BabyWalk: Going Farther in Vision-and-Language Navigation by Taking Baby Steps

Wang Zhu[*1] **Hexiang Hu**[*2] **Jiacheng Chen**[2] **Zhiwei Deng**[3]
Vihan Jain[4] **Eugene Ie**[4] **Fei Sha**[†2,4]

[1]Simon Fraser University [2]University of Southern California [3]Princeton University
[4]Google Research

Abstract

Learning to follow instructions is of fundamental importance to autonomous agents for vision-and-language navigation (VLN). In this paper, we study how an agent can navigate long paths when learning from a corpus that consists of shorter ones. We show that existing state-of-the-art agents do not generalize well. To this end, we propose BabyWalk, a new VLN agent that is learned to navigate by decomposing long instructions into shorter ones (BabySteps) and completing them sequentially. A special design memory buffer is used by the agent to turn its past experiences into contexts for future steps. The learning process is composed of two phases. In the first phase, the agent uses imitation learning from demonstration to accomplish BabySteps. In the second phase, the agent uses curriculum-based reinforcement learning to maximize rewards on navigation tasks with increasingly longer instructions. We create two new benchmark datasets (of long navigation tasks) and use them in conjunction with existing ones to examine BabyWalk's generalization ability. Empirical results show that BabyWalk achieves *state-of-the-art* results on several metrics, in particular, is able to follow long instructions better. The codes and the datasets are released on our project page https://github.com/Sha-Lab/babywalk.

1 Introduction

Autonomous agents such as household robots need to interact with the physical world in multiple modalities. As an example, in vision-and-language navigation (VLN) (Anderson et al., 2018), the agent moves around in a photo-realistic simulated environment (Chang et al., 2017) by following a sequence of natural language instructions. To infer its whereabouts so as to decide its moves, the agent infuses its visual perception, its trajectory and the instructions (Fried et al., 2018; Anderson et al., 2018; Wang et al., 2019; Ma et al., 2019a,b).

Arguably, the ability to understand and follow the instructions is one of the most crucial skills to acquire by VLN agents. Jain et al. (2019) shows that the VLN agents trained on the originally proposed dataset ROOM2ROOM (*i.e.* R2R thereafter) do *not* follow the instructions, despite having achieved high success rates of reaching the navigation goals. They proposed two remedies: a new dataset ROOM4ROOM (or R4R) that doubles the path lengths in the R2R, and a new evaluation metric Coverage weighted by Length Score (CLS) that measures more closely whether the ground-truth paths are followed. They showed optimizing the fidelity of following instructions leads to agents with desirable behavior. Moreover, the long lengths in R4R are informative in identifying agents who score higher in such fidelity measure.

In this paper, we investigate another crucial aspect of following the instructions: *can a VLN agent generalize to following longer instructions by learning from shorter ones?* This aspect has important implication to real-world applications as collecting annotated long sequences of instructions and training on them can be costly. Thus, it is highly desirable to have this generalization ability. After all, it seems that humans can achieve this effortlessly[1].

To this end, we have created several datasets of longer navigation tasks, inspired by R4R (Jain et al., 2019). We trained VLN agents on R4R and use the agents to navigate in ROOM6ROOM (*i.e.*, R6R) and ROOM8ROOM (*i.e.*, R8R). We contrast to the performance of the agents which are trained on those datasets directly ("in-domain"). The results

[*] Author contributed equally
[†] On leave from University of Southern California

[1] Anecdotally, we do not have to learn from long navigation experiences. Instead, we extrapolate from our experiences of learning to navigate in shorter distances or smaller spaces (perhaps a skill we learn when we were babies or kids).

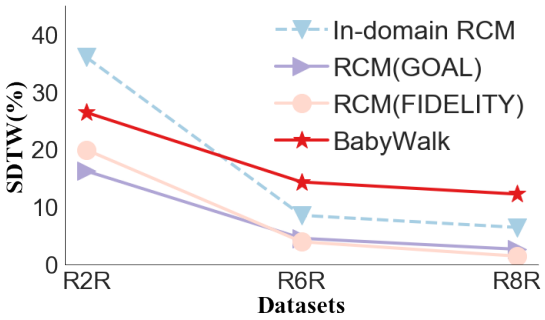

Figure 1: Performance of various VLN agents on generalizing from shorter navigation tasks to longer ones. The vertical axis is the newly proposed path-following metric SDTW (Magalhaes et al., 2019), the higher the better. BABYWALK generalizes better than other approaches across different lengths of navigation tasks. Sometimes, it even outperforms in-domain agents (the dashed line). See texts for details.

are shown in Fig. 1.

Our findings are that the agents trained on R4R (denoted by the purple and the pink solid lines) perform significantly worse than the *in-domain* agents (denoted the light blue dashed line). Also interestingly, when such *out-of-domain* agents are applied to the dataset R2R with *shorter* navigation tasks, they also perform significantly worse than the corresponding in-domain agent despite R4R containing many navigation paths from R2R. Note that the agent trained to optimize the aforementioned fidelity measure (RCM(fidelity)) performs better than the agent trained to reach the goal only (RCM(goal)), supporting the claim by Jain et al. (2019) that following instructions is a more meaningful objective than merely goal-reaching. Yet, the fidelity measure itself is not enough to enable the agent to transfer well to longer navigation tasks.

To address these deficiencies, we propose a new approach for VLN. The agent follows a long navigation instruction by decomposing the instruction into shorter ones ("micro-instructions", *i.e.*, BABY-STEPS), each of which corresponds to an intermediate goal/task to be executed sequentially. To this end, the agent has three components: (a) a memory buffer that summarizes the agent's experiences so that the agent can use them to provide the context for executing the next BABY-STEP. (b) the agent first learns from human experts in "bite-size". Instead of trying to imitate to achieve the ground-truth paths as a whole, the agent is given the pairs of a BABY-STEP and the corresponding human expert path so that it can learn policies of actions from shorter instructions. (c) In the second stage of learning, the agent refines the policies by curriculum-based reinforcement learning, where the agent is given increasingly longer navigation tasks to achieve. In particular, this curriculum design reflects our desiderata that the agent optimized on shorter tasks should generalize well to slightly longer tasks and then much longer ones.

While we do not claim that our approach faithfully simulates human learning of navigation, the design is loosely inspired by it. We name our approach BABYWALK and refer to the intermediate navigation goals in (b) as BABY-STEPS. Fig. 1 shows that BABYWALK (the red solid line) significantly outperforms other approaches and despite being out-of-domain, it even exceeds the performance of in-domain agents on R6R and R8R.

The effectiveness of BABYWALK also leads to an interesting twist. As mentioned before, one of the most important observations by Jain et al. (2019) is that the original VLN dataset R2R fails to reveal the difference between optimizing goal-reaching (thus ignoring the instructions) and optimizing the fidelity (thus adhering to the instructions). Yet, leaving details to section 5, we have also shown that applying BABYWALK to R2R can lead to equally strong performance on generalizing from shorter instructions (*i.e.*, R2R) to longer ones.

In summary, in this paper, we have demonstrated empirically that the current VLN agents are ineffective in generalizing from learning on shorter navigation tasks to longer ones. We propose a new approach in addressing this important problem. We validate the approach with extensive benchmarks, including ablation studies to identify the effectiveness of various components in our approach.

2 Related Work

Vision-and-Language Navigation (VLN) Recent works (Anderson et al., 2018; Thomason et al., 2019; Jain et al., 2019; Chen et al., 2019; Nguyen and Daumé III, 2019) extend the early works of instruction based navigation (Chen and Mooney, 2011; Kim and Mooney, 2013; Mei et al., 2016) to photo-realistic simulated environments. For instance, Anderson et al. (2018) proposed to learn a multi-modal Sequence-to-Sequence agent (Seq2Seq) by imitating expert demonstration. Fried et al. (2018) developed a method that augments the paired instruction and demonstration data using a learned speaker model, to teach the navigation

agent to better understand instructions. Wang et al. (2019) further applies reinforcement learning (RL) and self-imitation learning to improve navigation agents. Ma et al. (2019a,b) designed models that track the execution progress for a sequence of instructions using soft-attention.

Different from them, we focus on transferring an agent's performances on shorter tasks to longer ones. This leads to designs and learning schemes that improve generalization across datasets. We use a memory buffer to prevent mistakes in the distant past from exerting strong influence on the present. In imitation learning stage, we solve fine-grained subtasks (BABY-STEPS) instead of asking the agent to learn the navigation trajectory as a whole. We then use curriculum-based reinforcement learning by asking the agent to follow increasingly longer instructions.

Transfer and Cross-domain Adaptation There have been a large body of works in transfer learning and generalization across tasks and environments in both computer vision and reinforcement learning (Andreas et al., 2017; Oh et al., 2017; Zhu et al., 2017a,b; Sohn et al., 2018; Hu et al., 2018). Of particular relevance is the recent work on adapting VLN agents to changes in visual environments (Huang et al., 2019; Tan et al., 2019). To our best knowledge, this work is the first to focus on adapting to a simple aspect of language variability — the length of the instructions.

Curriculum Learning Since proposed in (Bengio et al., 2009), curriculum learning was successfully used in a range of tasks: training robots for goal reaching (Florensa et al., 2017), visual question answering (Mao et al., 2019), image generation (Karras et al., 2018). To our best knowledge, this work is the first to apply the idea to learning in VLN.

3 Notation and the Setup of VLN

In the VLN task, the agent receives a natural language instruction X composed of a sequence of sentences. We model the agent with an *Markov Decision Process* (MDP) which is defined as a tuple of a state space $\mathcal{S}$, an action space $\mathcal{A}$, an initial state $\boldsymbol{s}_1$, a stationary transition dynamics $\rho : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, and the discount factor γ for weighting future rewards. The agent acts according to a policy $\boldsymbol{\pi} : \mathcal{S} \times \mathcal{A} \to 0 \cup R^+$. The state and action spaces are defined the same as in (Fried et al., 2018) (cf. § 4.4 for details).

For each X, the sequence of the pairs $(\boldsymbol{s}, \boldsymbol{a})$ is called a trajectory $\mathrm{Y} = \left\{\boldsymbol{s}_1, \boldsymbol{a}_1, \ldots, \boldsymbol{s}_{|\mathrm{Y}|}, \boldsymbol{a}_{|\mathrm{Y}|}\right\}$ where $|\cdot|$ denotes the length of the sequence or the size of a set. We use $\hat{\boldsymbol{a}}$ to denote an action taken by the agent according to its policy. Hence, $\hat{\mathrm{Y}}$ denotes the agent's trajectory, while Y (or $\boldsymbol{a}$) denotes the human expert's trajectory (or action). The agent is given training examples of (X, Y) to optimize its policy to maximize its expected rewards.

