11-777 Spring 2021 Class Project
Action Feasibility in the ALFRED Task

Christian Deverall* **Jingyuan Li** * **Artidoro Pagnoni** *
{cdeveral, jingyua4, apagnoni}@andrew.cmu.edu

Abstract

The task of navigating a room to complete household tasks defined with natural language remains challenging for AI systems. In this project, we focus on the ALFRED dataset. We perform an in-depth analysis of the dataset, evaluate two competitive models, and identify their limitations. We find that even the best of these two models, MOCA, outputs a significant number of infeasible actions. We evaluate whether providing MOCA with a better action feasibility module improves the overall performance by reducing the infeasible actions taken by the agent.

1 Introduction

Understanding and following task-oriented instructions are natural and straightforward skills for human beings. This remains to a great challenge for robotic systems due to the difficulties in both visual and linguistic understanding. In this project we focus on the ALFRED dataset (Shridhar et al., 2020a) to explore the navigation and interaction of robot agents. The high-level task for the ALFRED dataset is to predict the correct sequence of household actions given textual instructions and visual feedback. It is unique in that the actions are long, compositional, and non-reversible. An example of this is that the agent must pick up a knife to slice a potato. Moreover, once a potato is sliced, it cannot be put back together. Actions consist of navigation within a room or interaction with an object. For interaction, a pixel-wise mask must be predicted whereby the object with the highest intersection-over-union score with the interaction mask is acted upon. The textual instructions come in the form of both high-level task descriptions and step-by-step commands. The visual feedback is ego-centric and the next image is provided after each action is performed. Compared to previous datasets, the ALFRED dataset contains a diverse range of tasks. Specifically, there are 7 high-level task types parameterized by a combination of 84 object classes and 120 household scenes.

In the ALFRED world, actions are not always valid. This may be due to a variety of reasons, for example, objects could be blocking the movement of the agent, or an object the agent may want to interact with may not be present in the image. While it is natural for humans to avoid taking such mistaken actions, it is challenging for the trained AI agents since they are not aware of the consequence of an action. An intuitive explanation for such inability is that the training paths don't involve any examples of actions that are not feasible. The agent is only provided with indirect information about action feasibility: it is guaranteed that all actions in the training trajectories are feasible and has to deduce by lack of examples including some actions that are not feasible. As a result, the agent tends to make mistakes when it's under new environments since they have no knowledge that tells them making mistakes leads to failure. As a reference, we observed that 28% and 40% (for seen and unseen scenes respectively) of trajectories are interrupted in MOCA (Singh et al., 2020) because they reach the limit of 10 errors. This indicates that the presence of many invalid actions causes a large portion of failures.

To help the agent make better decisions on action selection, we identify modeling action feasibility as an interesting avenue for improvement. This direction is particularly challenging in the context of Seq2Seq models such as those that have been employed in the ALFRED task since it requires the agent to not only understand correct paths but also understand incorrect actions. To achieve this goal, we propose to build methods that inform the feasibility of an action. Building such a model re-

*Everyone Contributed Equally – Alphabetical order

split	mean	std	min	max
low-level train	49.78	24.78	5	234
low-level val seen	46.98	20.22	13	137
low-level val unseen	50.12	25.61	12	165
high-level train	6.51	2.49	2	19
high-level val seen	6.47	2.62	3	16
high-level val unseen	6.16	2.33	3	14

Table 1: Statistics on the number of low-level and high-level actions for the splits of the data.

quires extra data that comprehensively describes the action spaces to an extent that is greater than what is given in the original Alfred dataset. To reach this goal, we collect a large set of image-action pairs that describe the action feasibility in each location in a room. With the data, we then trained a module to predict the feasibility of action. While simple, we verify that this model is effective in predicting the action feasibility in both seen and unseen spaces, demonstrating the feasibility of our hypothesis. Besides, we also tested the potential by informing the agent about action feasibility, where a clear improvement is demonstrated in tasks performed in unseen spaces.

