11-777 Spring 2021 Class Project

Pengkun Liu* **Ruoxin Xiong*** **Songhao Jia***
{pengkunl, ruoxinx, songhaoj}@andrew.cmu.edu

Abstract

In multi-modal machine learning area, how to deal with Out-of-Distribution (OOD) data is a critical question. Under the imitation learning scenario, without the exploration ability, the modal would get unpredictable result given OOD input. What's more, the scene and object distribution may also vary between training data and testing data. To address the problem, we propose two solutions. We first present a series targeted data augmentation method to improve the robustness of the model under different distributions. Secondly, we demonstrate a Graph Neural Network to capture the relationship between objects. This approach assists the model to identify objects that out of the distribution. We do all experiments on ALFRED dataset and our approach improves the performance of state-of-the-art model.

1 Introduction

Multi-model machine learning models have attained remarkable performance with the assumption that the test data distribution is similar to the training data and. However, these models are unable to identify inputs that are different from those observed during the training stage (Amodei et al., 2016). The reality is that this assumption doesn't hold true leading to a significant drop in their performances encountering with new situations or inputs, i.e., the Out-of-Distribution (OOD) data.

As for the ALFRED dataset (Shridhar et al., 2020a), it aims to build an instruction-based visual navigation and interaction system, in which a goal statement, step-wise instructions, and current egocentric scene together serve as guidance for generating reasonable actions and predicting interaction masks of objects. As a complex and challenging task, there are several potential subtasks we need to address. More precisely, training by the imitation learning without any exploration experiences, the model would fail and get unpredictable result given OOD input. What's more, the scene and object distribution may also vary between training data and testing data.

In this work, we first present a series targeted data augmentation method to improve the robustness of the model under different distributions. Secondly, we demonstrate a Graph Neural Network to capture the relationship between objects. This approach assists the model to identify objects that out of the distribution. The experimental results on ALFRED dataset could achieve a better performance than the state-of-the-art model.

2 Related Work

Vision & Language Navigation Vision-and-language navigation (VLN) tasks require an embodied intelligent agent to sequentially follow instruction guided visual navigation and object interactions in 3D environments through egocentric observations (Anderson et al., 2017; Wang et al., 2018; Fried et al., 2018; Ma et al., 2019b; Hao et al., 2020; Shridhar et al., 2020b).

In order to tackle the VLN tasks, Mei et al. (2015) proposed a long short-term memory recurrent neural networks (LSTM-RNN) sequence-to-sequence architecture to map natural language instructions to a sequence of navigation actions given observable environments. Anderson et al. (2017) introduced the Room-to-Room (R2R) dataset, the first benchmark for VLN tasks and explored several LSTM sequence-to-sequence baseline models. Wang et al. (2018) then established a hybrid Reinforced Planning Ahead (RPA) method that integrates model-free and model-based reinforcement learning (RL) strategies to enhance the model generalizability when transferring to unseen environments. Fried et al. (2018) built an em-

*Everyone Contributed Equally – Alphabetical order

bedded speaker-follower model that incorporates synthetic data augmentation, pragmatic reasoning and panoramic action space search, which dramatically improves the model performance on the R2R dataset. Lately, Wang et al. (2019) proposed a Reinforced Cross-Modal Matching (RCM) approach that combines RL and Self-Supervised Imitation Learning to encourage cross-modal matching locally and globally. Ma et al. (2019a) introduced a self-monitoring module that uses visual-textual grounding and progress monitor to accurately evaluate the navigation progress. To better generalize the model to unseen scenes, Tan et al. (2019) presented an environmental dropout method that improves the variability of the training environments.

Different from previous VLN benchmarks such as R2R dataset (2017) and Touchdown dataset (2019b), the recent proposed ALFRED dataset (2020b) provides both high-level goal statements and low-level language instructions for visual navigation and pixel-wise object interaction via egocentric observations. Shridhar et al. (2020b) set up a CNN-LSTM sequence-to-sequence baseline model with progress monitoring (2019a) for ALFRED tasks. Singh et al. (2020) proposed a Modular Object-Centric Approach (MOCA) that incorporates action policy module and visual perception module to predict sequential actions and produce pixel-wise interaction masks. Jansen (2020) investigated the visual semantic planning task in ALFRED by translating natural language instructions into detailed sequences of navigation actions.

Referring Instruction Comprehension To accomplish the series of tasks presented by ALFRED (Shridhar et al., 2020b), one of the crucial techniques is to locate the objects referred by instructions, and that is exactly what referring image localization and segmentation task is about. Hu et al. (2016) first proposed a model to address the problem in the 2D image domain, which generated segmentation output by U-Net-like (Ronneberger et al., 2015) upsampler based on composed textual, visual, and spatial features. After that, two main approaches were formed. One approach sought to improve the model performance by improving the ability to fuse the multi-modal features. Liu et al. (2017) further improved the jointly modeling mechanism by concatenating multi-modal features with the textual feature of each word and processing it via convolutional multimodal LSTM. Li et al. (2018) gradually fused visual features from high level to low level with textual features to enhance performance. Chen et al. (2019a) introduced a novelty per-pixel attention module to generate heatmap based on textual feature and each stage of visual feature. The module paid attention to the similar pixel between two types of feature, as well as the heatmap generated in the previous stage. Hu et al. (2020) represented a bi-directional interaction between language and vision, increasing the model's attention on the pointed object. Another approach was trying to address the problem by aligning the object relationships captured from both image and text. Yu et al. (2018) parsed the text expression into several embeddings and computing the matching score with the visual feature. Huang et al. (2020) highlighted all entities similar to the content words in the expression, then get the generated the final prediction by relationship words .Hui et al. (2020) construct a graph representing linguistic structure by Gather-Propagate-Distribute mechanism to select the most relevant segment.

Also, some works are trying to address the referring image segmentation problem in other data domains, such as video and 3D scenes. In the video domain, Khoreva et al. (2018) captured the segmentation in two steps. They predicted the box proposal first by the newly designed matching score to improve temporal consistency. Then it applied a segmentation convnet to recover the details of object masks. Seo et al. (2020) presented a memory module that contained the relevant information from the previous frames. It adopted a cross-modal attention module to fuse the feature among text, vision, and the memory module. Moreover, there are some breakthroughs in Referring 3D Instance Segmentation. Huang et al. (2021) built a novel GNN model to encodes both the spatial context of the 3D scene and the multi-modal features to propose corresponding 3D instances.

Unseen Objects and Scenes For a robot to work in an unstructured environment like ALFRED (Shridhar et al., 2020b), it must have the ability to recognize new objects and scenes that have not been seen before. It is impractical and infeasible to model every object in the environment. It is a challenging perception task for robot to recognize the unseen objects since the robot needs to learn the concept of "objects" and generalize it to unseen objects (Xie et al., 2020b,a). It is significant to build up a robust object and scene recognition module for robots interacting with objects in the

typical scenes.

For the unseen object recognition, Nagarajan and Grauman (2018) presented a new approach to model visual attributes as operators, learn a semantic embedding that explicitly factors out attributes from their accompanying objects, and also benefits from novel regularizers expressing attribute operators' effects. After that, Hsieh et al. (2019a) developed a co-attention and co-excitation framework to solve the one-shot object detection problem with a query image whose class label not included in the training data. For the one-shot object detection, Osokin et al. (2020) built one-stage system that performed localization and recognition jointly with dense correlation matching of learned local features to find correspondences. Furthermore, in order to solve the problem of Unseen Object Instance Segmentation (UOIS), Xie et al. (2020b) proposed UOIS-Net separately leverages synthetic RGB and synthetic depth to segment every arbitrary and potentially unseen object instance in the tabletop environments.

For the unseen scene recognition, Gordon et al. (2017) proposed the Hierarchical Interactive Memory Network (HIMN), consisting of a factorized set of controllers, allowing the system to operate at multiple levels of temporal abstraction. Zhu et al. (2017) introduced a target-driven model to achieve higher adaptability and flexibility of visual navigation. A key intuition is to share information between different training episodes. For example, agents explore common routes during the training stage while being trained for finding different targets. Various scenes share similar structures and statistics, like a fridge is high likely to near a microwave (Zhu et al., 2017). Yang et al. (2018) incorporated semantic priors in the task of semantic navigation by a graph convolutional networks for incorporating the prior knowledge into deep reinforcement learning framework to predict the actions. Tan et al. (2019) presented a generalizable navigational agent consisting of mixed imitation and reinforcement learning and fine-tuning via newly-introduced 'unseen' triplets (environment, path and instruction). In order to generate the unseen triplets, Tan et al. (2019) proposed a 'environmental dropout' method to mimic unseen environments. Moghaddam et al. (2020) proposed using externally learned prior knowledge of object locations with a neural graph named Graph-based Value Estimation (GVE) module for estimating the advantage function in the actor-critic RL algorithm.

3 Data Statistics and Analysis

Under the rapid growth of deep neural networks the combination between computer vision and natural language processing gain much attention. In this project, we mainly focus on the ALFRED dataset. The project aims to build an instruction-based visual navigation and interaction system, in which a goal statement, step-wise instructions, and current egocentric scene together serve as guidance for generating reasonable actions and predicting interaction masks of objects.

As a complex and challenging task, there are several potential subtasks we need to address. How does the system deal with unseen objects is what we want to mainly focus on. In this subtask, the system needs to generated bounding-boxes and masks given visual features of objects that have never been seen before. Identify and localize a never-before-seen object is a primary ability of humans, but is extremely difficult for machines, which causes a huge performance gap between humans and machines in the unseen split of the ALFRED dataset. Previous results on the ALFRED dataset, like (Singh et al., 2020), did not address this issue. However, some works, like (Hsieh et al., 2019b) and (Osokin et al., 2020), proposed some solutions, and we believe that the prediction performance of unseen objects can be further improved.