In our work, we introduce additional notations in the following. We will segment a (long) instruction X into multiple shorter sequences of sentences $\{\boldsymbol{x}_m, m = 1, 2, \cdots, \mathrm{M}\}$, to which we refer as BABY-STEPS. Each $\boldsymbol{x}_m$ is interpreted as a micro-instruction that corresponds to a trajectory by the agent $\hat{\boldsymbol{y}}_m$ and is aligned with a part of the human expert's trajectory, denoted as $\boldsymbol{y}_m$. While the alignment is not available in existing datasets for VLN, we will describe how to obtain them in a later section (§ 4.3). Throughout the paper, we also freely interexchange the term "following the mth micro-instruction", "executing the BABY-STEP $\boldsymbol{x}_m$", or "complete the mth subtask".

We use $t \in [1, |\mathrm{Y}|]$ to denote the (discrete) time steps the agent takes actions. Additionally, when the agent follows $\boldsymbol{x}_m$, for convenience, we sometimes use $t_m \in [1, |\hat{\boldsymbol{y}}_m|]$ to index the time steps, instead of the "global time" $t = t_m + \sum_{i=1}^{m-1} |\hat{\boldsymbol{y}}_i|$.

4 Approach

We describe in detail the 3 key elements in the design of our navigation agent: (i) a memory buffer for storing and recalling past experiences to provide contexts for the current navigation instruction (§ 4.1); (ii) an imitation-learning stage of navigating with short instructions to accomplish a single BABY-STEP (§ 4.2.1); (iii) a curriculum-based reinforcement learning phase where the agent learns with increasingly longer instructions (*i.e.* multiple BABY-STEPS) (§ 4.2.2). We describe new benchmarks created for learning and evaluation and key implementation details in § 4.3 and § 4.4 (with more details in the Suppl. Material).

4.1 The BABYWALK Agent

The basic operating model of our navigation agent BABYWALK is to follow a "micro instruction" $\boldsymbol{x}_m$ (*i.e.*, a short sequence of instructions, to which we also refer as BABY-STEP), conditioning on the context $\hat{\boldsymbol{z}}_m$ and to output a trajectory $\hat{\boldsymbol{y}}_m$. A schematic

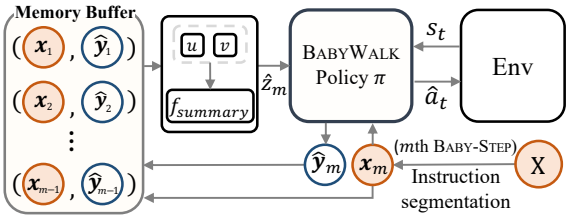

Figure 2: **The BabyWalk agent** has a memory buffer storing its past experiences of instructions $\boldsymbol{x}_m$, and its trajectory $\hat{\boldsymbol{y}}_m$. When a new Baby-Step $\boldsymbol{x}_m$ is presented, the agent retrieves from the memory a summary of its experiences as the history context. It takes actions conditioning on the context (as well as its state $\boldsymbol{s}_t$ and the previous action $\hat{\boldsymbol{a}}_t$). Upon finishing following the instruction. the trajectory $\hat{\boldsymbol{y}}_m$ is then sent to the memory to be remembered.

diagram is shown in Fig. 2. Of particularly different from previous approaches is the introduction of a novel memory module. We assume the Baby-Steps are given in the training and inference time – § 4.3 explains how to obtain them if not given *a prior* (Readers can directly move to that section and return to this part afterwards). The left of the Fig. 3 gives an example of those micro-instructions.

Context The context is a summary of the past experiences of the agent, namely the previous $(m-1)$ mini-instructions and trajectories:

$$\hat{\boldsymbol{z}}_m = g\big(f_{\text{SUMMARY}}(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{m-1}), f_{\text{SUMMARY}}(\hat{\boldsymbol{y}}_1, \cdots, \hat{\boldsymbol{y}}_{m-1})\big) \quad (1)$$

where the function g is implemented with a multi-layer perceptron. The summary function f_{SUMMARY} is explained in below.

Summary To map variable-length sequences (such as the trajectory and the instructions) to a single vector, we can use various mechanisms such as LSTM. We reported an ablation study on this in § 5.3. In the following, we describe the "forgetting" one that weighs more heavily towards the most recent experiences and performs the best empirically.

$$f_{\text{SUMMARY}}(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_{m-1}) = \sum_{i=1}^{m-1} \alpha_i \cdot u(\boldsymbol{x}_i) \quad (2)$$

$$f_{\text{SUMMARY}}(\hat{\boldsymbol{y}}_1, \cdots, \hat{\boldsymbol{y}}_{m-1}) = \sum_{i=1}^{m-1} \alpha_i \cdot v(\hat{\boldsymbol{y}}_i) \quad (3)$$

where the weights are normalized to 1 and inverse proportional to how far i is from m,

$$\alpha_i \propto \exp\big(-\gamma \cdot \omega(m-1-i)\big) \quad (4)$$

γ is a hyper-parameter (we set to $1/2$) and $\omega(\cdot)$ is a monotonically nondecreasing function and we simply choose the identity function.

Note that, we summarize over representations of "micro-instructions" ($\boldsymbol{x}_m$) and experiences of executing those micro-instructions $\hat{\boldsymbol{y}}_m$. The two encoders $u(\cdot)$ and $v(\cdot)$ are described in § 4.4. They are essentially the summaries of "low-level" details, *i.e.*, representations of a sequence of words, or a sequence of states and actions. While existing work often directly summarizes all the low-level details, we have found that the current form of "hierarchical" summarizing (*i.e.*, first summarizing each Baby-Step, then summarizing all previous Baby-Steps) performs better.

Policy The agent takes actions, conditioning on the context $\hat{\boldsymbol{z}}_m$, and the current instruction $\boldsymbol{x}_m$:

$$\hat{\boldsymbol{a}}_t \sim \boldsymbol{\pi}\left(\cdot|\boldsymbol{s}_t, \hat{\boldsymbol{a}}_{t-1}; u(\boldsymbol{x}_m), \hat{\boldsymbol{z}}_m\right) \quad (5)$$

where the policy is implemented with a LSTM with the same cross-modal attention between visual states and languages as in (Fried et al., 2018).

4.2 Learning of the BabyWalk Agent

The agent learns in two phases. In the first one, imitation learning is used where the agent learns to execute Baby-Steps accurately. In the second one, the agent learns to execute successively longer tasks from a designed curriculum.

4.2.1 Imitation Learning

Baby-Steps are shorter navigation tasks. With the mth instruction $\boldsymbol{x}_m$, the agent is asked to follow the instruction so that its trajectory matches the human expert's $\boldsymbol{y}_m$. To assist the learning, the context is computed from the human expert trajectory up to the mth Baby-Step (*i.e.*, in eq. (1), $\hat{\boldsymbol{y}}$s are replaced with $\boldsymbol{y}$s). We maximize the objective

$$\ell = \sum_{m=1}^{\text{M}} \sum_{t_m=1}^{|\boldsymbol{y}_m|} \log \boldsymbol{\pi}\left(\boldsymbol{a}_{t_m}|\boldsymbol{s}_{t_m}, \boldsymbol{a}_{t_m-1}; u(\boldsymbol{x}_m), \boldsymbol{z}_m\right)$$

We emphasize here each Baby-Step is treated independently of the others in this learning regime. Each time a Baby-Step is to be executed, we "preset" the agent in the human expert's context and the last visited state. We follow existing literature (Anderson et al., 2018; Fried et al., 2018) and use student-forcing based imitation learning, which uses agent's predicted action instead of the expert action for the trajectory rollout.

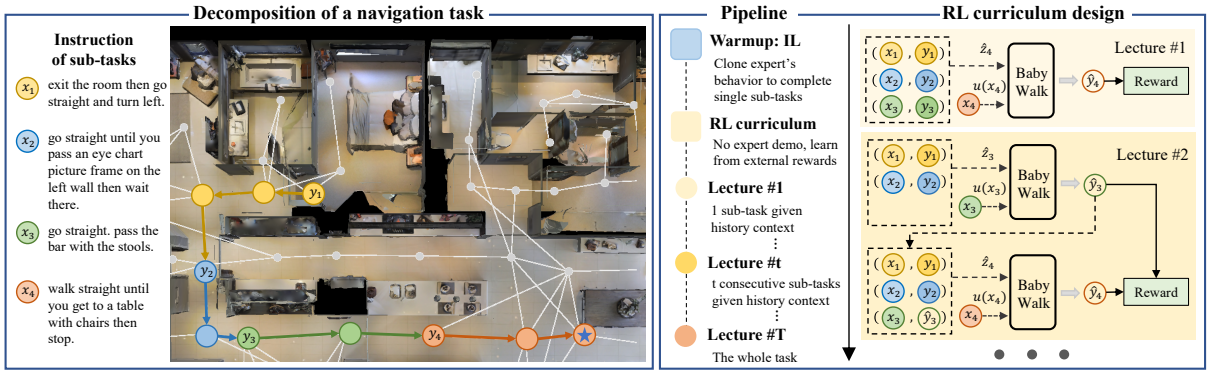

Figure 3: **Two-phase learning by BABYWALK.** (Left) An example instruction-trajectory pair from the R4R dataset is shown. The long instruction is segmented into four BABY-STEP instructions. We use those BABY-STEPs for imitation learning (§ 4.2.1) (Right) Curriculum-based RL. The BABYWALK agent warm-starts from the imitation learning policy, and incrementally learns to handle longer tasks by executing consecutive BABY-STEPs and getting feedback from external rewards (*c.f.* § 4.2.2). We illustrate two initial RL lectures using the left example.

4.2.2 Curriculum Reinforcement Learning

We want the agent to be able to execute multiple consecutive BABY-STEPs and optimize its performance on following longer navigation instructions (instead of the cross-entropy losses from the imitation learning). However, there is a discrepancy between our goal of training the agent to cope with the uncertainty in a long instruction and the imitation learning agent's ability in accomplishing shorter tasks *given the human annotated history*. Thus it is challenging to directly optimize the agent with a typical RL learning procedure, even the imitation learning might have provided a good initialization for the policy, see our ablation study in § 5.3.

Inspired by the curriculum learning strategy (Bengio et al., 2009), we design an incremental learning process that the agent is presented with a curriculum of increasingly longer navigation tasks. Fig. 3 illustrates this idea with two "lectures". Given a long navigation instruction X with M BABY-STEPs, for the kth lecture, the agent is given all the human expert's trajectory up to but not including the $(\mathrm{M}-k+1)$th BABY-STEP, as well as the history context $\boldsymbol{z}_{\mathrm{M}-k+1}$. The agent is then asked to execute the kth micro-instructions from $\boldsymbol{x}_{\mathrm{M}-k+1}$ to $\boldsymbol{x}_{\mathrm{M}}$ using reinforcement learning to produce its trajectory that optimizes a task related metric, for instance the fidelity metric measuring how faithful the agent follows the instructions.