2 Task Background

In this section, we present an analysis into the dataset and two baseline models. Our objective is to get a sense of the types of demonstrations that are present in the ALFRED dataset as well as examine the weaknesses and strengths of the current state-of-the-art models.

2.1 High-level Analysis

Action-Space In the ALFRED dataset, tasks are described by low-level and high-level actions. In this section, we examine the patterns behind these actions and the objects which are acted upon. In order to gauge the size of tasks and the granularity of actions, we provide statistics about the number of low-level and high-level actions per task in Table 1.

Low-level actions are often repeated in order to fully complete a high-level goal. For example, numerous fixed-distance movement actions are required to travel from one end of the kitchen to the other. This repetition pattern is relevant to action prediction because for highly repeated actions, it is likely that the next action is the exact same as the current one. In Figure 1, it can be seen that by far the most duplicated action is the ”MoveAhead” operation encompassing approximately 44% of all actions.

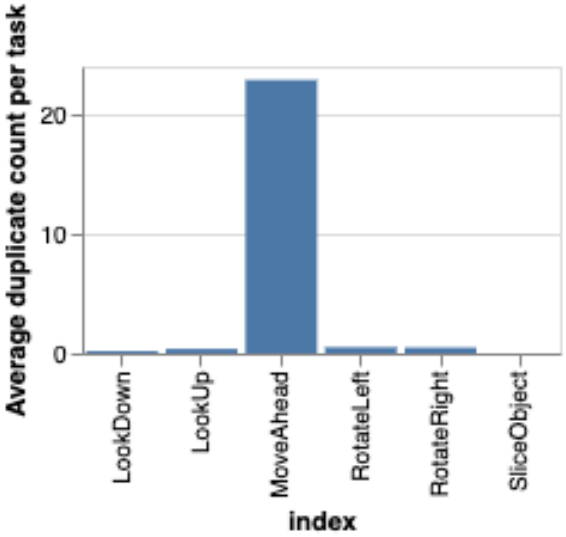

Figure 1: Average number of duplicate actions per task

split	mean	std	min	max
train	6.72	2.49	3	19
val seen	6.79	2.73	4	16
val unseen	6.26	2.33	4	14

Table 2: Statistics on the number of sentence instructions (sub-goals) for the splits of the data.

Instruction-based Analysis There are two types of instructions: the task description and the detailed instructions. The task description is generally a single sentence outlining the goal of the task while the detailed instructions break the task down into sub-goals and provide a sentence instruction for each sub-goal. We measure statistics on the number of sub-goals for each task in Table 2. We observe that there are up to 19 distinct sub-goals in each trajectory and that on average there are 6.7 sub-goals. The tasks are not simple as they require a minimum of three sub-goals.

2.2 Analysis of Baselines

The baselines we evaluate are the “Modular Object-Centric Approach” (MOCA) (Singh et al., 2020) model and the “Sequence-to-Sequence with Progress Monitor” model (baseline of the ALFRED proposal paper) (Shridhar et al., 2020a). We provide comparisons in two settings: “seen” and “unseen”. “Seen” refers to when all the test tasks take place in rooms that have been previously seen in

the training set.

Feasible Action Analysis In the ALFRED world, actions are not always feasible. This might be due to a variety of reasons, for example, objects could be blocking the movement of the agent, or an object the agent wants to interact with might not be present in the image. In Table 3, we report statistics on the reason why certain actions are considered invalid for the MOCA model. Our first observation is that in the seen scenes the vast majority (80%) of invalid actions are from two categories: the agent is unable to move (blocked) and the object was visible but not located correctly. In the unseen scenes this trend in the errors is even more accentuated with 87% of the errors being from these two categories (47% for the agent being blocked, and 40% for the object not being located correctly). These results indicate possible areas for improvement.