Also, another subtask involved in this problem is the sequential action prediction. The model needs to predict the action based on the former action, a textual feature of the current instruction and statement goal, and an egocentric scene. However, the flexibility of natural language expressions, the judgment of the current state, and varying length of different instructions all make it harder to address the problem and become one of the largest barriers to performance improvement.

3.1 Expert Demonstrations

Each task is characterized by task type, object, destination, and scene. In total, ALFRED *v2.1.0* dataset consists of 3,733 task parameters (i.e., train: 2,435, valid_seen: 242, valid_unseen: 85, test_seen: 483, and test_unseen: 488). The task parameters are detailed in Table 1. There are 120 household scenes in the ALFRED dataset, across kitchens, living rooms, bedrooms, and bathrooms. The 7 task types include pick and place, stack and place, pick

Parameters	Number	Category
Task type	7	pick and place, stack and place, pick two, clean and place, heat and place, cool and place, examine
Scenes	120	kitchens, living rooms, bedrooms, and bathrooms
Object classes	58	potato, tomato, lettuce, apple, mug, egg, and etc.
Receptacle classes	26	cabinet, countable, sink basin, fridge, toilet, dining table, and etc.
Actions	13	5 navigation actions, 7 interaction actions, and 1 stop action

Table 1: Task parameter settings in the ALFRED dataset.

two, clean and place, heat and place, cool and place, and examine. The agent interacts with the objects or environment with 13 predefined actions, including 7 interaction actions, 5 navigation actions, and 1 stop action.

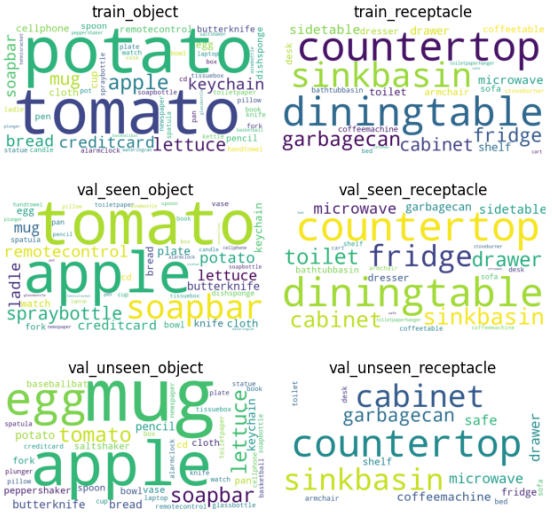


Figure 1: Word clouds of objects (left) and receptacle (right) in the train, validation *seen* and *unseen* fold.

For each task parameter set, three different expert demonstrations are generated by randomly placing the agents and objects in the scene. Fig. 1 illustrates the relative amount of objects (left) and receptacles (right) defined in tasks in the form of the word cloud. It shows that the distribution of objects and receptacles are in different categories among the train, validation *seen* and *unseen* fold. For example, two of the most important objects in the training subset are potato and tomato, while the two largest number of objects in the validation seen fold are tomato and apple.

3.2 Instructions

For each expert demonstration, ALFRED provides at least three different sets of language instructions. There are two kinds of instruction in each set: a goal statement and a list of step-by-step instructions, whose number varies from demonstration to demonstration and would fall somewhere between 3 to 19. Figure. 23 shows the distribution of step-wise instruction numbers in different datasets. We can observe that within each dataset, the distribution of instructions number is similar. Further, we also analyze the number of words used in both target and step-wise instruction. Figure. 24 and figure.25 show the details. We find that the range of words in goal instruction and step-wise instruction are [2, 34] and [2, 55], indicating that there would be some long-tail problems we need to deal with.

As for the distributions of the actions in the high level instructions, as shown in the Figure. 26, Go to Location, Pick up Object, Put Object and NoOp (Flags of the finished tasks) take up the majority of the high level actions. In the low level actions as shown in Figure.27, MoveAhead25, RotateRight90, RotateLeft90, LookDown15, LookUp15, PickupObject and PutObject are the major actions. Furthermore, the comparisons of the indexes of high level and low level actions are shown in Figure. 28, in the low level actions, the agents are high likely to take actions of indexes 0 and 3. While in the high level, the numbers of actions decrease as the indexes increasing.

3.3 Images and Masks

ALFRED dataset consists of 45,681 bounding boxes (bboxes) and masks separately (i.e., train: 42,354, valid seen: 1,636 and valid unseen: 1,691).Due to the setting of the environment, the dimensions of the screen are 300×300 pixels. Furthermore, the bboxes and masks are not labeled in the test dataset, therefore in the Figure. 2 the distribution of the masks and bboxes are compared in the train, valid seen and valid unseen datasets. It shows that in most of the scenes, the objects are located in the middle or upper-middle of the screen. In addition, masks are irregular polygons, while

bboxes are rectangles. Consequently, it is worth exploring which way to represent objects could result in better performances of the model.

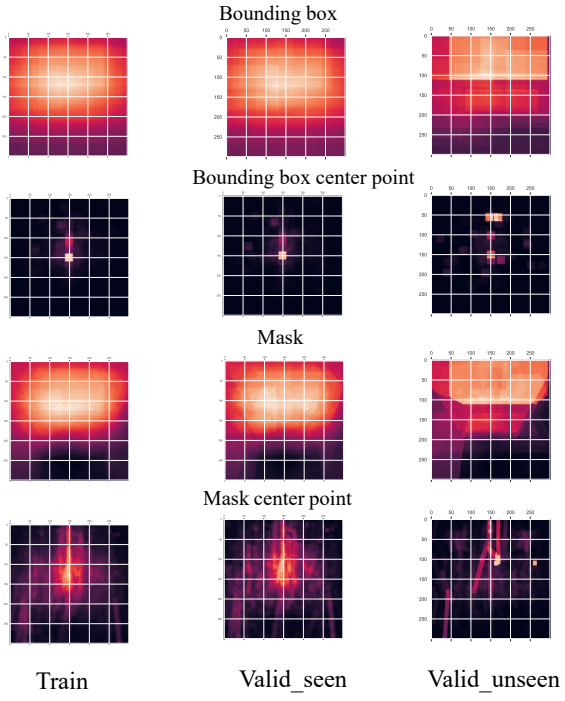


Figure 2: Distributions of the location of interactable objects. The first row and the third row demonstrate the distributions of whole bounding box and mask, and the second row and fourth row indicates the central points of them. The brightness of each pixel shows the frequency the object shown in this point.

4 Baseline and Metrics

As MOCA (2020a) has achieved the highest performance among all available results, we choose this approach as our baseline model. MOCA (2020a) integrates two modules, the action policy module and visual perception module, to predict sequential actions and generate pixel-wise interaction masks.

4.1 Proposed Baseline

Input ablations To explore the bias of vision and language modalities in MOCA model, the input ablations are shown in Table 6. The unimodal ablations demonstrate that visual and language information are essential for the ALFRED tasks. Particularly, the performance of MOCA is higher than previous methods (2020a) even with only one of the step-by-step language instructions (Instructions-Only) or goal statements (Goal-Only).

Model ablations To demonstrate the importance of each module, MOCA (2020) implemented model ablation experiments (see Table 7). As the model performance drastically drops after removing the individual components, the results illustrated the significance of each component of MOCA model. Table 8 shows the ablations on the Instance-Association in Time (IAT) and obstruction detection. Particularly, the performance of the MOCA model almost drops half after removing the IAT module, which indicates that it is essential to ensure the correct objects of interactions.

4.2 Existing metrics

Some metrics have been proposed and employed in the Sequence-to-Sequence baseline (2020a) and MOCA (2020) to evaluate the performance of their approach on the ALFRED dataset.

Task Success Rate There would be a specific goal statement for each task, which is used to express the overall task content. The task success rate is the proportion of the goal statement completed in the given dataset.

Goal Condition Success Rate For each task, the goal statement can be split into a variable number of sub-goals. Goal Condition Success Rate indicates the percentage of the sub-goals accomplished. It is worthy to notice that for each task, its goal statement would be completed if and only if all its sub-goals are completed.

Path-length-weighted Success Rate To guide the model's interaction with the simulated environment, ALFRED provides expert demonstrations as the ground truth, which adopt shortest path navigation. A path-length-weighted success rate is proposed to estimate the quality of the predicted action by comparing the total number of predicted actions and expert demonstrations. The equation is shown as follows:

$$p_s = s \times \frac{L_e}{max(Le, Lp)} \quad (1)$$

Where L_e and Lp indicates the total number of actions in expert demonstrations and model predictions.

4.3 Proposed Intrinsic Metrics

In order to address the detection and reasoning problem raised by unseen objects and unseen scenes with different textures, colors, shapes and object distributions in ALFRED dataset (2020a), several intrinsic metrics are proposed to measure

Model	Validation				Test			
	Seen		*Unseen*		*Seen*		*Unseen*	
	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond
Shridhar et al. (2020a)	3.70 (2.10)	10.00 (7.00)	0.00 (0.00)	6.90 (5.10)	3.98 (2.02)	9.42 (6.27)	0.39 (0.80)	7.03 (4.26)
Singh et al. (2020)	19.15 (13.60)	28.50 (22.30)	3.78 (2.00)	13.40 (8.30)	22.05 (15.10)	28.29 (22.05)	5.30 (2.72)	14.28 (9.99)
Human	-	-	-	-	-	-	91.00 (85.80)	94.50 (87.60)

Table 2: Task and Goal-Condition Success Rate for existing approaches. All values are percentages. * and - indicate the data will be replenished later and not available.

the complexity of tasks and visual environments, understanding of objects and scenes of the agent, which would help understand the performances of the proposed system.