As we increase k from 1 to M, the agent faces the challenge of navigating longer and longer tasks with reinforcement learning. However, the agent only needs to improve its skills from its prior exposure to shorter ones. Our ablation studies show this is indeed a highly effective strategy.

	R2R	R4R	R6R	R8R
Train seen instr.	14,039	233,532	89,632	94,731
Val unseen instr.	2,349	45,234	35,777	43,273
Avg instr. length	29.4	58.4	91.2	121.6
Avg # BABY-STEPs	1.8	3.6	5.6	7.4

Table 1: Datasets used for VLN learning and evaluation

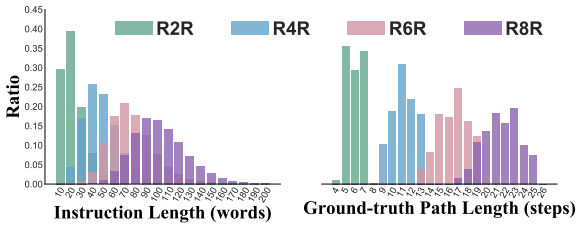

Figure 4: The distribution of lengths of instructions and ground-truth trajectories in our datasets.

4.3 New Datasets for Evaluation & Learning

To our best knowledge, this is the first work studying how well VLN agents generalize to long navigation tasks. To this end, we create the following datasets in the same style as in (Jain et al., 2019).

ROOM6ROOM and ROOM8ROOM We concatenate the trajectories in the training as well as the validation unseen split of the ROOM2ROOM dataset for 3 times and 4 times respectively, thus extending the lengths of navigation tasks to 6 rooms and 8 rooms. To join, the end of the former trajec-

tory must be within 0.5 meter with the beginning of the later trajectory. Table 1 and Fig. 4 contrast the different datasets in the # of instructions, the average length (in words) of instructions and how the distributions vary.

Table 1 summarizes the descriptive statistics of BABY-STEPS across all datasets used in this paper. The datasets and the segmentation/alignments are made publically available[2].

4.4 Key Implementation Details

In the following, we describe key information for research reproducibility, while the complete details are in the Suppl. Material.

States and Actions We follow (Fried et al., 2018) to set up the states as the visual features (*i.e*. ResNet-152 features (He et al., 2016)) from the agent-centric panoramic views in 12 headings $\times$ 3 elevations with 30 degree intervals. Likewise, we use the same panoramic action space.

Identifying BABY-STEPS Our learning approach requires an agent to follow micro-instructions (*i.e*., the BABY-STEPS). Existing datasets (Anderson et al., 2018; Jain et al., 2019; Chen et al., 2019) do not provide fine-grained segmentations of long instructions. Therefore, we use a template matching approach to aggregate consecutive sentences into BABY-STEPS. First, we extract the noun phrase using POS tagging. Then, we employs heuristic rules to chunk a long instruction into shorter segments according to punctuation and landmark phrase (*i.e*., words for concrete objects). We document the details in the Suppl. Material.

Aligning BABY-STEPS with Expert Trajectory *Without extra annotation*, we propose a method to approximately chunk original expert trajectories into sub-trajectories that align with the BABY-STEPS. This is important for imitation learning at the micro-instruction level (§ 4.2.1). Specifically, we learn a multi-label visual landmark classifier to identify concrete objects from the states along expert trajectories by using the landmark phrases extracted from the their instructions as weak supervision. For each trajectory-instruction pair, we then extract the visual landmarks of every state as well as the landmark phrases in BABY-STEP instructions. Next, we perform a dynamic programming procedure to segment the expert trajectories by aligning the visual landmarks and landmark phrases, using the confidence scores of the multi-label visual landmark classifier to form the function.

Encoders and Embeddings The encoder $u(\cdot)$ for the (micro)instructions is a LSTM. The encoder for the trajectory $\boldsymbol{y}$ contains two separate Bi-LSTMs, one for the state $\boldsymbol{s}_t$ and the other for the action $\boldsymbol{a}_t$. The outputs of the two Bi-LSTMs are then concatenated to form the embedding function $v(\cdot)$. The details of the neural network architectures (*i.e*. configurations as well as an illustrative figure), optimization hyper-parameters, *etc*. are included in the Suppl. Material.

Learning Policy with Reinforcement Learning In the second phase of learning, BABYWALK uses RL to learn a policy that maximizes the *fidelity-oriented* rewards (CLS) proposed by Jain et al. (2019). We use policy gradient as the optimizer (Sutton et al., 2000). Meanwhile, we set the maximum number of lectures in curriculum RL to be 4, which is studied in Section 5.3.

5 Experiments

We describe the experimental setup (§ 5.1),followed by the main results in § 5.2 where we show the proposed BABYWALK agent attains competitive results on both the in-domain dataset but also generalizing to out-of-the-domain datasets with varying lengths of navigation tasks. We report results from various ablation studies in § 5.3. While we primarily focus on the ROOM4ROOM dataset, we re-analyze the original ROOM2ROOM dataset in § 5.4 and were surprised to find out the agents trained on it can generalize.

5.1 Experimental Setups.

Datasets We conduct empirical studies on the existing datasets ROOM2ROOM and ROOM4ROOM (Anderson et al., 2018; Jain et al., 2019), and the two newly created benchmark datasets ROOM6ROOM and ROOM8ROOM, described in § 4.3. Table 1 and Fig. 4 contrast their differences.

Evaluation Metrics We adopt the following metrics: *Success Rate (*SR*)* that measures the average rate of the agent stopping within a specified distance near the goal location (Anderson et al., 2018), *Coverage weighted by Length Score (*CLS*)* (Jain et al., 2019) that measures the fidelity of the agent's path to the reference, weighted by the length score,

[2]Available at https://github.com/Sha-Lab/babywalk

Setting	In-domain R4R → R4R			Generalization to other datasets R4R → R2R			R4R → R6R			R4R → R8R			Average		
Metrics	SR↑	CLS↑	SDTW↑	SR↑	CLS↑	SDTW↑	SR↑	CLS↑	SDTW↑	SR↑	CLS↑	SDTW↑	SR↑	CLS↑	SDTW↑
SEQ2SEQ	25.7	20.7	9.0	16.3	27.1	10.6	14.4	17.7	4.6	20.7	15.0	4.7	17.1	19.9	6.6
SF^{+}	24.9	23.6	9.2	22.5	29.5	14.8	15.5	20.4	5.2	21.6	17.2	5.0	19.9	22.4	8.3
RCM(GOAL)$^{+}$	28.7	36.3	13.2	25.9	44.2	20.2	19.3	31.8	7.3	22.8	27.6	5.1	22.7	34.5	10.9
RCM(FIDELITY)$^{+}$	24.7	39.2	13.7	29.1	34.3	18.3	20.5	38.3	7.9	20.9	34.6	6.1	23.5	35.7	10.8
REGRETFUL$^{+\star}$	30.1	34.1	13.5	22.8	32.6	13.4	18.0	31.7	7.5	18.7	29.3	5.6	19.8	31.2	8.8
FAST$^{+\star}$	**36.2**	34.0	15.5	25.1	33.9	14.2	22.1	31.5	7.7	**27.7**	29.6	6.3	25.0	31.7	9.4
BABYWALK	29.6	47.8	**18.1**	**35.2**	48.5	27.2	**26.4**	44.9	13.1	26.3	44.7	**11.5**	**29.3**	46.0	17.3
BABYWALK $^{+}$	27.3	**49.4**	17.3	34.1	**50.4**	**27.8**	25.5	**47.2**	**13.6**	23.1	**46.0**	11.1	27.6	**47.9**	**17.5**

Table 2: VLN agents trained on the R4R dataset and evaluated on the unseen portion of the R4R (in-domain) and the other 3 out-of-the-domain datasets: R2R, R6R and R8R with different distributions in instruction length. The Suppl. Material has more comparisons. ($^{+}$: pre-trained with data augmentation. *: reimplemented or adapted from the original authors' public codes).

and the newly proposed *Success rate weighted normalized Dynamic Time Warping (*SDTW*)* that measures in more fine-grained details, the spatio-temporal similarity of the paths by the agent and the human expert, weighted by the success rate (Magalhaes et al., 2019). Both CLS and SDTW measure explicitly the agent's ability to follow instructions and in particular, it was shown that SDTW corresponds to human preferences the most. We report results in other metrics in the Suppl. Material.

Agents to Compare to Whenever possible, for all agents we compare to, we either re-run, reimplement or adapt publicly available codes from their corresponding authors with their provided instructions to ensure a fair comparison. We also "sanity check" by ensuring the results from our implementation and adaptation replicate and are comparable to the reported ones in the literature.

We compare our BABYWALK to the following: (1) the SEQ2SEQ agent (Anderson et al., 2018), being adapted to the panoramic state and action space used in this work; (2) the Speaker Follower (SF) agent (Fried et al., 2018); (3) the Reinforced Cross-Modal Agent (RCM) (Wang et al., 2019) that refines the SF agent using reinforcement learning with either *goal-oriented reward* (RCM(GOAL)) or *fidelity-oriented reward* (RCM(FIDELITY)); (4) the Regretful Agent (REGRETFUL) (Ma et al., 2019b) that uses a progress monitor that records visited path and a regret module that performs backtracking; (5) the Frontier Aware Search with Backtracking agent (FAST) (Ke et al., 2019) that incorporates global and local knowledge to compare partial trajectories in different lengths.

The last 3 agents are reported having state-of-the art results on the benchmark datasets. Except the SEQ2SEQ agent, all other agents depend on an additional pre-training stage with data augmentation (Fried et al., 2018), which improves cross-board. Thus, we train two BABYWALK agents: one with and the other without the data augmentation.

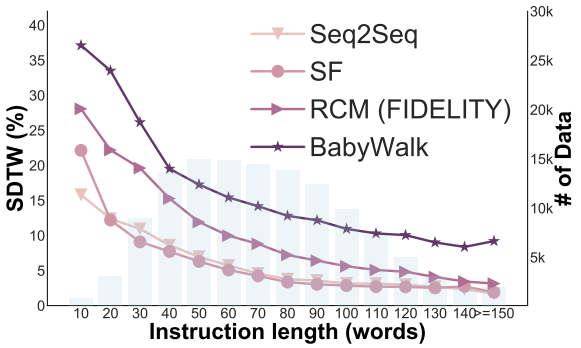

Figure 5: Performance by various agents on navigation tasks in different lengths. See texts for details.

5.2 Main results

In-domain Generalization This is the standard evaluation scenario where a trained agent is assessed on the unseen split from the same dataset as the training data. The leftmost columns in Table 2 reports the results where the training data is from R4R. The BABYWALK agents outperform all other agents when evaluated on CLS and SDTW.

When evaluated on SR, FAST performs the best and the BABYWALK agents do not stand out. This is expected: agents which are trained to reach goal do not necessarily lead to better instruction-following. Note that RCM(FIDELITY) performs well in path-following.