Furthermore, in the second part of the Table 3, we observe that 28% and 40% (for seen and unseen scenes respectively) of trajectories are interrupted because they reach the limit of 10 errors. This indicates that the presence of many invalid actions causes a large portion of failures. The two major areas for improvement in this direction are: 1. spatial detection of the validity of an action and 2. visual detection of objects. The fact that the trajectories predicted by the MOCA model contain so many infeasible actions provides the central motivation for our research, to provide the model with an awareness of whether predicted actions are feasible or infeasible.

3 Related Work

In this section, we investigate existing work related to our project. We focus on related techniques and tasks. The supporting techniques discussed here are (1) program synthesis/induction, (2) language-image grounding. The related task discussed here is multimodal embodied interaction.

3.1 Program Synthesis

One challenge of dealing with instructions expressed in natural language is that natural language is noisy. The same instructions can be expressed through different paraphrases and depending on the context the same instructions might describe different actions. The task of parsing natural language instructions into a representation that has a deterministic execution is generally referred to as program synthesis or inductive programming (Summers, 1977; Muggleton and De Raedt, 1994).

Neural Code Generation Previous work in program synthesis has explored the task of parsing natural language descriptions into source code written in a general-purpose programming language. Yin and Neubig (2017) proposes a neural architecture which uses a grammar to explicitly capture the syntax of the target language in an Abstract Syntax Tree. Dong and Lapata (2018) use a different approach which is based on coarse-to-fine decoding and decomposes the parsing process into two stages: a rough sketch of the meaning of input utterance is first generated, and it is further refined by filling in the missing information at a second stage.

Compositionality Instructions and questions are generally compositional. Being able to capture the compositional linguistic structure of natural language questions can help create and combine specialized modules. Andreas et al. (2016) uses a natural language parser to dynamically lay out a network composed of reusable modules that are jointly trained. The CLEVR dataset (Johnson et al., 2017) was proposed to test model's ability to perform compositional reasoning such as recognizing novel attribute combinations. In this sense it is similar to the ALFRED Shridhar et al. (2020a) instructions which test the performance of an agent in new action-object-scene combinations. Santoro et al. (2017) propose Relation Networks to augment CNNs and perform relational reasoning on the CLEVR dataset.

Semantic Parsing The problem of code program synthesis is related to semantic parsing where a natural language sentence is mapped into a cpmplete, formal meaning representation (Mooney, 2007). Several approaches have been proposed to leverage unlabeled data for the task of semantic parsing using question-answering and paraphrase models (Berant et al., 2013; Berant and Liang, 2014) .

3.2 Image Understanding

Natural Language Object Retrieval Traditional image recognition tasks such as object detection and semantic segmentation have performed extremely well when the textual input is highly defined. Often bounding boxes and segmentation masks are directly associated with a tag or label that defines the object. One limitation of this method is

	Seen		Unseen	
	Failure	Success	Failure	Success
Agent blocked	1514	139	2639	7
Agent state not allowed	127	2	79	0
Object visible but not located correctly	1569	28	2191	3
Object not found in scene	96	0	202	4
Object property not allowed	170	9	102	2
Others	163	0	124	0
Target state not allowed	105	0	46	0
No valid target	40	0	25	0
Object not visible	109	3	153	0
Object state not allowed	21	0	18	2
Frequency of error limit (10 errors) reached	0.28		0.40	
Number of Errors (when less then 10 errors)	4.44		4.91	
Average number of errors	6.03	1.05	6.99	0.78
Average length of sequences of errors	2.954	1.19	3.57	1.17

Table 3: Analysis of types of error messages returned by Thor for invalid actions. We compare failed and successful tasks as well as unseen and seen scenes. The first part of the table contains counts of errors of different types across the validation dataset. The second part computes statistics about the number of errors per trajectory.

that it cannot not leverage the complex structural information inherent to natural language. In (Xiao et al., 2017), natural language object retrieval is performed on a relatively small dataset using weak supervision. More specifically, the authors represent sentences as a parse tree, which enables learning at several levels of the tree for the same image caption. This method outperformed baselines that did not consider linguistic structure on the MS COCO and Visual Genome datasets. For the same task, (Wu et al., 2017) uses deep reinforcement learning to iteratively reshape a bounding box to localize the object.