Complexity of Task In order to measure the complexity of each task, we include a metrics that considers two sub metrics: Number of Instruction and Total Length of instruction.

1) **Number of Instruction** is the total number of low level instructions of each task for measuring the numbers of actions a agent needs to take to complete a task.

2) **Total Length of instructions** is the total lengths of texts of instruction of each task for measuring the numbers of texts a agent needs to understand to complete a task.

Complexity of visual environment In order to measure the complexity of each task, we include a metrics that considers four sub metrics: Number of Object, Object Distribution of Environment, Unseen objects recognition success, and Number of Obstruction.

1) **Number of Objects** including seen and unseen objects of each task measures the numbers of objects a agent needs to detect and recognize to complete a task.

2) **Average Pair-Wise Distance** is the total distances of objects' spatial coordinates in each task for measuring the degree of dispersion of objects in the environment.

3) **Unseen Object Recognition Success** In each epoch, when the textures, colors or shapes of the objects change, if the agent could successfully recognize the class of the object, then an unseen objects recognition success is defined.

4) **Number of Obstruction** Considering the unseen task "Put a hot potato slice on the counter", the agent succeeds if recognizing the unseen potato successfully during the episode. According to MOCA (2020), the agent tends to get stranded around immovable obstacles that would eventually lead to navigation failure. Therefore, it is significant to measure the numbers of agent strands occurred to understand the reasons behind this phenomenon.

5 Analysis of Baselines

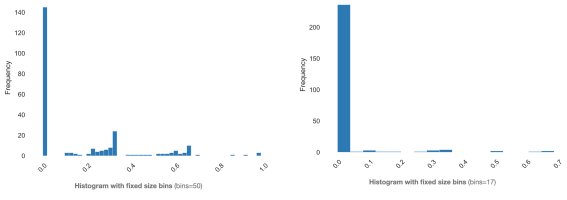
5.1 Overall task performance analysis

We first analyze the overall model performance using the metrics including task success rate, goal condition success rate, and path-length-weight success rate. Also, we report the model results running on our own machine.

Task Success Rate The task success rate is the proportion of the goal statement completed in the given dataset.

Goal Condition Success Rate Goal Condition Success Rate indicates the percentage of the subgoals accomplished.

Path-length-weighted Success Rate A path-length-weighted success rate is proposed to estimate the quality of the predicted action by comparing the total number of predicted actions and expert demonstrations, as shown in Fig. 3 and Fig. 4. Compared to the validation seen dataset, the MOCA model has achieved relatively lower performance on the unseen scenarios.

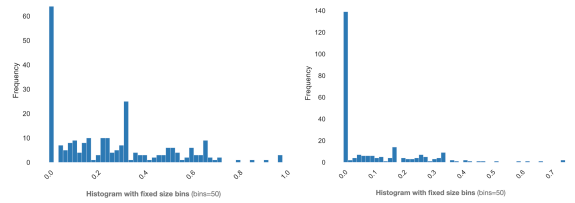


(a) Validation seen dataset (b) Validation unseen dataset

Figure 3: Path-length-weighted Success Rate

5.2 Intrinsic task performance

We propose several intrinsic metrics to measure the complexity of tasks and visual environ-



(a) Validation seen dataset (b) Validation unseen dataset

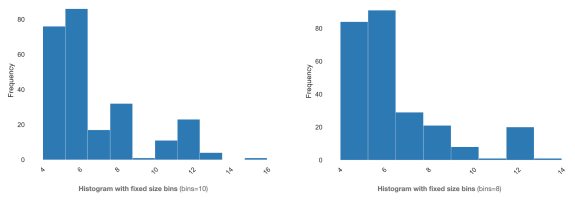
Figure 4: Goal Condition Path-length-weighted Success Rate

ments,understanding of objects and scenes of the agent,which would help understand the performances of the proposed system.

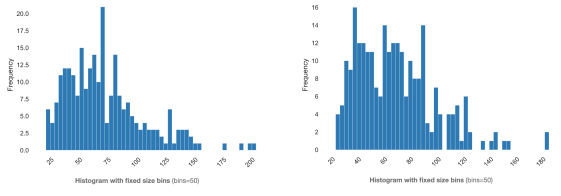
5.2.1 Complexity of task instructions

To evaluate the effects of task instruction on the model performance, the authors first analyze the complexity of step-by-step instructions considering the number of sequential instructions and total length of the instructions.

As shown in Fig. 5 (a), the minimum number of goal instructions is 4 while the maximum number is 16. And the average number of goal instructions is 6.6. As shown in Fig. 5 (b), the minimum length of instructions (i.e., word counting) is only 21 and the maximum length is 205 with an average length of 72.6 words.



(a) Statistics of the instruction counts



(b) Statistics of the instruction length

Figure 5: Statistics of step-by-step goal instructions

1) **Number of Instructions**

Number of instruction is the total number of low level instructions of each task for measuring the numbers of actions a agent needed to take to complete a task.

We analyze the effects of the number of sequential instructions on the model performance, as shown in Fig. 33. On both seen and unseen dataset, the model is more likely to complete the goal instructions with less number of instructions. Particularly, on the unseen scenarios, almost every successful cases with goal condition success rate greater than 0.2 have less than 9 sequential instructions. We also investigate the effects of the number of sequential instructions on the Path-length-weighted Success Rate (see Fig. 34) and goal condition Path-length-weighted Success Rate (see Fig. 35). We can conclude that the number of instructions negatively affect the model performance including the task success rate, goal condition success rate, and Path-length-weighted Success Rate.

2) **Total Length of instructions**

Total Length of instructions is the total lengths of texts of instruction of each task for measuring the numbers of texts a agent needed to understand to complete a task.

Then, we analyze the effects of the total Length of instructions on the model performance, as shown in Fig. 36. Similar to the number of instructions, the length of instructions also negatively affect the performance of goal-condition success rate, especially for the unseen scenes. Most of the success tasks with goal condition rate greater than 0.2 has instructions less than 120 words.

We also explore the effects of the length of sequential instructions on the Path-length-weighted Success Rate (see Fig. 37) and goal condition Path-length-weighted Success Rate (see Fig. 38). Similar to the number of instructions, we can conclude that the length of instructions negatively affect the model performance including the task success rate, goal condition success rate, and Path-length-weighted Success Rate.

5.2.2 Complexity of visual environment

1) **Number of Objects**

In order to have a deeper understanding of the number of objects whether affect the performances of the model, the comparison of success rates with different object numbers have been done, as shown in the Table 9 and Figure 6 and 7. It should be noticed that the success rates do not significantly decrease with the increments of the object numbers in the environments. But in a small scale, the success rates do decrease as the numbers of objects increase. This phenomenon will be explained by the success rates are affected by many factors like the types of the tasks.

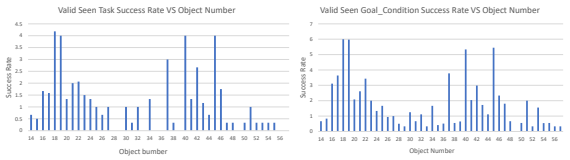


Figure 6: Valid Seen Success Rate VS Object Number

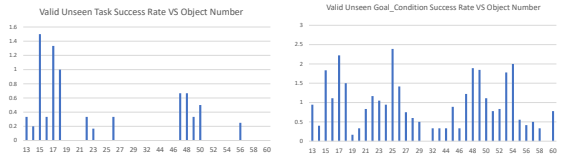


Figure 7: Valid Unseen Success Rate VS Object Number

2) Average Pair-Wise Distance

In order to have a deeper understanding of the distances of different objects whether affect the performances of the model, the comparison of success rates with different object distances have been done, as shown in the Table 9. It should be noticed that the success rates do not significantly decrease with the increments of the average pair-wise distances on the valid seen data set. But in the valid unseen data set, the success rates do decrease as the average pair-wise distances increase. This phenomenon illustrates that an agent is more sensitive to the complexity of unknown environments compared to the known environments.

Figure 8: Valid Seen Success Rate VS Average Pair-Wise Distance

3) Object Recognition Success

To further discuss about the performance of MOCA (2020) about object recognition, we first analyse the success rate of model identifying the object can be interacted with on different data splits. MOCA has high performance on recognizing the category of the object need to be interacted with on training and validation-Seen splits. Even though the model performs relative poorly on validation-Unseen split, the performance is comparable with others. We believe that due to the categories to be predicted are shown up in the goal statement, the

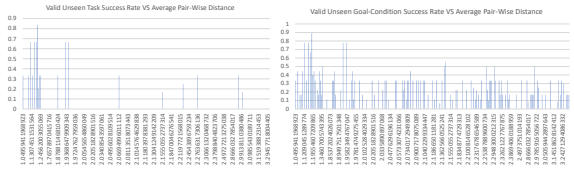


Figure 9: Valid Unseen Success Rate VS Average Pair-Wise Distance

modal may more rely on the goal statement to make the prediction. Therefore, changing the distribution of the visual feature may not hurt the performance that much.

Secondly, to have a deeper understanding about the relationship between recognition success rate and the object category, we calculate the probability of successful identification of different kinds of objects. The results is shown as Fig 10:

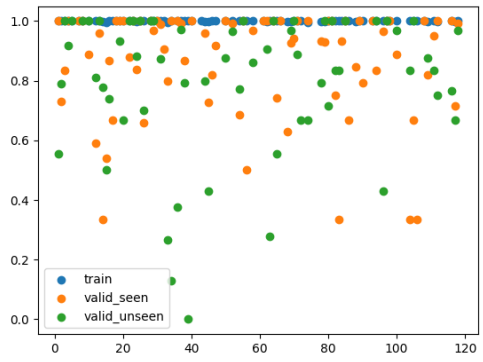


Figure 10: **Identification success rate on different categories**. Vertical axis and horizontal axis indicate success rate and category id.