Setting	R4R → R4R			R4R → others		
Metrics	SR↑	CLS↑	SDTW ↑	SR↑	CLS↑	SDTW ↑
$f_{\text{SUMMARY}} =$						
NULL	18.9	43.1	9.9	17.1	42.3	9.6
LSTM$(\cdot)$	25.8	44.0	14.4	25.7	42.1	14.3
$f_{\text{SUMMARY}} = \sum_{i=1}^{m-1} \alpha_i \cdot (\cdot)$, *i.e.*, eqs. (2,3)						
$\gamma = 5$	27.5	46.8	15.8	26.7	44.4	14.9
$\gamma = 0.5$	27.3	**49.4**	**17.3**	**27.6**	**47.9**	**17.5**
$\gamma = 0.05$	**27.5**	47.7	16.2	26.0	45.5	15.2
$\gamma = 0$	26.1	46.6	15.1	25.1	44.3	14.4

Table 3: The memory buffer is beneficial to generalizing to different tasks from on which the agent is trained.

Out-of-domain Generalization While our primary goal is to train agents to generalize well to longer navigation tasks, we are also curious how the agents perform on shorter navigation tasks too. The right columns in Table 2 report the comparison. The BABYWALK agents outperform all other agents in all metrics except SR. In particular, on SDTW, the generalization to R6R and R8R is especially encouraging, resulting almost twice those of the second-best agent FAST. Moreover, recalling from Fig. 1, BABYWALK's generalization to R6R and R8R attain even better performance than the RCM agents that are trained *in-domain*.

Fig. 5 provides additional evidence on the success of BABYWALK, where we have contrasted to its performance to other agents' on following instructions in different lengths *across all datasets*. Clearly, the BABYWALK agent is able to improve very noticeably on longer instructions.

Qualitative Results Fig. 6 contrasts visually several agents in executing two (long) navigation tasks. BABYWALK's trajectories are similar to what human experts provide, while other agents' are not.

5.3 Analysis

Memory Buffer is Beneficial Table 3 illustrates the importance of having a memory buffer to summarize the agent's past experiences. Without the memory (NULL), generalization to longer tasks is significantly worse. Using LSTM to summarize is worse than using forgetting to summarize (eqs. (2,3)). Meanwhile, ablating γ of the forgetting mechanism concludes that $\gamma = 0.5$ is the optimal to our hyperparameter search. Note that when $\gamma = 0$, this mechanism degenerates to taking average of the memory buffer, and leads to inferior results.

Setting	R4R → R4R			R4R → others		
Metrics	SR↑	CLS↑	SDTW ↑	SR↑	CLS↑	SDTW ↑
IL	24.7	27.9	11.1	24.2	25.8	10.2
IL+RL	25.0	45.5	13.6	25.0	43.8	14.1
IL+ CRL w/ LECTURE #						
1*st*	24.1	44.8	13.5	24.1	43.1	13.6
2*nd*	26.7	45.9	15.2	26.2	43.7	14.8
3*rd*	**27.9**	47.4	17.0	26.7	45.4	16.3
4*th*	27.3	**49.4**	**17.3**	**27.6**	**47.9**	**17.5**

Table 4: BABYWALK's performances with curriculum-based reinforcement learning (CRL), which improves imitation learning without or with reinforcement learning (IL+RL).

Eval	→ R6R			→ R8R		
Training	SR↑	CLS↑	SDTW↑	SR↑	CLS↑	SDTW↑
R2R	21.7	**49.0**	11.2	20.7	**48.7**	9.8
R4R	**25.5**	47.2	**13.6**	**23.1**	46.0	**11.1**
Eval	→ R2R			→ R4R		
Training	SR↑	CLS↑	SDTW↑	SR↑	CLS↑	SDTW↑
R2R	**43.8**	**54.4**	**36.9**	21.4	**51.0**	13.8
R4R	34.1	50.4	27.8	**27.3**	49.4	**17.3**

Table 5: (Top) BABYWALK trained on R2R is nearly as effective as the agent trained on R4R when generalizing to longer tasks. (Bottom) BABYWALK trained on R2R adapts to R4R better than the agent trained in the reverse direction.

Curriculum-based RL (CRL) is Important Table 4 establishes the value of CRL. While imitation learning (IL) provides a good warm-up for SR, significant improvement on other two metrics come from the subsequent RL (IL+RL). Furthermore, CRL (with 4 "lectures") provides clear improvements over direct RL on the entire instruction (*i.e.*, learning to execute all BABY-STEPS at once). Each lecture improves over the previous one, especially in terms of the SDTW metric.

5.4 Revisiting ROOM2ROOM

Our experimental study has been focusing on using R4R as the training dataset as it was established that as opposed to R2R, R4R distinguishes well an agent who just learns to reach the goal from an agent who learns to follow instructions.

Given the encouraging results of generalizing to longer tasks, a natural question to ask, *how well can an agent trained on* R2R *generalize?*

Results in Table 5 are interesting. Shown in the top panel, the difference in the averaged performance of generalizing to R6R and R8R is not

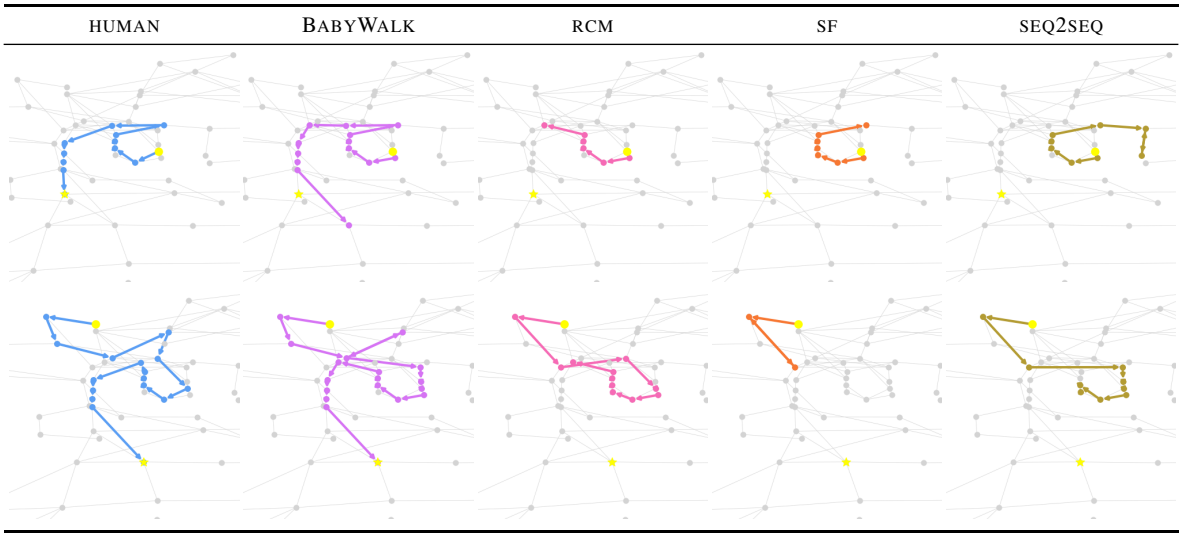

Figure 6: Trajectories by human experts and VLN agents on two navigation tasks. More are in the Suppl. Material.

significant. The agent trained on R4R has a small win on R6R presumably because R4R is closer to R6R than R2R does. But for even longer tasks in R8R, the win is similar.

In the bottom panel, however, it seems that R2R → R4R is stronger (incurring less loss in performance when compared to the in-domain setting R4R → R4R) than the reverse direction (*i.e.*, comparing R4R → R2R to the in-domain R2R → R2R). This might have been caused by the noisier segmentation of long instructions into BABY-STEPS in R4R. (While R4R is composed of two navigation paths in R2R, the segmentation algorithm is not aware of the "natural" boundaries between the two paths.)

6 Discussion

There are a few future directions to pursue. First, despite the significant improvement, the gap between short and long tasks is still large and needs to be further reduced. Secondly, richer and more complicated variations between the learning setting and the real physical world need to be tackled. For instance, developing agents that are robust to variations in both visual appearance and instruction descriptions is an important next step.

Acknowledgments We appreciate the feedback from the reviewers. This work is partially supported by NSF Awards IIS-1513966/1632803/1833137, CCF-1139148, DARPA Award#: FA8750-18-2-0117, DARPA-D3M - Award UCB-00009528, Google Research Awards, gifts from Facebook and Netflix, and ARO# W911NF-12-1-0241 and W911NF-15-1-0484.

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 *CVPR*.

Jacob Andreas, Dan Klein, and Sergey Levine. 2017. Modular multitask reinforcement learning with policy sketches. In *ICML*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. 2017. Matterport3D: Learning from RGB-D data in indoor environments. In *3DV*.

David L Chen and Raymond J Mooney. 2011. Learning to interpret natural language navigation instructions from observations. In *AAAI*.

Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. 2019. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *CVPR*.

Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. 2017. Reverse curriculum generation for reinforcement learning. In *CoRL*.

Daniel Fried, Ronghang Hu, Volkan Cirik, Anna Rohrbach, Jacob Andreas, Louis-Philippe Morency, Taylor Berg-Kirkpatrick, Kate Saenko, Dan Klein, and Trevor Darrell. 2018. Speaker-follower models for vision-and-language navigation. In *NeurIPS*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Hexiang Hu, Liyu Chen, Boqing Gong, and Fei Sha. 2018. Synthesized policies for transfer and adaptation across tasks and environments. In *NeurIPS*.

Haoshuo Huang, Vihan Jain, Harsh Mehta, Alexander Ku, Gabriel Magalhaes, Jason Baldridge, and Eugene Ie. 2019. Transferable representation learning in vision-and-language navigation. In *ICCV*.

Vihan Jain, Gabriel Magalhaes, Alex Ku, Ashish Vaswani, Eugene Ie, and Jason Baldridge. 2019. Stay on the path: Instruction fidelity in vision-and-language navigation. In *EMNLP*.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. 2018. Progressive growing of gans for improved quality, stability, and variation. In *ICLR*.

Liyiming Ke, Xiujun Li, Yonatan Bisk, Ari Holtzman, Zhe Gan, Jingjing Liu, Jianfeng Gao, Yejin Choi, and Siddhartha Srinivasa. 2019. Tactical rewind: Self-correction via backtracking in vision-and-language navigation. In *CVPR*.

Joohyun Kim and Raymond Mooney. 2013. Adapting discriminative reranking to grounded language learning. In *ACL*.

Chih-Yao Ma, Jiasen Lu, Zuxuan Wu, Ghassan AlRegib, Zsolt Kira, Richard Socher, and Caiming Xiong. 2019a. Self-monitoring navigation agent via auxiliary progress estimation. In *ICLR*.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *CVPR*.