Visual QA In contrast to the previous papers which only localize objects within images, the visual question answering task involves answering a more complex question about the image. In (Antol et al., 2015), which initially proposed the VQA task, the best performing model uses an LSTM to separately encode the text while a CNN encodes the image. (Yang et al., 2015) improves on this by introducing stacked attention networks, which iteratively queries the image. By repeatedly refining queries that combine both image and textual data, the latter attention layers can focus on the most relevant parts of the image. Recently, papers such as (Wu et al., 2019) have provided innovations in the way that the image and textual encodings are fused together. Typical models take the product between both encodings, however this paper proposes the idea that the product of the difference between feature elements is a superior fusion.

3.3 Multimodal Embodied Interaction

The ability to understand and follow instructions is of great importance for robotic systems that aims to assist human in the real world. For its practicability, methods allowing the robots to follow instructions and complete specific goals have attracted great attention. One of the related tasks is called visual language navigation, which aims to generate sequences of actions from human instructions that guide the agent to complete tasks in specific scenes. The Room-to-Room dataset was proposed to simulate the real-world circumstance where instructions, scene images, and corresponding actions are provided, to accelerate the development of visual language navigation (Anderson et al., 2018). To deal with the visual language navigation task, the multi-modal mapping between action sequence and instructions is built, which not only allows the agent to plan before taking actions but also enables data augmentation which improves the robustness of the model (Fried et al., 2018). To allow the learning from environmental exploration and failure experiences, reinforcement learning techniques are also used in the navigation task (Ma et al., 2019; Wang et al., 2018). Besides per-

formance improvements, many related works are also working on addressing specific assumptions made in the Room-to-Room dataset, including the discrete space assumption (Krantz et al., 2020), and the known environment assumption (tan). While the Room-to-Room dataset has modeled VLN tasks comprehensively, one limitation that cannot be addressed is the lack of interaction which is one of the eventual goals of real-world robotic systems. To take a step forward, the ALFRED dataset is proposed (Shridhar et al., 2020b), with both navigation and interaction taken into consideration. Due to the requirement of interaction, the agent needs not only to decide the actions to take but also to figure out which object in the environment should be acted on, introducing extra complexity to the already hard problem. To address the issue, MOCA (Singh et al., 2020) with modules dealing with visual perception and action policy was proposed, to make the task feasible by dividing different components of the targets. The above describes works and tasks related to our project. In this project, we are intended to work on the exploration of and interaction with environment by the agent.

4 Proposed Approach

After performing an analysis of the two baselines "Modular Object-Centric Approach" (MOCA) (Singh et al., 2020) model and "Sequence-to-Sequence with Progress Monitor" model (baseline of the ALFRED proposal paper) (Shridhar et al., 2020a), we identify some areas that could lead to performance improvements. In this report we explain in detail what specific limitations were of greatest interest to our team in the context of a multimodal machine learning class, and how we plan to mitigate these limitations for the MOCA model.

The proposed approach involves improving the model's understanding of its surrounding, in particular, its understanding of the feasibility of actions. We also propose two other approaches which are meant to improve the quality of the predicted mask which was one of the sources of action infeasibility.

4.1 Action Feasibility Auxiliary Prediction

Along with predicting the next action in MOCA, we propose an auxiliary prediction task. The model predicts which actions are feasible at each time step along the golden trajectories provided in the training set. We gather the data on which actions are feasible at each time step of the golden trajectories by querying the simulator offline before training. The current prediction task only requires the agent to predict the next best action to undertake and does not provide any clue to the model about any other differences in the available actions. Our hypothesis is that with the auxiliary action feasibility prediction task, the agent would develop an understanding of the available actions beyond whether they are the best next step but whether they are also feasible.