According to the Fig 10, we can notice that MOCA (2020) can get perfect performance on Training split for almost all categories. For Validation splits, although there is a gap between the performance on seen split and unseen split, they share a roughly same trend. Last, the identification success rate varies greatly among different categories. To further improve the overall object identification performance, the categories with lower recognition success rate will be our focus.

Finally, the change of scene is also one of the important reasons that affect the object recognition rate. A new scene is somehow a new noise distribution for target object recognition. Fig 11 shows the relationship between scene and recognition success rate:

From the above figure, we can draw a similar

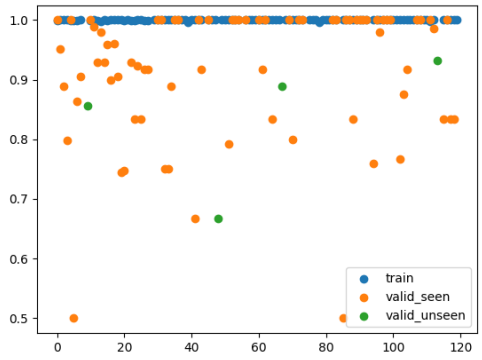


Figure 11: **Identification success rate on different scenes**. Vertical axis and horizontal axis indicate success rate and scene id.

conclusion to the previous experiment: MOCA (2020) works perfect on all scenes in Train split; Validation-Unseen and Validation-Seen splits share the similar performance with homogeneous scene; the recognition performance varies greatly among different scenes.

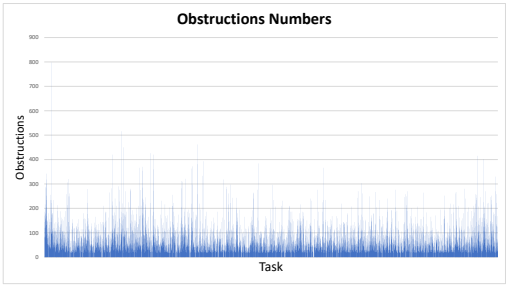


Figure 12: Obstruction Numbers

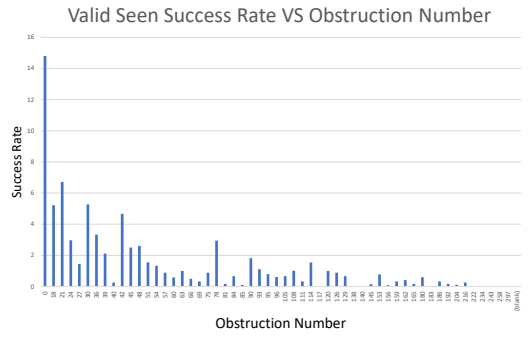


Figure 13: Valid Seen Success Rate VS Obstruction Number

4) **Number of Obstruction**

We define that if agents perform repeated actions several times then the agents meet the obstruction.

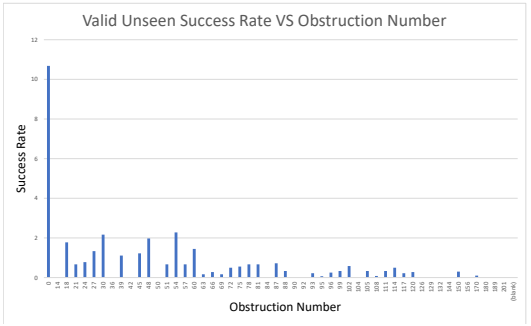


Figure 14: Valid Unseen Success Rate VS Obstruction Number

Percentage	Total Loss	Loss Action Low	Loss Action Low Mask
1	0.7748	0.1782	0.5926
2	0.8437	0.2224	0.3672
5	1.0720	0.4856	0.4498

Table 3: Losses on train split with different proportions. All results are obtained after 17k training steps.

As shown in the Figure 12, 13 and 7, in most of the tasks, agents will take actions when encountering the obstructions. Furthermore, the number of obstructions is negative correlated to the success rate of the tasks no matter in the seen and unseen environments, which means that the obstructions would significantly affect the performance of the models.

5.3 Quantify the Memorisation Ability

In order to estimate the memorisation performance of MOCA (2020), we randomly choose 1%, 2% and 5% of training dataset and use them to train the modal, observing the total loss under three kinds of situations. All data All The total losses are shown in Tab 3.

From Tab 3 we can see that the with the decrease of training data, all kinds of losses are also decreased. However, the modal can not get perfect performance even on the smallest task, which is just 1% of original training set, indicating that the memorisation ability of MOCA is still limited.

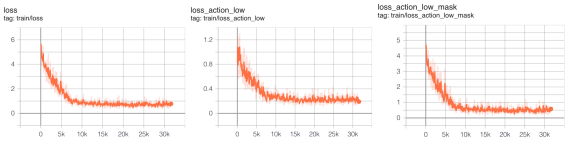


Figure 15: MOCA model training process on 1% of training dataset

6 Proposed Approach

6.1 Motivation

ALFRED (Shridhar et al., 2020a) is a language-based navigation dataset, which contains several difficult sub tasks, including navigation, interaction, and so on. Therefore, diversity of data is important for the model to learn and understand the environment. However, manually tagged data is very expensive, especially in such complex task. What's more, we also notice that the data distribution is inconsistency between training and validation phases, not only some unseen objects would occur during validation, but domain shift as well. To address the two main problem, our approach can be divided into two parts: data diversity enrichment and data relationship extraction.

(a) Alarm clock of unseen dataset (b) Alarm clock of seen dataset

Figure 16: Seen and unseen objects

6.2 Data Diversity Enrichment

To address the serious delay of the ALFRED simulator generating new scene according to the previous scene and predicted action, based on manually annotated trajectories, SOTA models adopt imitation learning mechanism to train the modal. However, unlike online reinforcement learning method, which can definitely get feedback for any actions under any scene based on the interaction with the real environment, imitation learning algorithm must rely entirely on given static dataset, so that it is impossible to do any exploration. If any OOD data input the modal, the result would be unpredictable. The main reason behind such phenomenon would be the limitation of dataset. The static dataset fails to cover all state space and its corresponding transitions.

Semantic Instruction Set Augmentation To solve this problem, one of our approaches is to do semantic instruction set augmentation. Since different users have different ranges of task concerns, the resulting instructions differ even for the same trajectory. Therefore, in the current setting of imitation learning, the lack of exploration of different semantic instructions and the cost of adding new semantic instruction annotations make it necessary to perform targeted augmentation of the semantic instruction set. We adopt the approach of mixing existing instructions under different annotations for semantic instruction enhancement. For example, let's say that any existing instruction would be a^i_j, where i is the id of the annotator and j is the id of current instruction. Assume that there are m different annotators and n different instructions in each annotation, we define any augmented annotation b^i_j as follow:

$$b^i_j = a^{i^*}_j, i^* \in [0, 1, ..., k], \forall j \in [0, 1, ..., m] \quad (2)$$

Object Augmentation Another solution we propose is to do data augmentation on the objects in the scene. When the model is used in real life, it is inevitable that there will be objects in the input that have not been seen during the training process, so it is necessary to input as many kinds of objects as possible during the training process. However, due to the limitation of the simulator, it is difficult to modify the objects in the scene directly, so we use the rotated objects to solve this problem. Fig.17 shows one example of object rotation.

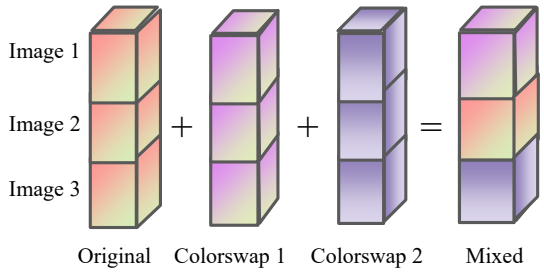
(a) Original Pose of laptop (b) Changed locations of laptop (c) Changed rotations of laptop

Figure 17: Change the locations and rotations of the laptop

This method has two main benefits. One is the ability to cope with the problems caused by the simulator environment during data augmentation. We found that data augmentation by other means may lead to simulator errors due to excessive modifications to the original data, and it is difficult to determine whether the amount of data modification is appropriate through simple commands, making it difficult to deploy in our experiments on a large scale. The rotation operation can maximize object diversity while reducing the chance of simulator errors. Second, the material of an object is composed of materials in all directions. In the original setup,

only one of the orientations is used as input to the model for each training session, which in effect reduces the diversity of objects in the dataset. By artificially rotating the object, we can expose the whole material of the object to the dataset better and maximize the diversity of the available data.

Mixing Different Feature Maps Furthermore, we propose another solution by mixing different feature map of the videos in order to enrich the colors and textures of the objects and scenes as shown in the Fig. 18. More precisely, one cube stands for one image, the video consists of a sequences of images (cubes). By changing the color of the videos, the different colorswaps are generated to mix with the original videos. For example, the different feature maps are shown in the Fig. 18



(a) Data augmentation by mixing different feature maps

(b) Example of the different feature maps

Figure 18: Data augmentation by mixing different feature maps

6.3 Data Relationship Extraction

The scene priors (functional/semantic) cam help the agents navigate navigate to targets in unknown scenes or find unseen objects (Yang et al., 2019). For example, if the agents have never seen the potato before while the goal-condition task is to search for the potato in a new kitchen. If agents have ever found apples before, a similar searching strategy can be applied. In addition, since the vegetables like potato are typically stored in the fridge, the agents could navigate to the fridge first. In this regard, the semantic and functional prior could help the agents search for novel scenes.