Gabriel Magalhaes, Vihan Jain, Alexander Ku, Eugene Ie, and Jason Baldridge. 2019. Effective and general evaluation for instruction conditioned navigation using dynamic time warping. In *NeurIPS ViGIL Workshop*.

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 *ICLR*.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*.

Khanh Nguyen and Hal Daumé III. 2019. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. In *EMNLP*.

Junhyuk Oh, Satinder Singh, Honglak Lee, and Pushmeet Kohli. 2017. Zero-shot task generalization with multi-task deep reinforcement learning. In *ICML*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Sungryull Sohn, Junhyuk Oh, and Honglak Lee. 2018. Hierarchical reinforcement learning for zero-shot generalization with subtask dependencies. In *NeurIPS*.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NeurIPS*.

Hao Tan, Licheng Yu, and Mohit Bansal. 2019. Learning to navigate unseen environments: Back translation with environmental dropout. In *EMNLP*.

Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. 2019. Vision-and-dialog navigation. In *CoRL*.

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*.

Yuke Zhu, Daniel Gordon, Eric Kolve, Dieter Fox, Li Fei-Fei, Abhinav Gupta, Roozbeh Mottaghi, and Ali Farhadi. 2017a. Visual semantic planning using deep successor representations. In *ICCV*.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017b. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *ICRA*.

Supplementary Material

In this supplementary material, we provide details omitted in the main text. The content is organized as what follows:

- Section A. Details on identifying BABY-STEP instructions and aligning BABY-STEPs with expert trajectories. (§ 4.3 and § 4.4 of the main text)
- Section B. Implementation details of the navigation agent, reward function used in RL and optimization hyper-parameters. (§ 4.4 of the main text)
- Section C. Additional experimental results, including in-domain & transfer results of different dataset trained models, sanity check of our reimplementation, and extra analysis of BABYWALK. (§ 5.1 and § 5.2 of the main text)

A Details on BABY-STEP Identification and Trajectory Alignments

In this section, we describe the details of how BABY-STEPs are identified in the annotated natural language instructions and how expert trajectory data are segmented to align with BABY-STEP instructions.

A.1 Identify BABY-STEPs

We identify the navigable BABY-STEPs from the natural language instructions of R2R, R4R, R6R and R8R, based on the following 6 steps:

1. **Split sentence and chunk phrases.** We split the instructions by periods. For each sentence, we perform POS tagging using the SpaCy (Honnibal and Montani, 2017) package to locate and chunk all plausible noun phrases and verb phrases.
2. **Curate noun phrases.** We curate noun phrases by removing the stop words (*i.e.*, the, for, from *etc.*) and isolated punctuations among them and lemmatizing each word of them. The purpose is to collect a concentrated set of semantic noun phrases that contain potential visual objects.
3. **Identify "landmark words".** Next, given the set of candidate visual object words, we filter out a blacklist of words that either do not correspond to any visual counterpart or are misclassified by the SpaCy package. The word blacklist includes:

   `end, 18 inch, head, inside, forward, position, ground, home, face, walk, feet, way, walking, bit, veer, 've, next, stop, towards, right, direction, thing, facing, side, turn, middle, one, out, piece, left, destination, straight, enter, wait, don't, stand, back, round`

   We use the remaining noun phrases as the "landmark words" of the sentences. Note that this step identifies the "landmark words" for the later procedure which aligns BABY-STEPs and expert trajectories.
4. **Identifying verb phrases.** Similarly, we use a verb blacklist to filter out verbs that require no navigational actions of the agent. The blacklist includes: `make, turn, face, facing, veer`.
5. **Merge non-actionable sentences.** We merge the sentence without landmarks and verbs into the next sentence, as it is likely not actionable.
6. **Merge stop sentences.** There are sentences that only describe the stop condition of a navigation action, which include verb-noun compositions indicating the stop condition. We detect the sentences starting with `wait, stop, there, remain, you will see` as the sentences that only describe the stop condition and merge them to the previous sentence. Similarly, we detect sentences starting with `with, facing` and merge them to the next sentence.

After applying the above 6 heuristic rules to the language instruction, we obtain chunks of sentences that describes the navigable BABY-STEPs of the whole task (*i.e.*, a sequence of navigational sub-goals.).

A.2 Align Expert Trajectories with identified BABY-STEPs

In the previous section, we describe the algorithm for identifying BABY-STEP instructions from the original natural language instructions of the dataset. Now we are going to describe the procedure of aligning BABY-STEPs with the expert trajectories, which segments the expert trajectories according to the BABY-STEPs to create the training data for the learning pipeline of our BABYWALK agent. Note

that during the training, our BABYWALK *does not rely on the existence of ground-truth alignments* between the (micro)instructions and BABY-STEPS trajectories.

Main Idea The main idea here is to: 1) perform visual landmark classification to produce confidence scores of landmarks for each visual state $\boldsymbol{s}$ along expert trajectories; 2) use the predicted landmark scores and the "landmark words" in BABY-STEPS to guide the alignment between the expert trajectory and BABY-STEPS. To achieve this, we train a visual landmark classifier with weak supervision — trajectory-wise existence of landmark objects. Next, based on the predicted landmark confidence scores, we use dynamic programming (DP) to chunk the expert trajectory into segments and assign the segments to the BABY-STEPS.

Weakly Supervised Learning of the Landmark Classifier Given the pairs of aligned instruction and trajectories (X, Y) from the original dataset, we train a landmark classifier to detect landmarks mentioned in the instructions. We formulate it as a multi-label classification problem that asks a classifier $f_{\text{LDMK}}(\boldsymbol{s}_t; \mathcal{O})$ to predict all the landmarks $\mathcal{O}_{\mathrm{X}}$ of the instruction X given the corresponding trajectory Y. Here, we denotes all possible landmarks from the entire dataset to be $\mathcal{O}$, and the landmarks of a specific instruction X to be $\mathcal{O}_{\mathrm{X}}$. Concretely, we first train a convolutional neural network (CNN) based on the visual state features $\boldsymbol{s}_t$ to independently predict the existence of landmarks at every time step, then we aggregate the predictions across all time steps to get trajectory-wise logits $\boldsymbol{\psi}$ via max-pooling over all states of the trajectory.

$$\boldsymbol{\psi} = \texttt{max}\left\{f_{\text{LDMK}}(\boldsymbol{s}_t; \mathcal{O}) \mid t = 1, \ldots, |\mathrm{Y}|\right\}$$

Here f_{LDMK} denotes the independent state-wise landmark classifier, and $\boldsymbol{\psi}$ is the logits before normalization for computing the landmark probability. For the specific details of f_{LDMK}, we input the 6×6 panorama visual feature (*i.e.* ResNet-152 feature) into a two-layer CNN (with kernel size of 3, hidden dimension of 128 and ReLU as non-linearity layer) to produce feature activation with spatial extents, followed by a global averaging operator over spatial dimensions and a multi-layer perceptron (2-layer with hidden dimension of 512 and ReLU as non-linearity layer) that outputs the state-wise logits for all visual landmarks $\mathcal{O}$. We then max pool all the state-wise logits along the trajectory and compute the loss using a trajectory-wise binary cross-entropy between the ground-truth landmark label (of existence) and the prediction.

Aligning BABY-STEPS and Trajectories with Visual Landmarks Now, sppose we have a sequence of BABY-STEP instructions $\mathrm{X} = \{\boldsymbol{x}_m,\ m = 1, \ldots, \mathrm{M}\}$, and its expert trajectory $\mathrm{Y} = \{\boldsymbol{s}_t,\ t = 1, \ldots, |\mathrm{Y}|\}$, we can compute the averaged landmark score for the landmarks $\mathcal{O}_{\boldsymbol{x}_m}$ that exists in this sub-task instruction $\boldsymbol{x}_m$ on a single state $\boldsymbol{s}_t$:

$$\Psi(t, m) = \frac{\mathbf{1}\left[\boldsymbol{o}_m \in \mathcal{O}_{\boldsymbol{x}_m}\right]^\top f_{\text{LDMK}}(\boldsymbol{s}_t; \mathcal{O})}{|\mathcal{O}_{\boldsymbol{x}_m}|}$$

Here $\mathbf{1}\left[\boldsymbol{o}_m \in \mathcal{O}\right]$ represents the one-hot encoding of the landmarks that exists in the BABY-STEP $\boldsymbol{x}_m$, and $|\mathcal{O}_{\boldsymbol{x}_m}|$ is the total number of existed landmarks. We then apply dynamic programming (DP) to solve the trajectory segmentation specified by the following Bellman equation (in a recursive form).

$$\Phi(t, m) = \begin{cases} \Psi(t, m), & \text{if } t = 1 \\ \Psi(t, m) + \max\limits_{i \in \{1, \ldots, t-1\}} \left\{\Phi(i,\ m-1)\right\}, & \text{otherwise} \end{cases}$$

Here, $\Phi(t, m)$ represents the maximum potential of choosing the state $\boldsymbol{s}_t$ as the end point of the BABY-STEP instruction $\boldsymbol{x}_m$. Solving this DP leads to a set of correspondingly segmented trajectories $\mathrm{Y} = \{\boldsymbol{y}_m,\ m = 1, \ldots, \mathrm{M}\}$, with $\boldsymbol{y}_m$ being the m-th BABY-STEP sub-trajectory.

B Implementation details

B.1 Navigation Agent Configurations

Figure 7 gives an overview of the unrolled version of our full navigation agent.

Panoramic State-Action Space (Fried et al., 2018) We set up the states $\boldsymbol{s}_t$ as the stacked visual feature of agent-centric panoramic views in 12 headings $\times$ 3 elevations with 30 degree intervals. The visual feature of each view is a concatenation of the ResNet-152 feature vector of size 2048 and the orientation feature vector of size 128 (The 4-dimensional orientation feature $[\sin(\phi); \cos(\phi); \sin(\omega); \cos(\omega)]$ are tiled 32 times). We use similar single-view visual feature of size 2176 as our action embeddings.