To achieve this goal, we define an auxiliary loss function that tells the agent about feasible and infeasible actions. In specific, in the Action Policy Module of MOCA, we add an extra linear layer before the action output, denoted as $\hat{A}$, representing the feasibility of the action. We want the agent to learn from the environment which action is feasible at each time step. And each element in $\hat{A}$ is activated by sigmoid function σ. We train this additional layer using focal loss (Lin et al., 2017):

$$Loss = -\frac{1}{n_a}\sum_{i=1}^{n_a} a^i(1-\hat{a}^i)^\gamma log(\hat{a}^i)+(1-a^i)\hat{a}^{i\gamma} log(1-\hat{a}^i) \tag{1}$$

where γ is a tunable parameter. Here we use focal loss instead of conventional binary cross entropy loss because most actions are feasible and only a minority of actions are not feasible. Focal loss helps to reduce the influence of imbalanced samples.

4.2 Action Feasibility Mask

We propose a second stand-alone model that only uses the visual input and the interaction mask to predict the feasibility of an action. This model can be trained in a supervised fashion using the simulator sampling different locations and querying the feasibility of actions at those locations.

Since the second model requires only visual input and action feasibility output independent of the previous step, the model can be trained offline. To train this model, we need a group of images $I_1, I_2, ..., I_t$ pre-collected from the simulator, and the corresponding action feasibility labels $F_1, F_2, ..., F_t$ at the location where the corresponding images are collected. The action feasibility in F_n is 1 if the action is either a feasible navigational action or a feasible interactive action that acts on any objects in I_n, otherwise, it's 0. Given a visual feature extractor (e.g. ResNet, DenseNet) $\Phi(\emptyset = cdot)$, we assume the output in the last

layer of the model to be F_{logits} of size n_a, where n_a is the number of action. F_{logits} is the score of action feasibility without any activation functions (i.e. it's the output from a linear layer). Then we generate the eventual action feasibility mask using the element-wise sigmoid function σ, i.e.

$$\hat{F} = \sigma(F_{logits}) \tag{2}$$

Then we use the binary cross-entropy loss on each action for calculating the loss function, i.e.

$$Loss = -\frac{1}{n_a}\sum_{i=0}^{n_a} F^i log(\hat{F}^i) + (1-F^i) log((1-\hat{F}^i)) \tag{3}$$

where F^i is the feasibility of the i^{th} action. By optimizing this target, we are able to train a module to tell the action feasibility.

After the module is trained, we are able to use it for better inference. In specific, given the action preference prediction given by the module to be $\hat{A}$, the initial strategy to select action is

$$a* = (\hat{A}) \tag{4}$$

To avoid bad actions, we now select the target action through

$$a* = (\hat{A} \odot \hat{F}) \tag{5}$$

where $\odot$ is element-wise multiplication.

5 Experiments

In this section, we will introduce the experiments based on our approaches. Note that due to time and resource constraints we did not evaluate the auxiliary loss approach which involved retraining the full MOCA model. The experiments only relate to the stand-along action feasibility module.

5.1 Settings

The model used for action feasibility prediction is a ResNet-18 with 2 fully connected layers. The threshold for positive prediction is 0.7 (value of logit), i.e. we only assign the sample as a feasible action if the output logits have a value greater than 0.7. To train such a model, we used Adam Optimizer with the learning rate 1×10^{-5}, batch size of 32, and trained model for 20 epochs. Weight decay is also used to prevent overfitting.

5.2 Baselines

We present below the two baselines that we plan to evaluate against our proposed approach along with the metrics that will be used for the evaluation.

Sequence-to-Sequence with Progress Monitor Along with collecting the ALFRED dataset, Shridhar et al. (2020a) proposed a baseline model for the task. They model the agent with a CNN-LSTM sequence-to-sequence architecture. A bidirectional-LSTM is used to generate a representation of the language instructions. A CNN encodes the visual input at each time step. And a decoder LSTM generates a sequence of low-level actions while attending over the encoded language and visual input. The model is trained to produce the actions and associated interaction masks from expert trajectories using imitation learning. The interaction mask is produced by a three layer deconvolution network trained with binary cross-entropy loss on ground-truth object segmentations.