Scenes	Target objects	Parent Objects
Kitchen	Toaster, Mug, Apple, CoffeeMahine, Bread	Fridge, TableTop, Sink, Shelf
Bedroom	Lamp, Pillow, AlarmClock	Shelf, Dresser, Desk, Drawer
Bathroom	Mirror, Towel, Mirror	Cabinet, Drawer, Toilet
Living Room	Vase, Painting, RemoteControl	Drawer, Shelf, Sofa

Table 4: Parent and target object list

Followed the work of Yang et al. (2019), we can construct the knowledge graphs using the objects relationships based on the image-captions of the Visual Genome dataset (Krishna et al., 2016). The visual relationship datatset - Visual Genome has more than 100K images and each of them are annotated with objects and their relationships. To develop the domain-specific relationship for ALFREAD dataset, we filter out those objects where both object nodes occur in the AI2-THOR environments for more than 3 times. Table 4 are some examples summarized by Qiu et al. (2020). These "parent objects" consist of the larger objects present in a room, which also happen to be spatially/semantically related to the target object. In total, we used 53 common object classes within the ALFREAD dataset (Qiu et al., 2020).

6.4 Experimental Results and Analysis

Table 5 shows the Task and Goal-Condition Success Rate for the proposed approaches. The results demonstrate that the annotation mixing and object rotation can improve the model performance while the feature mixing approach impairs the performance of the MOCA model. In particular, the approach with instruction enhancement achieved the highest performance on the val-seen and image augmentation with rotated object works best in val-unseen dataset.

Specifically, it is easy to see from the first two approaches to data augmentation that both the augmentation of semantic instructions and the augmentation of input images are significantly helpful for the training of the model and the perception of the simulated environment. Both approaches result in better performance than training the model directly on the original ALFRED dataset, suggesting that increasing the complexity of the training set is helpful for model training. In particular, we observe that the enhancement for semantic instructions makes the model perform better on the val-seen split, while the enhancement for input images

Model	Val-seen		Val-unseen	
	Task	Goal-Cond	Task	Goal-Cond
MOCA	10.59	17.74	2.01	7.78
+Inst	11.71	19.35	2.14	8.72
+Rotation	11.03	18.45	2.35	9.16
+Feature	9.96	16.61	1.97	7.34

Table 5: Task and Goal-Condition Success Rate for MOCA under different data augmentation. **+Inst**, **+Rotation** and **+Feature** here indicate **Semantic Instruction Set Augmentation**, **Object Augmentation** and **Mixing Different Feature Maps** mentioned in **Proposed Approach** part. For accelerate training, compared with original training configuration, we increase the batch size and learning rate. All values are percentages

makes the model perform better on the val-unseen split. We believe that the main reason for this is the nature of the val-unseen branch itself. val-unseen split has a clear distinction between scenes and objects in the input images, but there is no clear difference in the semantic instructions compared to the val-seen split. Therefore, when we focus on enhancing the images themselves, the trained model works better for the val-unseen split; and because we enhance the images themselves, we reduce the proportion of original images in the training set, thus reducing the number of training sessions for normal objects, resulting in the val-seen branch being less effective than the enhancement of semantic instructions.

In contrast to the first two methods of data enhancement, the results presented for the mixture of image features are not as satisfactory. We can see that the training results are not only inferior to the first two data augmentation methods, but even worse than the results of MOCA training on the original data. We speculate that mixing features from different sources within the same trajectory may lead to an overly complex dataset. We have already found in the memory mechanism section of the data analysis part that the complexity of the existing model is relatively small compared to the ALFRED dataset, so when the complexity of the dataset is further expanded, the model is not able to acquire the features properly in the limited training time, and thus does not get the desired results. Therefore, increasing the complexity of the model may be the next major direction to improve the model performance.

7 Limitations and Future Research

7.1 Auxiliary Information Enhancement

Visual data enhancement Currently, the ALFRED dataset only provides 300x300 RGB images for model input. Segmentation and depth images could help the agents better understand the surrounding environments and predict spatial relationships (see Fig. 19). Although the simulator provide instance segmentation masks and depth images by replaying each trajectory, the additional depth and mask from the simulator are not available on the testing dataset. Therefore, the proposed method takes RGB images as input and outputs instance segmentation and depth images using auxiliary losses.

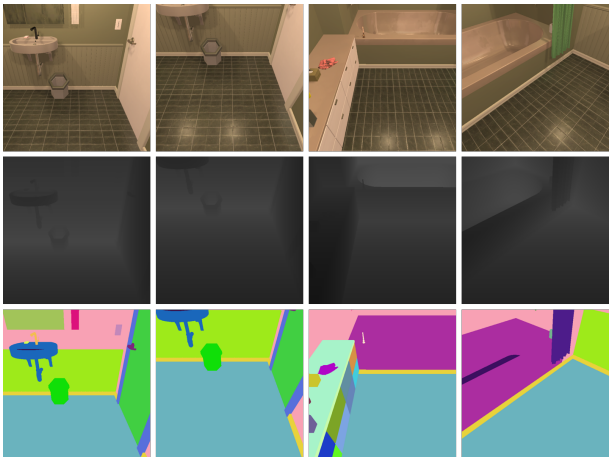


Figure 19: Data augumentation for input images

Modelling multi-task loss functions are essential for predicting semantic masks and depth information. We used a shared encoder and combined two task decoders to generate segmentation maps and depth images, respectively. The encoder is built on ResNet-18 as provided by the dataset. For the two task architecture, the loss function is computed as:

$$\mathcal{L}_{Total} = \lambda_{Seg}\mathcal{L}_{Seg} + \lambda_{Dep}\mathcal{L}_{Dep} \quad (3)$$

where $\mathcal{L}_{Seg}$ is instance segmentation loss by averaging pixel wise cross-entropy for each predicted label and ground truth label. $\mathcal{L}_{Dep}$ is depth estimation loss computed as mean absolute error between predicted depth and true depth pixels.

Given RGB images, the estimated segmentation maps, and estimated depth images, we can construct environment maps around the agents with back projection.

Scene priors enhancement Inspired by the human navigation behaviors in unseen scenes, we try to integrate the semantic/spatial/functional priors learned in previous scenarios (Yang et al., 2019). The developed scene priors can assist in encoding the spatial/functional relationships between different object classes and provide navigation guidance for novel objects in unseen environments. For example, to navigate to mugs in new environment, we may tend to look at cabinets. If we want to find fruits, we could open the fridge. Therefore, we integrate semantic priors in the task of visual navigation.

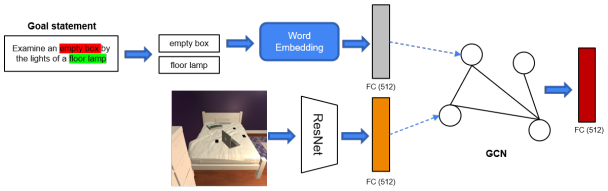


Figure 20: Scene priors for goal condition navigation

In the section 3.3, we first extract the scene priors from large-scale visual relationship dataset - Visual Genome dataset (Krishna et al., 2016). Then we represented the domain-specific priors for ALFREAD dataset in the form of graph representation and applied the Graph Conventional Networks (GCNs) to calculate the relational features with a binary adjacency matrix (Yang et al., 2019). Fig. 20 illustrates our idea. The input of the GCNs node are joint embedding of visual features and semantic statement. The target of goal statement is first extracted and then embedded to 512-d feature. Similarly, the visual information of current frame is also represented in 512-d features. The GCNs take these two type of features as input and resulting in 512-d features.The joint feature is further fed into the moca framework (Singh et al., 2020).

7.2 Update Training Mechanism

Both MOCA and baseline modal adopt imitation learning to address the problem. Although imitation learning would be a wonderful method to solve such a problem in a supervised way, its defect is also obvious: the input distribution may vary between the training phase and validation phase. For all input images in the training phase, they are all scenes along the target trajectory, which means their distribution would be consistent with the content of task and goal-conditions, otherwise the instructions would not be accomplished. For input images in validation and testing phases, they are automatically generated by the simulator based on the previous predicted action. If the predicted action is incorrect, the input image would also be unexpected, and its distribution would not be similar to the textual instruction. Fig. 21 illustrates our idea.

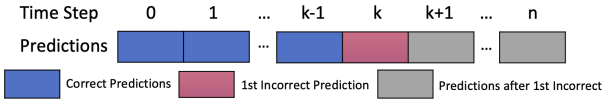


Figure 21: This figure shows action predictions based on an arbitrary trajectory. After the first incorrect action prediction came up (red rectangle in k-th step), the input images of all follow-up steps (grey rectangles) might not be along with the trajectory, making it impossible to return correct answer based on textual instruction (e.g. need to walk straightly in instruction, but facing a wall in image).

From Fig.21, we can find that step k is critical. For each step before the k-th step, since its former action predicted corrected, the input image for this time step would also correct, which is consistent with the training phase. Therefore, under nowadays scenario, it is perfectly feasible to optimize the model in the training phase so that the value of k in the testing phase is as close to n as possible. Moreover, since the image input after time k may be irrelevant to instruction, the model would be hard to predict well after time-step k under the current training mechanism. Therefore, for the current training mechanism, focusing on how to push k as near n as possible is one possible way to achieve a better result in validation and testing phases.

To achieve the goal, we should prioritize the earlier tasks to get them done better, and one way to do that is to improve the weight of loss coming from the early time step and maintain the total loss value. We present the Index Loss as follow:

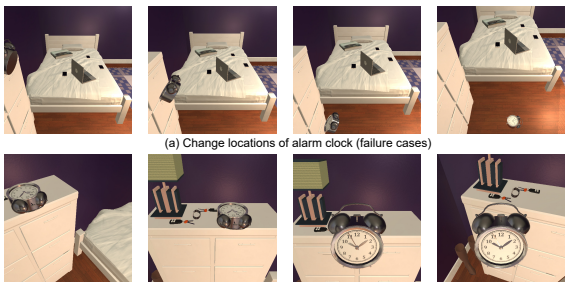
$$\mathcal{L}_{Total} = \Sigma_{t=0}^{n} \frac{n \cdot (n-t)}{\Sigma_{i=0}^{n} i} \cdot \mathcal{L}_t \qquad (4)$$

where n is the total number of time steps.