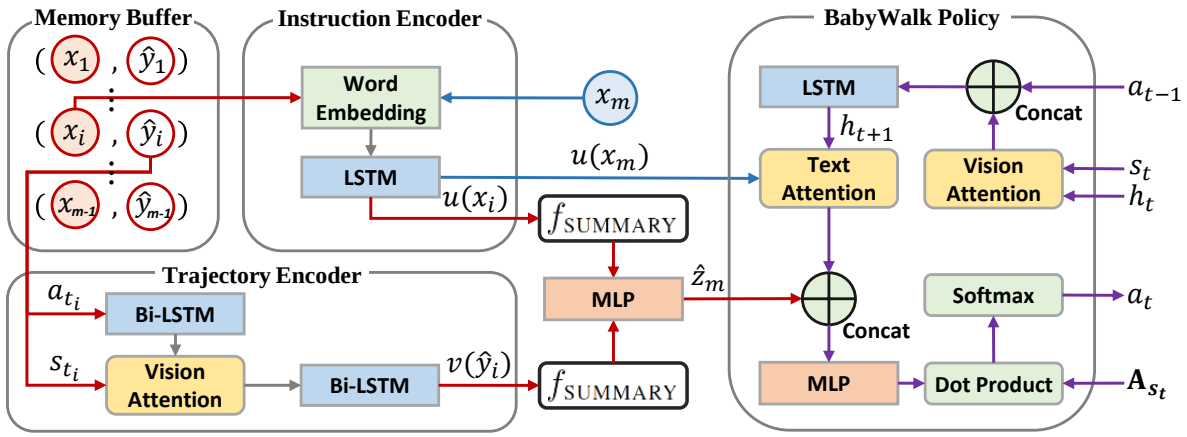

Figure 7: Our network architecture at the m-*th* BABY-STEP sub-task. **Red line** represents the procedure of encoding context variable $\boldsymbol{z}_m$ via summarizing the BABY-STEP trajectory $f_{\text{SUMMARY}}(v(\hat{\boldsymbol{y}}_1), \ldots, v(\hat{\boldsymbol{y}}_{m-1}))$ and the corresponding (micro)instruction $f_{\text{SUMMARY}}(u(\boldsymbol{x}_1), \ldots, u(\boldsymbol{x}_{m-1}))$ in the memory buffer. **Blue line** represents the procedure of encoding the (micro)instruction $u(\boldsymbol{x}_m)$ of the current BABY-STEP. **Purple line** represents the detailed decision making process of our BABYWALK policy ($\mathbf{A}_{\boldsymbol{s}_t}$ is denoted as the set of navigable directions at $\boldsymbol{s}_t$ as defined by Fried et al. (2018))

Encoders Instruction encoder $u(\cdot)$ for the instructions is a single directional LSTM with hidden size 512 and a word embedding layer of size 300 (initialized with GloVE embedding (Pennington et al., 2014)). We use the same encoder for encoding the past experienced and the current executing instruction. Trajectory encoder $v(\cdot)$ contains two separate bidirectional LSTMs (Bi-LSTM), both with hidden size 512. The first Bi-LSTM encodes $\boldsymbol{a}_{t_i}$ and outputs a hidden state for each time step t_i. Then we attends the hidden state to the panoramic view $\boldsymbol{s}_{t_i}$ to get a state feature of size 2176 for each time step. The second Bi-LSTM encoders the state feature. We use the trajectory encoder just for encoding the past experienced trajectories.

BABYWALK Policy The BABYWALK policy network consists of one LSTM with two attention layers and an action predictor. First we attend the hidden state to the panoramic view $\boldsymbol{s}_t$ to get state feature of size 2176. The state feature is concatenated with the previous action embedding as a variable to update the hidden state using a LSTM with hidden size 512. The updated hidden state is then attended to the context variables (output of $u(\cdot)$). For the action predictor module, we concatenate the output of text attention layer with the summarized past context $\hat{\boldsymbol{z}}_m$ in order to get an action prediction variable. We then get the action prediction variable through a 2-layer MLP and make a dot product with the navigable action embeddings to retrieve the probability of the next action.

Model Inference During the inference time, the BABYWALK policy only requires running the heuristic BABY-STEP identification on the test-time instruction. No need for oracle BABY-STEP trajectory during this time as the BABYWALK agent is going to roll out for each BABY-STEP by itself.

B.2 Details of Reward Shaping for RL

As mentioned in the main text, we learn policy via optimizing the *Fidelity-oriented reward* (Jain et al., 2019). Now we give the complete details of this reward function. Suppose the total number of roll out steps is $\mathrm{T} = \sum_{i=1}^{\mathrm{M}} |\hat{\boldsymbol{y}}_i|$, we would have the following form of reward function:

$$r(\boldsymbol{s}_t, \boldsymbol{a}_t) = \begin{cases} 0, & \text{if } t < \mathrm{T} \\ \text{SR}(\mathrm{Y}, \hat{\mathrm{Y}}) + \text{CLS}(\mathrm{Y}, \hat{\mathrm{Y}}), & \text{if } t = \mathrm{T} \end{cases}$$

Here, $\hat{\mathrm{Y}} = \hat{\boldsymbol{y}}_1 \oplus \ldots \oplus \hat{\boldsymbol{y}}_{\mathrm{M}}$ represents the concatenation of BABY-STEP trajectories produced by the navigation agent (and we note $\oplus$ as the concatenation operation).

B.3 Optimization Hyper-parameters

For each BABY-STEP task, we set the maximal number of steps to be 10, and truncate the corresponding BABY-STEP instruction length to be 100. During both the imitation learning and the curriculum reinforcement learning procedures, we fix the learning rate to be 1e-4. In the imitation

learning, the mini-batch size is set to be 100. In the curriculum learning, we reduce the mini-batch size as curriculum increases to save memory consumption. For the 1*st*, 2*nd*, 3*rd* and 4*th* curriculum, the mini-batch size is set to be 50, 32, 20, and 20 respectively. During the learning, we pre-train our BABYWALK model for 50000 iterations using the imitation learning as a warm-up stage. Next, in each lecture (up to 4) of the reinforcement learning (RL), we train the BABYWALK agent for an additional 10000 iterations, and select the best performing model in terms of SDTW to resume the next lecture. For executing each instruction during the RL, we sample 8 navigation episodes before performing any back-propagation. For each learning stage, we use separate Adam optimizers to optimize for all the parameters. Meanwhile, we use the L2 weight decay as the regularizer with its coefficient set to be 0.0005. In the reinforcement learning, the discounted factor γ is set to be 0.95.

C Additional Experimental Results

In this section, we describe a comprehensive set of evaluation metrics and then show transfer results of models trained on each dataset, with all metrics. We provide additional analysis studying the effectiveness of template based BABY-STEP identification. Finally we present additional qualitative results.

Complete set of Evaluation Metrics. We adopt the following set of metrics:

- *Path Length (*PL*)* is the length of the agent's navigation path.
- *Navigation Error (*NE*)* measures the distance between the goal location and final location of the agent's path.
- *Success Rate (*SR*)* that measures the average rate of the agent stopping within a specified distance near the goal location (Anderson et al., 2018)
- *Success weighted by Path Length (*SPL*)* (Anderson et al., 2018) measures the success rate weighted by the inverse trajectory length, to penalize very long successful trajectory.
- *Coverage weighted by Length Score (*CLS*)* (Jain et al., 2019) that measures the fidelity of the agent's path to the reference, weighted by the length score, and the newly proposed
- *Normalized Dynamic Time Warping (*NDTW*)* that measures in more fine-grained details, the spatio-temporal similarity of the paths by the agent and the human expert (Magalhaes et al., 2019).
- *Success rate weighted normalized Dynamic Time Warping (*SDTW*)* that further measures the spatio-temporal similarity of the paths weighted by the success rate (Magalhaes et al., 2019). CLS, NDTW and SDTW measure explicitly the agent's ability to follow instructions and in particular, it was shown that SDTW corresponds to human preferences the most.

C.1 Sanity Check between Prior Methods and Our Re-implementation

Data Splits	R2R Validation Unseen			
Perf. Measures	PL	NE↓	SR↑	SPL
Reported Results				
SEQ2SEQ (Fried et al., 2018)	-	7.07	31.2	-
SF^{+} (Fried et al., 2018)	-	6.62	35.5	-
RCM^{+} (Wang et al., 2019)	14.84	5.88	42.5	-
REGRETFUL$^{+\star}$ (Ma et al., 2019b)	-	5.32	50.0	41.0
FAST$^{+\star}$ (Ke et al., 2019)	21.17	4.97	56.0	43.0
Re-implemented Version				
SEQ2SEQ	15.76	6.71	33.6	25.5
SF^{+}	15.55	6.52	35.8	27.6
RCM^{+}	11.15	6.18	42.4	38.6
REGRETFUL$^{+\star}$	13.74	5.38	48.7	39.7
FAST$^{+\star}$	20.45	4.97	56.6	43.7

Table 6: **Sanity check** of model trained on R2R and evaluated on its validation unseen split ($^{+}$: pre-trained with data augmentation; $\star$:reimplemented or readapted from the original authors' released code).

As mentioned in the main text, we compare our re-implementation and originally reported results of baseline methods on the R2R datasets, as Table 6. We found that the results are mostly very similar, indicating that our re-implementation are reliable.

C.2 Complete Curriculum Learning Results

We present the curriculum learning results with all evaluation metrics in Table 7.

C.3 Results of BABY-STEP Identification

We present an additional analysis comparing different BABY-STEP identification methods. We compare our template-based BABY-STEP identification with a simple method that treat each sentence as an BABY-STEP (referred as sentence-wise), both using the complete BABYWALK model with the same training routine. The results are shown in the

Datasets	Metrics	IL	IL+RL	IL+ CRL w/ LECTURE # 1st	2nd	3rd	4th
R2R	PL	22.4	12.0	11.6	13.2	10.6	9.6
	NE↓	6.8	7.1	6.8	6.8	6.7	**6.6**
	SR↑	28.1	29.8	29.9	33.2	32.2	**34.1**
	SPL↑	15.7	24.3	24.9	26.6	27.5	**30.2**
	CLS↑	28.9	46.2	46.6	47.2	48.1	**50.4**
	NDTW↑	30.6	43.8	42.5	41.0	47.7	**50.0**
	SDTW↑	16.5	23.2	23.1	24.3	25.7	**27.8**
R4R	PL	43.4	22.8	23.9	25.5	21.4	19.0
	NE↓	8.4	8.6	8.5	8.4	**8.0**	8.2
	SR↑	24.7	25.0	24.1	26.7	**27.9**	27.3
	SPL↑	8.2	11.2	11.0	12.3	13.7	**14.7**
	CLS↑	27.9	45.5	44.8	45.9	47.4	**49.4**
	NDTW↑	24.3	34.4	32.8	33.7	38.4	**39.6**
	SDTW↑	11.1	13.6	13.5	15.2	17.0	**17.3**
R6R	PL	68.8	35.3	37.0	40.6	33.2	28.7
	NE↓	9.4	9.5	9.4	9.4	**8.9**	9.2
	SR↑	22.7	23.7	21.9	23.4	24.7	**25.5**
	SPL↑	4.2	7.2	6.4	6.8	8.1	**9.2**
	CLS↑	24.4	43.0	41.8	42.3	44.2	**47.2**
	NDTW↑	17.8	28.1	26.0	26.9	30.9	**32.7**
	SDTW↑	7.7	10.8	9.7	11.0	12.7	**13.6**
R8R	PL	93.1	47.5	50.0	55.3	45.2	39.9
	NE↓	10.0	10.2	10.2	10.1	**9.3**	10.1
	SR↑	21.9	21.4	20.4	22.1	**23.1**	23.1
	SPL↑	4.3	6.1	5.5	6.1	6.8	**7.4**
	CLS↑	24.1	42.1	41.0	41.5	43.9	**46.0**
	NDTW↑	15.5	24.6	22.9	23.8	27.7	**28.2**
	SDTW↑	6.4	8.3	7.9	9.2	10.5	**11.1**
Average	PL	51.8	26.8	27.9	30.6	25.1	22.1
	NE↓	8.5	8.7	8.5	8.5	**8.1**	8.3
	SR↑	24.7	25.5	24.6	27.0	27.5	**28.1**
	SPL↑	8.6	13.1	12.9	13.9	15.1	**16.5**
	CLS↑	26.6	44.5	43.9	44.6	46.2	**48.6**
	NDTW↑	23.0	33.9	32.2	32.4	37.4	**39.0**
	SDTW↑	11.0	14.8	14.4	15.7	17.3	**18.4**

Table 7: Ablation on BABYWALK after each learning stage (trained on R4R).

