language navigation. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 6551–6557. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *arXiv preprint arXiv:1508.01991*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. 2017. AI2-THOR: An interactive 3D environment for visual AI. *arXiv preprint arXiv:1712.05474*.

John Lafferty, Andrew McCallum, and Fernando CN Pereira. 2001. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Brenden M Lake and Marco Baroni. 2018. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. *Proceedings of the International Conference on Machine Learning (ICML)*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*.

Matt MacMahon, Brian Stankiewicz, and Benjamin Kuipers. 2006. Walk the talk: Connecting language, knowledge, and action in route instructions. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. 2019. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Hongyuan Mei, Mohit Bansal, and Matthew Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*.

Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. 2018. Mapping instructions to actions in 3D environments with visual goal prediction. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2667–2678.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 2661–2670. JMLR.org.

Senthil Purushwalkam, Maximilian Nickel, Abhinav Gupta, and Marc'Aurelio Ranzato. 2019. Task-driven modular networks for zero-shot compositional learning. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 3593–3602.

Laura Ruis, Jacob Andreas, Marco Baroni, Diane Bouchacourt, and Brenden M Lake. 2020. A benchmark for systematic generalization in grounded language understanding. *arXiv preprint arXiv:2003.05161*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A benchmark for interpreting grounded instructions for everyday tasks. In *Computer Vision and Pattern Recognition (CVPR)*.

Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. 2020. Moca: A modular object-centric approach for interactive instruction following. *arXiv preprint arXiv:2012.03208*.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R Walter, Ashis Gopal Banerjee, Seth J Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Conference on Artificial Intelligence (AAAI)*, volume 1, page 2.

Adam Vogel and Dan Jurafsky. 2010. Learning to follow navigational directions. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 806–814. Association for Computational Linguistics.

Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. 2018. Mattnet: Modular attention network for referring expression comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hyperparameter	Value
Optimizer	Adam
Learning Rate	1e-4
Batch Size	8
Hidden State Dim	512
Word/Action Embedding Dim	100
Zero-Goal	True
Zero-Instr	False
Lang. Dropout	0.0
Vision Dropout	0.0
Input Dropout	0.0
Attn. Dropout	0.0
Actor Dropout	0.0
LSTM Dropout	0.3
Mask Loss Wt.	1.0
Action Loss Wt.	1.0

Table 3: Model hyperparameters. These settings largely follow the default parameters used by Shridhar et al. (2020).

A Implementation Details

A.1 Model and Training Hyperparameters

We list the hyperparameters used for all models in Table 3, we refer the reader to (Shridhar et al., 2020) for more details on the usage of each hyperparameter. Submodules are each structured identically to the monolithic baseline (e.g. each one had a 512 dimensional hidden state).

A.2 Hardware and Training Times

Models were trained on a Quadro RTX 6000 24GB GPU running on a machine with a 14 core Intel Xeon Gold 5120 CPU, with a runtime of approximately 14 hours. Evaluation was done on a V100 16GB GPU on a machine with a 4-core CPU. Subgoal evaluation took approximately 8 hours per split, and full trajectory evaluation approximately 1 hour.

A.3 Evaluation

We evaluate our model using the evaluation code provided by Shridhar et al. (2020).[4]

B ALFRED Dataset Details

In ALFRED, the agent observes a first person view, navigates with discrete grid movement, and uses objects by outputting a segmentation mask over its image observation. The dataset contains approximately 25K expert instruction-trajectory pairs, pertaining to about 8K unique trajectories.

B.1 Task Types

The dataset contains demonstrations for 7 different kinds of tasks.

Pick & Place The agent must pickup a specified object, bring it to a destination, and place it. For example, “Pick up a vase, place it on the coffee table."

Examine in Light The agent must pickup an object and bring it to a light source. For example, “Examine the remote control under the light of the floor lamp ."

Heat & Place The agent must pickup an object, put it in the microwave, toggle the microwave, take the object out of the microwave, and finally place the heated object at a specified location. For example: “Put a heated apple next to the lettuce on the middle shelf in the refrigerator."

Cool & Place This is the same as above, but with a refrigerator instead of a microwave. For example, “Drop a cold potato slice in the sink."

Clean & Place The agent must put an object into the sink and turn on the water to clean the object. Then, it must be placed at a specified location. For example, “Put a washed piece of lettuce on the counter by the sink."

Stack & Place The agent must pick up an object, place it into a receptacle, and then bring the stacked objects to a specified location and place them. For example, “Move the pan on the stove with a slice of tomato in it to the table."

Pick Two & Place The agent must pickup an object, place it somewhere, then pick up another instance of that object and put it in the same place. For example, “Place two CDs in top drawer of black cabinet."

These last two task types, **Stack & Place** and **Pick Two & Place**, are the ones held out in the *Novel Tasks* generalization experiments.

[4] https://github.com/askforalfred/alfred