The ALFRED task requires long action sequences to be completed. Shridhar et al. (2020a) propose to use two auxiliary losses to monitor the progress of the agent towards the completion of the task. The first loss involves predicting the current process (a scalar between 0 and 1) using the decoder hidden state, the visual input, the attended language instructions, and the previous action. In addition to the global progress, the model is trained to predict the number of sub-goals completed so far.

Modular Object-Centric Approach (MOCA) The Modular Object-Centric Approach (MOCA) (Singh et al., 2020) decomposes the interactive instruction following procedure into action, and policy generation tasks. Two modularized networks, named action policy module (APM) and visual perception module (VPM), deal with the action planning and visual localization tasks respectively via attentive filtering visual and language information. The method also proposed to use an obstruction detection module to address the difficulties in being stuck around the obstacles. The method successfully gains improvements from previous methods and is of state-of-the-art position.

For training MOCA, the objective function is composed of the loss for APM and VPM. To guide the discovery of objects, the basic objective functions are two cross-entropy losses that enforce the model to discover objects and the corresponding

action classes for each step. Two loss functions helping to achieve subgoals, and monitor the overall progress.

5.3 Metrics

Here we discuss the metrics that we use to measure the performance in Alfred dataset. For the action feasibility model, we use classification accuracy and average precision as the metric to evaluate the model.

Task Success To evaluate whether tasks have been successfully completed, the authors define two categories for success. For the task success metric, a score of 1 is given to a tasks trial when all of the goal conditions for that task have been met. If at least one goal condition is not met at the end of the sequence, a score of 0 is given. The main limitation of this metric is that the state-of-the-art models still perform poorly on the dataset, thus the percentage of tasks that meet every goal condition is very low.

Goal-Condition Success To overcome this limitation, goal-condition success measures the proportion of goal-conditions that have been met for the task. For example, if the completion of a task requires 5 goal-conditions and only 2 are met during testing, a score of 0.4 will be given. In general, tasks with longer sequences of instructions have a lower goal-condition success as the directive is more complex. As a result, it is important to use the same test splits across experiments to keep the task length constant.

Path Weighted Metrics A major limitation of both metrics is that one could attain a relatively high score by performing a long sequence of random acts. To overcome this, the authors perform path weighting (PW), which gives a lower score to action sequences that are long compared to the expert demonstration. The weighting follows the below equation where p_s is the path weighted score, s is the non-path weighted score, L^* is the length of the expert demonstration sequence and $\hat{L}$ is the length of the predicted sequence.

$$p_s = s * \frac{L^*}{max(L^*, \hat{L})}$$

Sub-Goal Evaluation The last metric is sub-goal evaluation. Initially, the model is forced to follow the beginning part of the expert demonstration. Afterwards, the model must predict actions to complete the remaining sub-goals given the full textual instruction and the most recent image input. This metric allows analysis into how previous action sequences affect the prediction of future action sequences.

Reports/Report/feas_infeas.png

Figure 2: On the left is an image where "MoveAhead_15" is feasible and on the right, "MoveAhead_15"is infeasible because there is a door blocking the path.

5.4 Data Collection

To train an action feasibility prediction model we need to gather a dataset with ground truth labels. Given time constraints, we focus on navigational actions and not interactive actions. Through experimentation, we found that the actions "LookUp_15", "LookDown_15", "RotateLeft_90" and "RotateRight_90" were always feasible. The only navigation action that could be infeasible was "MoveAhead_90". Thus, our dataset consists of pairs of images and boolean values determining whether "MoveAhead_90" was feasible or not for that image. An example of typical images can be observed in figure Figure 2.