Datasets	Metrics	Sentence-wise	Template based
R2R	PL	10.3	9.6
	NE↓	6.8	**6.6**
	SR↑	28.7	**34.1**
	SPL↑	24.9	**30.2**
	CLS↑	48.3	**50.4**
	NDTW↑	43.6	**50.0**
	SDTW↑	22.4	**27.8**
R4R	PL	20.9	19.0
	NE↓	**8.2**	8.2
	SR↑	26.3	**27.3**
	SPL↑	12.7	**14.7**
	CLS↑	46.4	**49.4**
	NDTW↑	35.5	**39.6**
	SDTW↑	15.9	**17.3**
R6R	PL	32.1	28.7
	NE↓	**9.0**	9.2
	SR↑	22.5	**25.5**
	SPL↑	7.5	**9.2**
	CLS↑	44.2	**47.2**
	NDTW↑	29.3	**32.7**
	SDTW↑	11.1	**13.6**
R8R	PL	42.9	39.9
	NE↓	**9.8**	10.1
	SR↑	21.2	**23.1**
	SPL↑	6.3	**7.4**
	CLS↑	43.2	**46.0**
	NDTW↑	25.5	**28.2**
	SDTW↑	9.3	**11.1**
Average	PL	24.2	22.1
	NE↓	**8.3**	8.3
	SR↑	25.2	**28.1**
	SPL↑	13.8	**16.5**
	CLS↑	45.9	**48.6**
	NDTW↑	34.6	**39.0**
	SDTW↑	15.4	**18.4**

Table 8: BABYWALK Agent performances between different segmentation rules (trained on R4R). Refer to text for more details.

Table 8. Generally speaking, the template based BABY-STEP identification provides a better performance.

C.4 In-domain Results of Models Trained on Instructions with Different lengths

As mentioned in the main text, we display all the in-domain results of navigation agents trained on R2R, R4R, R6R, R8R, respectively. The complete results of all different metrics are included in the Table 9. We note that our BABYWALK agent consistently outperforms baseline methods on each dataset. It is worth noting that on R4R, R6R and R8R datasets, RCM(GOAL)$^+$ achieves better results in SPL. This is due to the aforementioned fact that they often take short-cuts to directly reach the goal, with a significantly short trajectory. As a consequence, the success rate weighted by inverse path length is high.

C.5 Transfer Results of Models Trained on Instructions with Different lengths

For completeness, we also include all the transfer results of navigation agents trained on R2R, R4R, R6R, R8R, respectfully. The complete results of all different metrics are included in the Table 10. According to this table, we note that models trained on R8R can achieve the best overall transfer learning performances. This could because of the fact that R8R trained model only needs to deal with interpo-

Datasets	Metrics	SEQ2SEQ	SF+	RCM(GOAL)+	RCM(FIDELITY)+	BABYWALK	BABYWALK +
R2R → R2R	PL	15.8	15.6	11.1	10.2	10.7	10.2
	NE↓	6.7	6.5	6.2	6.2	6.2	**5.9**
	SR↑	33.6	35.8	42.4	42.1	42.6	**43.8**
	SPL↑	25.5	27.6	38.6	38.6	38.3	**39.6**
	CLS↑	38.5	39.8	52.7	52.6	52.9	**54.4**
	NDTW↑	39.2	41.0	51.0	50.8	53.4	**55.3**
	SDTW↑	24.9	27.2	33.5	34.4	35.7	**36.9**
R4R → R4R	PL	28.5	26.1	12.3	26.4	23.8	19.0
	NE↓	8.5	8.3	7.9	8.4	**7.9**	8.2
	SR↑	25.7	24.9	28.7	24.7	**29.6**	27.3
	SPL↑	14.1	16.0	**22.1**	11.6	14.0	14.7
	CLS↑	20.7	23.6	36.3	39.2	47.8	**49.4**
	NDTW↑	20.6	22.7	31.3	31.3	38.1	**39.6**
	SDTW↑	9.0	9.2	13.2	13.7	**18.1**	17.3
R6R → R6R	PL	34.1	43.4	11.8	28.0	28.4	27.2
	NE↓	9.5	9.6	**9.2**	9.4	9.4	9.3
	SR↑	18.1	17.8	18.2	20.5	21.7	**22.0**
	SPL↑	9.6	7.9	**14.8**	7.4	7.8	8.1
	CLS↑	23.4	20.3	31.6	39.0	47.1	**47.4**
	NDTW↑	19.3	17.8	25.9	25.8	32.6	**33.4**
	SDTW↑	6.5	5.9	7.6	9.5	11.5	**11.8**
R8R → R8R	PL	40.0	53.0	12.4	42.3	35.6	39.1
	NE↓	9.9	10.1	10.2	10.7	**9.6**	9.9
	SR↑	20.2	18.6	19.7	18.2	**22.3**	22.0
	SPL↑	12.4	9.8	**15.4**	5.3	7.3	7.0
	CLS↑	19.8	16.3	25.7	37.2	46.4	**46.4**
	NDTW↑	15.8	13.5	19.4	21.6	**29.6**	28.3
	SDTW↑	5.1	4.4	5.8	7.6	**10.4**	10.1

Table 9: **Indomain results.** Each model is trained on the training set of R2R, R4R, R6R and R8R datasets, and evaluated on the corresponding unseen validation set (+: pre-trained with data augmentation).

lating to shorter ones, rather than extrapolating to longer instructions, which is intuitively an easier direction.

C.6 Additional Qualitative Results

We present more qualitative result of various VLN agents as Fig 8. It seems that BABYWALK can produce trajectories that align better with the human expert trajectories.

Datasets	Metrics	SEQ2SEQ	SF+	RCM(GOAL)+	RCM(FIDELITY)+	REGRETFUL+⋆	FAST+⋆	BABYWALK	BABYWALK+
R2R → R4R	PL	28.6	28.9	13.2	14.1	15.5	29.7	19.5	17.9
	NE↓	9.1	9.0	9.2	9.3	**8.4**	9.1	8.9	8.9
	SR↑	18.3	16.7	14.7	15.2	19.2	13.3	**22.5**	21.4
	SPL↑	7.9	7.4	8.9	8.9	10.1	7.7	**12.6**	11.9
	CLS↑	29.8	30.0	42.5	41.2	46.4	41.8	50.3	**51.0**
	NDTW↑	25.1	25.3	33.3	32.4	31.6	33.5	38.9	**40.3**
	SDTW↑	7.1	6.7	7.3	7.2	9.8	7.2	**14.5**	13.8
R2R → R6R	PL	39.4	41.4	14.2	15.7	15.9	32.0	29.1	25.9
	NE↓	9.6	9.8	9.7	9.8	**8.8**	9.0	10.1	9.8
	SR↑	20.7	17.9	22.4	22.7	24.2	**26.0**	21.4	21.7
	SPL↑	11.0	9.1	17.7	**18.3**	16.6	16.5	7.9	8.8
	CLS↑	25.9	26.2	37.1	36.4	40.9	37.7	48.4	**49.0**
	NDTW↑	20.5	20.8	26.6	26.1	16.2	21.9	30.8	**32.6**
	SDTW↑	7.7	7.2	8.2	8.4	6.8	8.5	11.2	**11.2**
R2R → R8R	PL	52.3	52.2	15.3	16.9	16.6	34.9	38.3	34.0
	NE↓	10.5	10.5	11.0	11.1	**10.0**	10.6	11.1	10.5
	SR↑	16.9	13.8	12.4	12.6	16.3	11.1	19.6	**20.7**
	SPL↑	6.1	5.6	7.4	7.5	7.7	6.2	6.9	**7.8**
	CLS↑	22.5	24.1	32.4	30.9	35.3	33.7	48.1	**48.7**
	NDTW↑	17.1	18.2	23.9	23.3	8.1	14.5	26.7	**29.1**
	SDTW↑	4.1	3.8	4.3	4.3	2.4	2.4	9.4	**9.8**
Average	PL	40.1	40.8	14.2	15.6	16.0	32.2	29.0	25.9
	NE↓	9.7	9.8	10.0	10.1	**9.1**	9.6	10.0	9.7
	SR↑	18.6	16.1	16.5	16.8	19.9	16.8	21.2	**21.3**
	SPL↑	8.3	7.4	11.3	**11.6**	11.5	10.1	9.1	9.5
	CLS↑	26.1	26.8	37.3	36.2	40.9	37.7	48.9	**49.6**
	NDTW↑	20.9	21.4	27.9	27.3	18.6	23.3	32.1	**34.0**
	SDTW↑	6.3	5.9	6.6	6.6	6.3	6.0	**11.7**	11.6