However, the Goal-Condition Success Rate on the valid unseen data of the model trained with the Index loss is 8.63 %, which is less than the result of MOCA (13.40%). Furthermore, the Goal-Condition Success Rate on the valid seen of the

model trained with the Index loss is 21.24%, which is also less than the result of MOCA (28.50%).

7.3 Diminish the distribution inconsistency between training and validation data



(a) Change locations of alarm clock (failure cases)

(b) Change rotations of alarm clock (success cases)

Figure 22: Failure cases of changing locations

As shown in the Figure 2, the authors found out that almost all interactable objects are located in the center of training set, however the distributions of the object centers are different in the validation phase. Therefore, three possible methods could be taken into considerations: (1) Shuffle the interactable objects center in training set; (2) Re-clip the image to move the centers of the interactable objects; (3) Image boundary extensions or generations by generative adversarial network (GAN) or other images from simulator.

Get images from multiple viewing angles The agent's performance in navigation is a bottleneck to the overall performance (Storks et al., 2021). Because in all ALFRED tasks, the agent needs to navigate to a new location. If the agent fails to arrive the correct locations, consequently it will not find the required objects. Furthermore, with the pre-trained model in ALFRED, the subgoal success rate of the navigation is 61.9 %, relatively low compared to other types of subgoal success rate, such as the cool or toggle an object with 90 % (Storks et al., 2021). In order to enhance the mapping between the visual understanding and language grounding, additional inputs of agent's visual views are collected. At each time step, the agent collects visual inputs for eight panoramic view angles, and applies the trained object detection model to them.

8 Conclusion

We explore the problem of interactive instruction following on the ALFRED benchmark. To address individual challenges of OOD data and biased the scene and object distributions, we propose two solutions based on the MOCA framework: data diversity enrichment and data relationship extraction. We first applied a series targeted data augmentation method to improve the robustness of the model under different scenes. In addition, we used a Graph Neural Network to capture the relationship between objects, which can help the model identify objects that out of the distribution. The results demonstrate that the annotation mixing and object rotation can efficiently improve the model performance on ALFRED dataset.

Model	Validation				Test			
	Seen		*Unseen*		*Seen*		*Unseen*	
	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond
Singh et al. (2020)	19.15 (13.60)	28.50 (22.30)	3.78 (2.00)	13.40 (8.30)	22.05 (15.10)	28.29 (22.05)	5.30 (2.72)	14.28 (9.99)
Input Ablations								
No Language	2.68 (1.30)	9.58 (6.30)	0.00 (0.00)	6.65 (3.00)	1.04 (0.77)	7.00 (4.70)	0.26 (0.05)	6.68 (3.63)
No Vision	0.24 (0.10)	5.88 (4.70)	0.00 (0.00)	7.17 (6.10)	0.00 (0.00)	4.36 (3.28)	0.39 (0.13)	7.46 (4.37)
Goal-Only	5.37 (2.70)	14.32 (10.00)	0.24 (0.10)	8.49 (4.80)	3.65 (1.99)	10.55 (7.68)	1.37 (0.47)	9.03 (5.40)
Instructions-Only	4.15 (2.60)	12.75 (9.60)	0.49 (0.30)	7.45 (5.30)	5.81 (3.60)	11.49 (8.99)	0.59 (0.25)	7.44 (4.46)

Table 6: Input ablations for MOCA. All values are percentages.

Components				Validation-Seen		Validation-Unseen	
Decoupling Policy and Perception	Object-centric Mask Prediction	Dynamic Filters	Color Swap	Task	Goal-Cond	Task	Goal-Cond
x	x	x	x	19.15 (13.60)	28.50 (22.30)	3.78 (2.00)	13.40 (8.30)
x	x	x		16.71 (10.60)	23.71 (16.00)	2.80 (1.20)	12.03 (6.60)
x	x			13.66 (9.20)	23.99 (17.40)	2.31 (1.40)	10.57 (7.50)
x		x		3.78 (2.10)	10.91 (7.40)	0.37 (0.30)	7.55 (4.30)
	x	x		11.34 (6.30)	20.10 (12.90)	2.07 (1.10)	12.74 (7.30)

Table 7: Model ablations for MOCA. All values are percentages.

Model		Validation-Seen		Validation-Unseen	
Instance-Association in Time (IAT)	Obstruction Detection	Task	Goal-Cond	Task	Goal-Cond
x	x	19.15 (13.60)	28.50 (22.30)	3.78 (2.00)	13.40 (8.30)
x		10.61 (7.10)	20.34 (15.60)	1.46 (0.80)	10.52 (6.60)
	x	16.10 (11.40)	23.80 (19.30)	3.53 (1.80)	12.41 (8.40)

Table 8: Ablations of Instance-Association in Time (IAT) and Obstruction Detection for MOCA. All values are percentages.

Proposed Metrics	**Value Interval**	**Validation**			
		Seen		*Unseen*	
		Task	Goal-Cond	Task	Goal-Cond
Number of Instructions	[4, 8)	15.85(19.19)	34.18(28.10)	2.56(1.98)	16.77(10.53)
	[8, 12)	2.32(9.98)	27.91(23.45)	0.24(1.55)	7.50(5.30)
	[12, 16]	0.61(3.25)	9.41(7.26)	0(0)	1.45(1.23)
Total Length of Instructions	[20, 70)	11.46(16.00)	32.16(25.11)	2.31(2.16)	17.61(10.64)
	[70, 120)	6.46(14.78)	27.69(22.55)	1.10(1.42)	11.64(7.85)
	[120, 170)	0.85(5.72)	20.57(17.42)	0(0)	1.22(0.46)
	[170, 220]	0(0)	5.56(4.09)	-	-
Number of Objects	[13, 22)	18	28.3334	5.0333	10.5111
	[23, 31)	6.8333	9.6944	0.5	7.9833
	[32, 40)	9.6667	14.3583	0	0
	[41, 49)	12.25	18.1139	2.1667	9.2333
	[49, 60)	2.6667	6.2111	0.25	6.3611
Average Pair-Wise Distance	[1, 1.57)	3.3333	5	4.7	8.0111
	[1.58, 2.15)	20.3333	30.025	2	14.1222
	[2.16, 2.73)	16.4	25.925	0.5833	7.4611
	[2.74, 3.31)	6.6833	10.8278	0.5	4.55
	[3.32, 3.89)	2.6667	4.9444	0	0

Table 9: Task and Goal-Condition Success Rate vs Proposed metrics. We will use this table to record each approach separately. For each metric, the corresponding path weighted metrics are given in (parentheses). All values are percentages. * indicates the data will be replenished later.

References

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. 2016. Concrete problems in AI safety. *CoRR*, abs/1606.06565.

Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian D. Reid, Stephen Gould, and Anton van den Hengel. 2017. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *CoRR*, abs/1711.07280.

Ding-Jie Chen, Songhao Jia, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. 2019a. See-through-text grouping for referring image segmentation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 7453–7462.

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

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. *CoRR*, abs/1806.02724.

Daniel Gordon, Aniruddha Kembhavi, Mohammad Rastegari, Joseph Redmon, Dieter Fox, and Ali Farhadi. 2017. IQA: visual question answering in interactive environments. *CoRR*, abs/1712.03316.

Weituo Hao, Chunyuan Li, Xiujun Li, Lawrence Carin, and Jianfeng Gao. 2020. Towards learning a generic agent for vision-and-language navigation via pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13137–13146.

Ting-I Hsieh, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. 2019a. One-shot object detection with co-attention and co-excitation.

Ting-I Hsieh, Yi-Chen Lo, Hwann-Tzong Chen, and Tyng-Luh Liu. 2019b. One-shot object detection with co-attention and co-excitation. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 2721–2730.

Ronghang Hu, Marcus Rohrbach, and Trevor Darrell. 2016. Segmentation from natural language expressions.

Zhiwei Hu, Guang Feng, Jiayu Sun, Lihe Zhang, and Huchuan Lu. 2020. Bi-directional relationship inferring network for referring image segmentation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 4423–4432. IEEE.

Pin-Hao Huang, Han-Hung Lee, Hwann-Tzong Chen, and Tyng-Luh Liu. 2021. Text-guided graph neural networks for referring 3d instance segmentation.

Shaofei Huang, Tianrui Hui, Si Liu, Guanbin Li, Yunchao Wei, Jizhong Han, Luoqi Liu, and Bo Li. 2020. Referring image segmentation via cross-modal progressive comprehension.

Tianrui Hui, Si Liu, Shaofei Huang, Guanbin Li, Sansi Yu, Faxi Zhang, and Jizhong Han. 2020. Linguistic structure guided context modeling for referring image segmentation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part X*, volume 12355 of *Lecture Notes in Computer Science*, pages 59–75. Springer.

Peter A. Jansen. 2020. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions.

Anna Khoreva, Anna Rohrbach, and Bernt Schiele. 2018. Video object segmentation with language referring expressions. *CoRR*, abs/1803.08006.

R. Krishna, Yuke Zhu, O. Groth, J. Johnson, K. Hata, J. Kravitz, Stephanie Chen, Yannis Kalantidis, L. Li, D. Shamma, Michael S. Bernstein, and Li Fei-Fei. 2016. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123:32–73.

Ruiyu Li, Kai-Can Li, Yi-Chun Kuo, Michelle Shu, Xiaojuan Qi, Xiaoyong Shen, and Jiaya Jia. 2018. Referring image segmentation via recurrent refinement networks. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 5745–5753. IEEE Computer Society.