In the training dataset, the same scene numbers are frequently reused with a variation in the layout of small items such as credit cards. Since the presence of these small items is largely irrelevant to whether or not moving ahead is feasible, we store one scene per scene number. For each unique scene, we get all the navigable, non-occupied x, y, z positions of which there is an average of 92 per scene. Within these positions, we rotate the agent four times and store an image as well as a boolean

feasibility value for each rotation. We perform the same data collection on the unseen dataset in order to perform validation evaluations. There are 108 unique training scene numbers and 4 unique unseen scene numbers.

5.5 Results

In this section, we will introduce our results. We report results on two experiments to verify the effectiveness of our hypothesis. We would like to verify 1) if the action feasibility prediction could help, and 2) if it is possible to build such a model that predicts the feasibility actions. To answer the first question, we build an oracle feasibility predictor that perfectly predicts the feasibility of every navigational action, and we use the oracle outcome to prune the action space that the agent will operate on. When the oracle predict that the action is not feasible, we used two strategies to select the alternative action, includes 1) the most likely navigational action, and 2) the most likely action (including interactive actions). In the following, we will discuss these settings in detail. Here we report the results on Alfred validation set.

Is action feasibility prediction going to help the task success? As mentioned above, to answer this question, we build two oracle models that perfectly predict the action feasibility and prune the action space. The performance of such a model is very important since it represents the upper bound performance of the actual action predictor, i.e., the performance of the trained action feasibility predictor should not be higher than that from the oracle. It also reveals some pitfalls and more fundamental issues in the robot. The results are shown in Table 4. In the table, the MOCA term means the trained model from MOCA paper. Seq2Seq + PM is the model provided by alfred's original paper. MOCA + Oracle All means the alternative action selected is from all other actions except for the excluded ones. MOCA + Oracle Nav is the oracle model that selects only the navigational actions.

It can be observed from the table that when the model has seen the room in the training set, the improvement from using the oracle predictor is very limited, while in the unseen cases there is a clear improvement. We believe this is due to the MOCA model already implicitly learning to predict the action feasibility during training in the seen environments. Furthermore, the MOCA has also a similar module inside it that helps avoid making

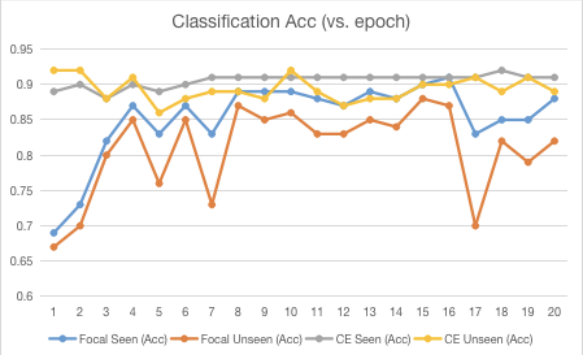

Figure 3: Training plot

mistakes by forcing the agent to turn after hitting the wall for some time, which thus makes the action feasibility prediction less useful. However, in the unseen cases, the MOCA is not very good at generalizing the from the training set and performs more infeasible actions on average. In these cases, the action feasibility prediction becomes helpful.

We also realize that there exist some more fundamental issues in the pipeline of MOCA. To be more specific, these models might have limitations in terms of planning, and using an action feasibility module might be too late to correct such fundamental issues.

Is it possible to identify the feasibility of an action? To answer this question, we used the data collected with the approach described before and trained a separate model. In short, the answer to this question is yes, and the trained model transfers well from seen to unseen spaces, indicating the potential improvement in unseen cases, based on the results from the oracle model. From Table 5, we can observe that the module succeeds in predicting the action feasibility and reaches an overall accuracy in seen spaces of 90%. While the performance in seen rooms is less informative, it's also observed that the prediction performance under unseen tasks is very competitive. We also compared the Focal Loss and Cross-Entropy Loss, showing that focal loss is helpful for the prediction of negative samples, it harms the performance in predicting positive samples. The plot showing the accuracy during training is shown in Figure 3.