(a) R2R trained model

Datasets	Metrics	SEQ2SEQ	SF+	RCM(GOAL)+	RCM(FIDELITY)+	REGRETFUL+⋆	FAST+⋆	BABYWALK	BABYWALK+
R4R → R2R	PL	16.2	17.4	10.2	17.7	20.0	26.5	12.1	9.6
	NE↓	7.8	7.3	7.1	6.7	7.5	7.2	**6.6**	6.6
	SR↑	16.3	22.5	25.9	29.1	22.8	25.1	**35.2**	34.1
	SPL↑	9.9	14.1	22.5	18.2	14.0	16.3	28.3	**30.2**
	CLS↑	27.1	29.5	44.2	34.3	32.6	33.9	48.5	**50.4**
	NDTW↑	29.3	31.8	41.1	33.5	28.5	27.9	46.5	**50.0**
	SDTW↑	10.6	14.8	20.2	18.3	13.4	14.2	27.2	**27.8**
R4R → R6R	PL	40.8	38.5	12.8	33.0	19.9	26.6	37.0	28.7
	NE↓	9.9	9.5	9.2	9.3	9.5	8.9	**8.8**	9.2
	SR↑	14.4	15.5	19.3	20.5	18.0	22.1	**26.4**	25.5
	SPL↑	6.8	8.4	**15.2**	8.5	10.6	13.7	8.1	9.2
	CLS↑	17.7	20.4	31.8	38.3	31.7	31.5	44.9	**47.2**
	NDTW↑	16.4	18.3	23.5	23.7	23.5	23.0	30.1	**32.7**
	SDTW↑	4.6	5.2	7.3	7.9	7.5	7.7	13.1	**13.6**
R4R → R8R	PL	56.4	50.8	13.9	38.7	20.7	28.2	50.0	39.9
	NE↓	10.1	9.5	9.5	9.9	9.5	**9.1**	9.3	10.1
	SR↑	20.7	21.6	22.8	20.9	18.7	**27.7**	26.3	23.1
	SPL↑	10.4	11.8	**16.9**	9.0	9.2	13.7	7.2	7.4
	CLS↑	15.0	17.2	27.6	34.6	29.3	29.6	44.7	**46.0**
	NDTW↑	13.4	15.1	19.5	21.7	19.0	17.7	27.1	**28.2**
	SDTW↑	4.7	5.0	5.1	6.1	5.6	6.9	**11.5**	11.1
Average	PL	37.8	35.6	12.3	29.8	20.2	27.1	33.0	26.1
	NE↓	9.3	8.8	8.6	8.6	8.8	8.4	**8.2**	8.6
	SR↑	17.1	19.9	22.7	23.5	19.8	25.0	**29.3**	27.6
	SPL↑	9.0	11.4	**18.2**	11.9	11.3	14.6	14.5	15.6
	CLS↑	19.9	22.4	34.5	35.7	31.2	31.7	46.0	**47.9**
	NDTW↑	19.7	21.7	28.0	26.3	23.7	22.9	34.6	**37.0**
	SDTW↑	6.6	8.3	10.9	10.8	8.8	9.6	17.3	**17.5**

(b) R4R trained model

Datasets	Metrics	SEQ2SEQ	SF+	RCM(GOAL)+	RCM(FIDELITY)+	BABYWALK	BABYWALK+
R6R → R2R	PL	14.5	19.4	8.1	15.5	9.4	9.2
	NE↓	7.7	7.1	7.6	7.5	6.8	**6.8**
	SR↑	19.3	21.9	19.6	22.6	**31.3**	30.6
	SPL↑	13.3	11.6	17.2	14.1	**28.3**	27.8
	CLS↑	32.1	26.2	43.2	34.3	49.9	**50.0**
	NDTW↑	31.9	30.8	39.7	32.4	**49.5**	49.4
	SDTW↑	13.1	13.3	15.3	14.3	**25.9**	25.4
R6R → R4R	PL	25.2	33.0	11.6	25.7	18.1	17.7
	NE↓	8.7	8.6	8.5	8.4	8.4	**8.2**
	SR↑	24.2	22.4	23.6	**25.4**	24.3	24.3
	SPL↑	13.7	9.3	**17.5**	10.6	12.8	12.9
	CLS↑	25.8	21.4	35.8	34.8	48.6	**48.6**
	NDTW↑	22.9	20.6	29.8	26.5	39.0	**39.4**
	SDTW↑	9.3	7.5	10.8	11.1	15.1	**15.1**
R6R → R8R	PL	43.0	52.8	14.2	29.9	38.3	36.8
	NE↓	9.9	9.9	**9.6**	9.7	10.2	10.0
	SR↑	20.1	20.3	20.3	**22.4**	20.8	21.0
	SPL↑	11.2	9.4	**14.9**	8.1	6.6	6.8
	CLS↑	20.6	18.3	27.7	38.9	45.9	**46.3**
	NDTW↑	16.3	15.2	21.9	22.2	28.4	**29.3**
	SDTW↑	5.6	5.0	6.4	6.8	9.6	**9.9**
Average	PL	27.6	35.1	11.3	23.7	21.9	21.2
	NE↓	8.8	8.5	8.6	8.5	8.5	**8.3**
	SR↑	21.2	21.5	21.2	23.5	**25.5**	25.3
	SPL↑	12.7	10.1	**16.5**	10.9	15.9	15.8
	CLS↑	26.2	22.0	35.6	36.0	48.1	**48.3**
	NDTW↑	23.7	22.2	30.5	27.0	39.0	**39.4**
	SDTW↑	9.3	8.6	10.8	10.7	**16.9**	16.8

(c) R6R trained model

Datasets	Metrics	SEQ2SEQ	SF+	RCM(GOAL)+	RCM(FIDELITY)+	BABYWALK	BABYWALK+
R8R → R2R	PL	13.7	19.3	7.8	17.8	9.1	9.8
	NE↓	7.6	7.3	8.0	8.2	6.8	**6.7**
	SR↑	18.7	23.4	14.8	19.2	30.0	**32.1**
	SPL↑	13.3	12.9	12.9	10.6	27.0	**28.2**
	CLS↑	32.7	26.6	37.9	28.9	**49.5**	49.3
	NDTW↑	32.4	29.9	34.9	25.9	48.9	**48.9**
	SDTW↑	12.7	14.5	11.1	10.5	24.6	**26.2**
R8R → R4R	PL	23.1	31.7	11.1	32.5	17.4	19.0
	NE↓	8.7	8.8	8.7	9.2	**8.2**	8.5
	SR↑	23.6	21.8	23.2	21.7	24.4	**24.4**
	SPL↑	15.1	10.5	**18.2**	7.4	12.6	12.5
	CLS↑	24.9	20.8	32.3	29.4	48.1	**48.5**
	NDTW↑	22.3	19.7	26.4	20.6	**39.1**	38.5
	SDTW↑	8.8	7.7	9.3	8.4	14.9	**15.2**
R8R → R6R	PL	30.9	42.2	11.9	39.9	26.6	29.2
	NE↓	9.7	9.9	9.9	10.1	**9.0**	9.3
	SR↑	15.4	14.7	14.8	20.0	**22.9**	22.9
	SPL↑	8.6	6.7	**11.6**	5.3	8.4	7.9
	CLS↑	22.2	18.5	29.1	33.5	**46.9**	46.6
	NDTW↑	18.5	15.9	22.5	20.1	**33.3**	31.8
	SDTW↑	5.5	4.7	6.0	7.8	**12.1**	11.8
Average	PL	22.6	31.1	10.3	30.1	17.7	19.3
	NE↓	8.7	8.7	8.9	9.2	**8.0**	8.2
	SR↑	19.2	20.0	17.6	20.3	25.8	**26.5**
	SPL↑	12.3	10.0	14.2	7.8	16.0	**16.2**
	CLS↑	26.6	22.0	33.1	30.6	**48.2**	48.1
	NDTW↑	24.4	21.8	27.9	22.2	**40.4**	39.7
	SDTW↑	9.0	9.0	8.8	8.9	17.2	**17.7**

(d) R8R trained model

Table 10: **Transfer results** of R2R, R4R, R6R, R8R trained model evaluated on their complementary unseen validation datasets (+: pre-trained with data augmentation; ⋆: reimplemented or readapted from the original authors' released code).

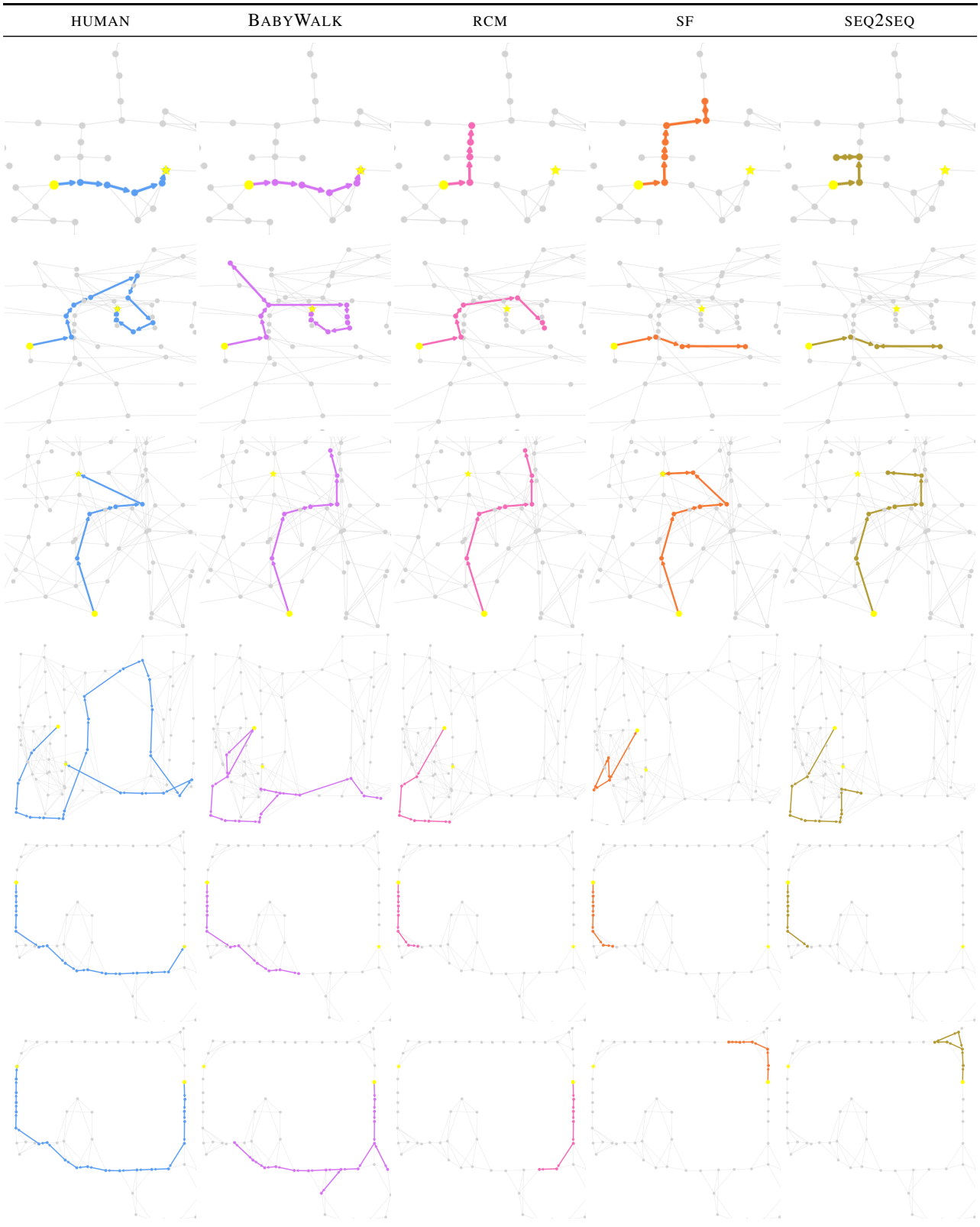

Figure 8: Additional trajectories by human experts and VLN agents on two navigation tasks.