Chenxi Liu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, and Alan L. Yuille. 2017. Recurrent multimodal interaction for referring image segmentation. *CoRR*, abs/1703.07939.

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 *Proceedings of the International Conference on Learning Representations (ICLR)*.

Chih-Yao Ma, Zuxuan Wu, Ghassan AlRegib, Caiming Xiong, and Zsolt Kira. 2019b. The regretful agent: Heuristic-aided navigation through progress estimation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Hongyuan Mei, Mohit Bansal, and Matthew R. Walter. 2015. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. *CoRR*, abs/1506.04089.

Mahdi Kazemi Moghaddam, Qi Wu, Ehsan Abbasnejad, and Javen Qinfeng Shi. 2020. Optimistic agent: Accurate graph-based value estimation for more successful visual navigation.

Tushar Nagarajan and Kristen Grauman. 2018. Attributes as operators: Factorizing unseen attribute-object compositions. In *Proceedings of the European Conference on Computer Vision (ECCV)*.

Anton Osokin, Denis Sumin, and Vasily Lomakin. 2020. Os2d: One-stage one-shot object detection by matching anchor features.

Yiding Qiu, Anwesan Pal, and Henrik I Christensen. 2020. Learning hierarchical relationships for object-goal navigation. In *The Conference on Robotic Learning*, Boston, MA.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597.

Seonguk Seo, Joon-Young Lee, and Bohyung Han. 2020. URVOS: unified referring video object segmentation network with a large-scale benchmark. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part XV*, volume 12360 of *Lecture Notes in Computer Science*, pages 208–223. Springer.

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

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020b. Alfred: A benchmark for interpreting grounded instructions for everyday tasks.

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

Shane Storks, Qiaozi Gao, Govind Thattai, and Gokhan Tur. 2021. Are we there yet? learning to localize in embodied instruction following.

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

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the CVF/IEEE Conference on Computer Vision and Pattern Recognition*, Long Beach, CA, USA. CVF/IEEE.

Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. 2018. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. *CoRR*, abs/1803.07729.

Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. 2020a. The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation. In *Proceedings of the Conference on Robot Learning*, volume 100 of *Proceedings of Machine Learning Research*, pages 1369–1378. PMLR.

Christopher Xie, Yu Xiang, Arsalan Mousavian, and Dieter Fox. 2020b. Unseen object instance segmentation for robotic environments.

Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. 2018. Visual semantic navigation using scene priors.

Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. 2019. Visual semantic navigation using scene priors. In *Proceedings of (ICLR) International Conference on Learning Representations*.

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. *CoRR*, abs/1801.08186.

Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. 2017. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3357–3364.