6 Conclusion

In conclusion, in this project we explore the ALFRED task. We identified the issue of action feasibility as a possible area of improvement for the MOCA model. We proposed two approaches to

	Seen		Unseen	
	Task (PW)	Goal-Cond (PW)	Task (PW)	Goal-Cond (PW)
Seq2Seq + PM	3.7 (2.1)	10.0 (7.0)	0.0 (0.0)	6.9 (5.1)
MOCA	**20.8** (14.4)	**29.7** (**22.6**)	2.8 (1.4)	12.8 (7.6)
MOCA + Oracle All	20.0 (**14.7**)	29.5 (**22.6**)	3.5 (2.0)	**14.2** (**8.8**)
MOCA + Oracle Nav	18.4 (12.9)	27.5 (21.5)	4.1 (**2.2**)	14.1 (8.6)

Table 4: Performance of MOCA+Oracle models.

	Focal Seen	Focal Unseen	CE Seen	CE Unseen
Accuracy	0.91	0.87	0.92	0.89
Pos AP	0.85	0.81	0.97	0.95
Neg AP	0.89	0.83	0.6	0.55

Table 5: Action prediction performance from trained model, in the held-out validation dataset. Here Pos AP and neg AP is the average precision on positive and negative samples respectively. (Feasible action/ Infeasible action)

solve this issue. Due to time constraints we only performed experiments on one of these approaches: the stand-along action feasibility module. This module, trained on data from the simulator, predicts the feasibility of the "Move Ahead" action given an image. Our model reaches 90% accuracy in the seen cases with no significant loss in performance in unseen cases. Finally, our oracle experiments show that this external action feasibility module can improve performance in terms of success rate and goal conditioned success in the unseen scenes.

In future work, we would like to explore the performance of the auxiliary loss approach which does not require a separate feasibility module but updates the MOCA model itself. Finally, our experiments indicate that other components of the MOCA model may be limiting the benefits from modeling action feasibility.

References

P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. 2018. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3674–3683.

Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2016. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48.

Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: visual question answering. *CoRR*, abs/1505.00468.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, pages 1533–1544.

Jonathan Berant and Percy Liang. 2014. Semantic parsing via paraphrasing. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1415–1425.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. *arXiv preprint arXiv:1805.04793*.

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 *Neural Information Processing Systems (NeurIPS)*.

Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. 2017. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2901–2910.

Jacob Krantz, Erik Wijmans, Arjun Majundar, Dhruv Batra, and Stefan Lee. 2020. Beyond the nav-graph: Vision and language navigation in continuous environments. In *European Conference on Computer Vision (ECCV)*.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988.

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

Raymond J Mooney. 2007. Learning for semantic parsing. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 311–324. Springer.

Stephen Muggleton and Luc De Raedt. 1994. Inductive logic programming: Theory and methods. *The Journal of Logic Programming*, 19:629–679.

Adam Santoro, David Raposo, David GT Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. *arXiv preprint arXiv:1706.01427*.

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 *The IEEE 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. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

Phillip D Summers. 1977. A methodology for lisp program construction from examples. *Journal of the ACM (JACM)*, 24(1):161–175.

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. In *ECCV (16)*, pages 38–55.

Chenfei Wu, Jinlai Liu, Xiaojie Wang, and Ruifan Li. 2019. Differential networks for visual question answering. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(01):8997–9004.

Fan Wu, Zhongwen Xu, and Yi Yang. 2017. An end-to-end approach to natural language object retrieval via context-aware deep reinforcement learning. *CoRR*, abs/1703.07579.

Fanyi Xiao, Leonid Sigal, and Yong Jae Lee. 2017. Weakly-supervised visual grounding of phrases with linguistic structures. *CoRR*, abs/1705.01371.

Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alexander J. Smola. 2015. Stacked attention networks for image question answering. *CoRR*, abs/1511.02274.

Pengcheng Yin and Graham Neubig. 2017. A syntactic neural model for general-purpose code generation. *arXiv preprint arXiv:1704.01696*.