A Appendix

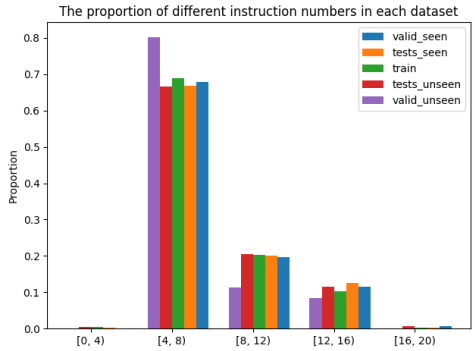


Figure 23: Instruction number in expert demonstration

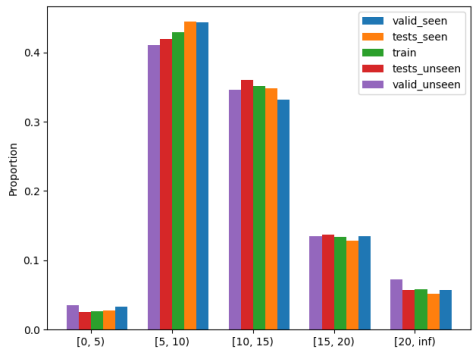


Figure 24: Distribution of word number in step-by-step instruction

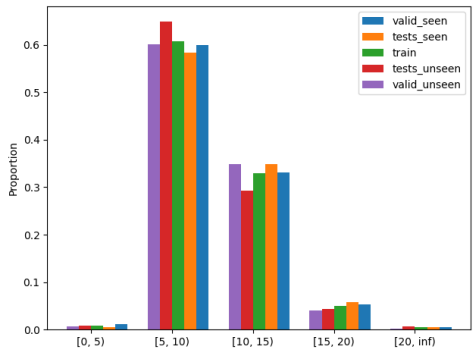


Figure 25: Distribution of word number in goal instruction

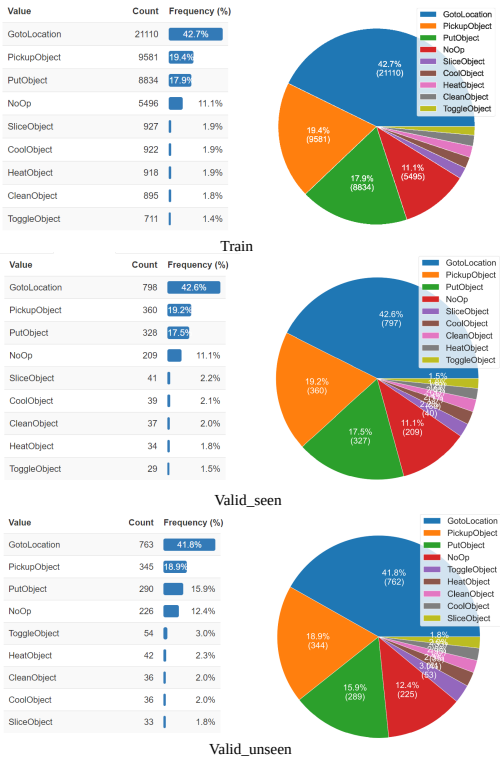


Figure 26: The distributions of the actions in the high level actions

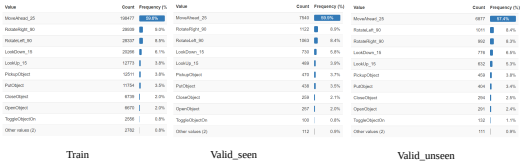


Figure 27: The distributions of the actions in the low level actions

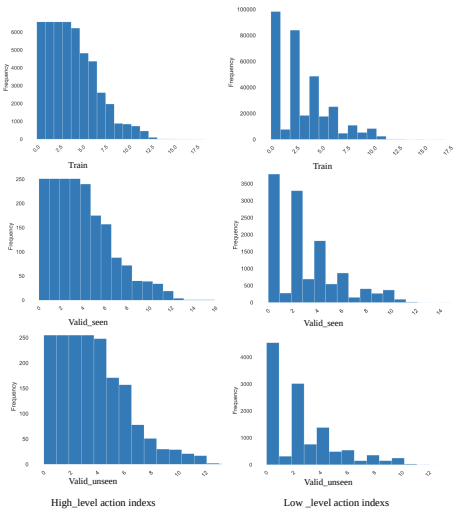


Figure 28: The comparisons of the indexes of high level and low level actions

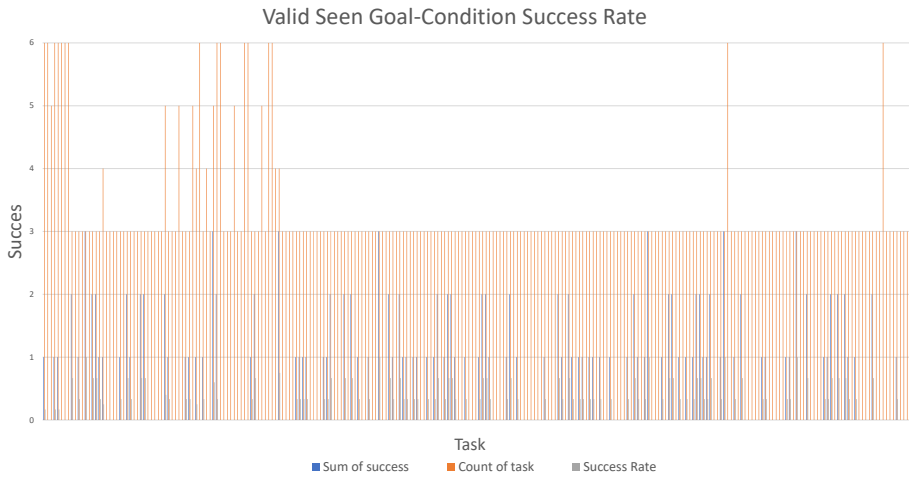


Figure 29: Valid Seen Task Success Rate

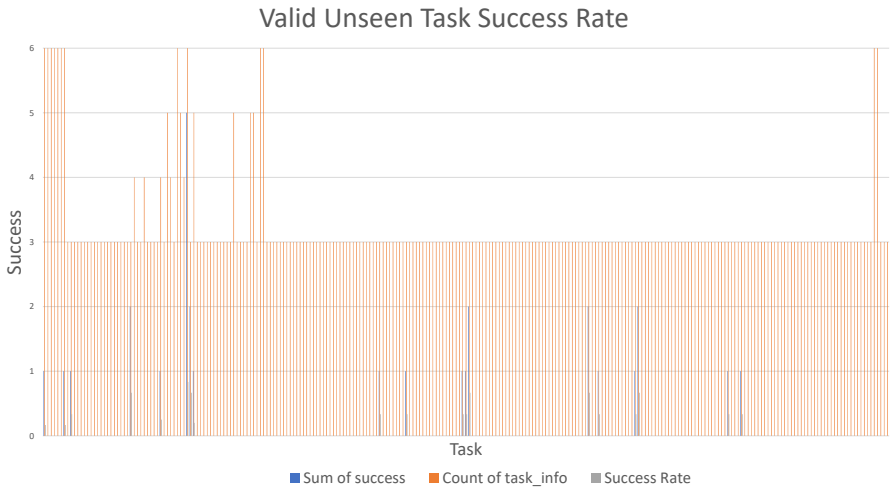


Figure 30: Valid Unseen Task Success Rate

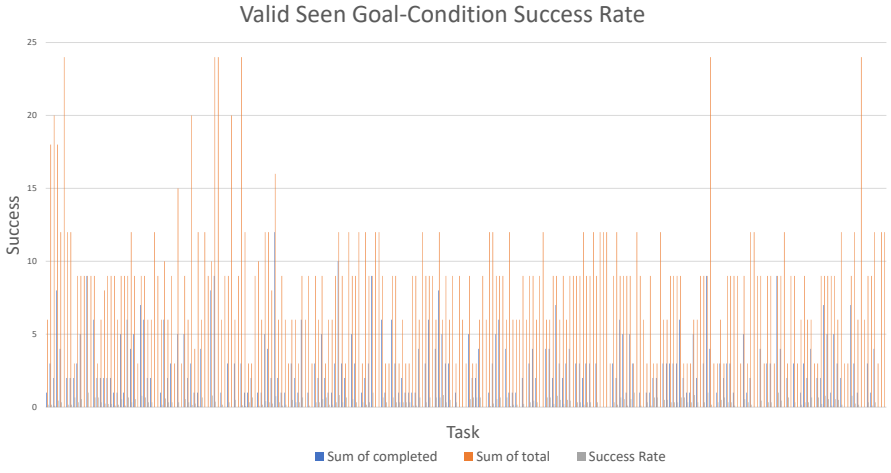


Figure 31: Valid Seen Goal-Condition Success Rate

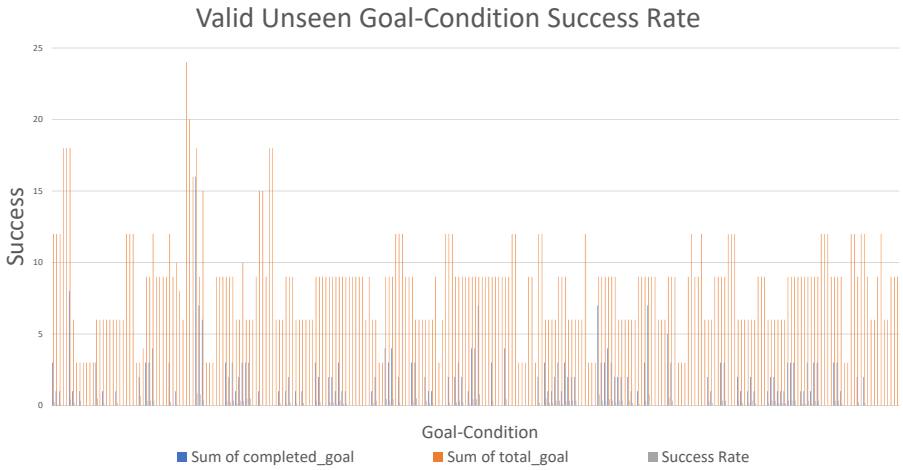
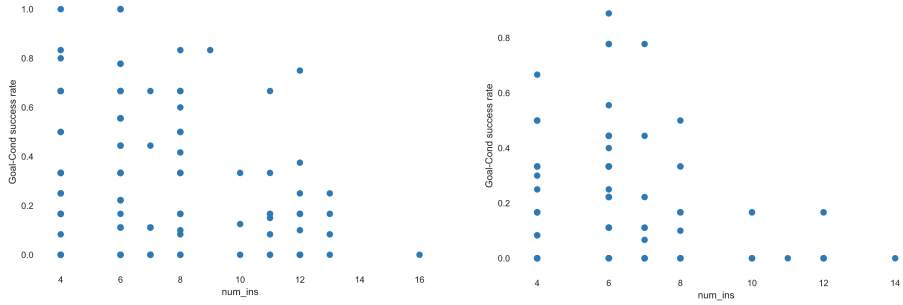
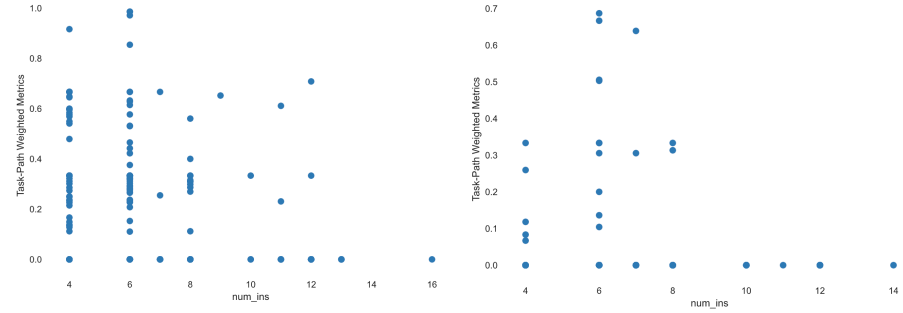


Figure 32: Valid Unseen Goal-Condition Success Rate



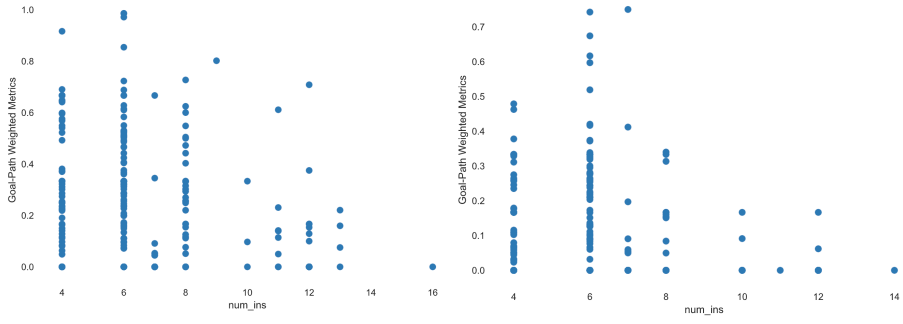
(a) Validation seen dataset (b) Validation unseen dataset

Figure 33: The goal condition success rate with different number of instructions



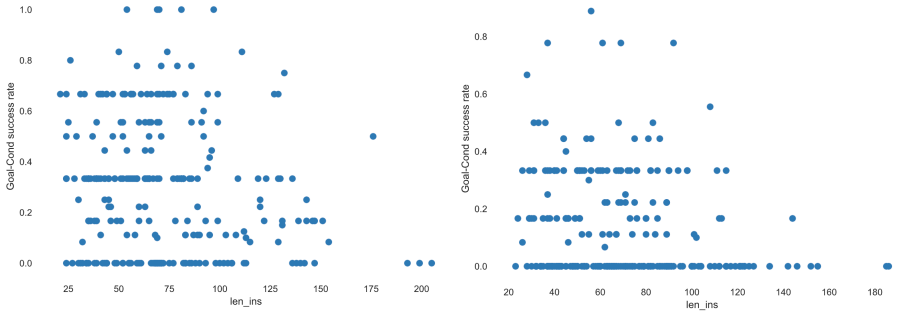
(a) Validation seen dataset (b) Validation unseen dataset

Figure 34: The Path-length-weighted Success Rate with different length of instructions



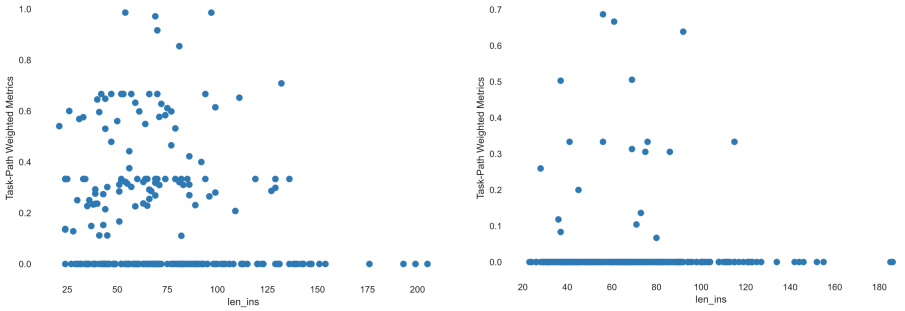
(a) Validation seen dataset (b) Validation unseen dataset

Figure 35: The Goal Condition Path-length-weighted Success Rate with different length of instructions



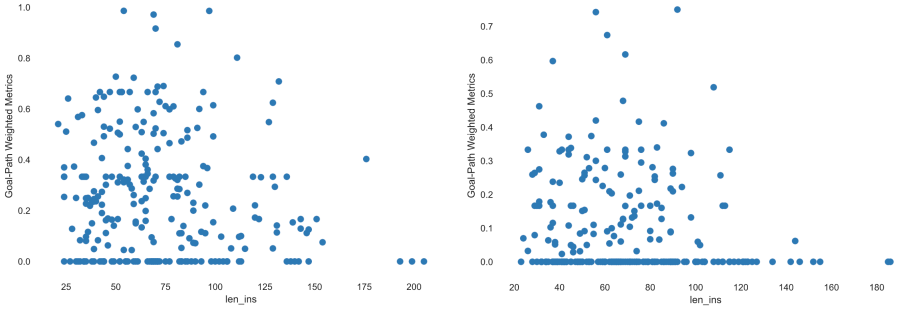
(a) Validation seen dataset (b) Validation unseen dataset

Figure 36: The goal condition success rate with different length of instructions



(a) Validation seen dataset (b) Validation unseen dataset

Figure 37: The Path-length-weighted Success Rate with different length of instructions



(a) Validation seen dataset (b) Validation unseen dataset

Figure 38: The Goal Condition Path-length-weighted Success Rate with different length of instructions