11-777 Spring 2021 Class Project

Andrew Singh* **Ankit Ramchandani*** **Vashisth Parekh***
{andrewsi, aramchan, vparekh}@andrew.cmu.edu

Abstract

Template for 11-777 Reports using the ACL 2021 Style File

1 Introduction and Problem Definition (1-1.25 pages)

Thesis statement or Hypothesis we are aiming to prove
"Our approach is better is not a hypothesis"

*Everyone Contributed Equally – Alphabetical order

2 Related Work and Background

2.1 ALFRED

Though ALFRED is a relatively new task, a multitude of approaches have recently been proposed that demonstrate improved performance over the baseline introduced in (Shridhar et al., 2020a). The baseline model consists of a CNN to encode the visual input at each timestep, a bi-LSTM to encode the language directives, and a decoder LSTM to infer the action at each timestep while attending over the language encoding. (Corona et al., 2020) used the same architecture as the baseline, except they maintained separate modules for each *subgoal type* (e.g., GOTO, PICKUP). They then used a high-level controller to choose which module to execute at each step based on the language directives. (Singh et al., 2020) proposed a vision module for generating interaction masks and an action module for predicting actions, with the vision module first predicting the class of the object of interest and then generating the pixel-wise interaction mask given the predicted object class. (Storks et al., 2021) addressed ALFRED's long action sequences by training the model to execute one subgoal at a time rather than all subgoals at once, and they address the agent's poor navigation performance by augmenting the agent's perception with additional viewing angles that are used for training an object detection module and for predicting the agent's orientation angle relative to the goal.

Unlike methods that learn a direct mapping from observations to actions end-to-end, (Saha et al., 2021) proposed a truly modular framework that is able to learn from unaligned or weakly aligned data as opposed to requiring expert demonstrations. Their mapping module includes a novel mapping scheme based on graph convolutional networks (Kipf and Welling, 2016) for improved navigation, and their language module leverages a pre-trained model to perform joint intent detection and slot filling on the language directives.

While the previously mentioned works improve upon the modeling approach proposed in the original ALFRED paper, (Shridhar et al., 2020b) also proposed a new environment to address the challenge of generalizing to unseen tasks. They aligned tasks in ALFRED with a purely textual environment, TextWorld (Côté et al., 2018), allowing agents to first learn in an abstract setting in order to generalize better in the embodied setting. They additionally introduced a modular architecture to demonstrate the effectiveness of their ALFWorld environment, consisting of a state estimator that translates visual observations to text, an abstract text agent pre-trained in TextWorld that generates high-level actions from textual observations and a goal, and a controller that translates high-level actions to sequences of low-level actions in the embodied environment.

To the best of our knowledge, the papers discussed above cover all of the approaches proposed thus far that attempt the full ALFRED task.

2.2 Related Tasks

2.2.1 Vision and Language Navigation

While ALFRED requires navigation and interaction with objects based on visual and language input, a related task is just vision and language based navigation (VLN). (Fried et al., 2018) proposed a policy consisting of two modules: an instruction follower model that produced a step-by-step action sequence from visual and textual input, and a speaker model that predicted the probability that a particular language instruction describes a given sequence. To predict the final trajectory, multiple trajectories were first generated by the follower model, and the one most likely to match with the natural language description (as assessed by speaker model) was chosen. (Wang et al., 2019) proposed an improvement over the previous method by learning a LSTM (Hochreiter and Schmidhuber, 1997) and attention-based policy using reinforcement learning (RL). They also used a "speaker model" which was used to generate an intrinsic reward for the RL algorithm based on the probability of the language description matching the predicted action sequence. (Wang et al., 2018) proposed a method to jointly perform imitation and reinforcement learning using a policy that consisted of an "action predictor" used to predict the final action based on inputs coming from model-free and model-based RL modules. The model-free module was an LSTM and attention-based network similar to other approaches (Shridhar et al., 2020a; Wang et al., 2019). The model-based module "imagined" multiple different trajectories in the future, and produced a combined representation to the action predictor. (Wani et al., 2020) performed several experiments on a long-horizon navigation task in a realistic 3D setting to empirically show that using a semantic map-like memory can significantly boost navigation performance. (Hao et al., 2020) pre-

trained their model on image-text-action triplets in a self-supervised manner. Their model was able to generalize better in unseen environments, improving the SOTA in the Room-to-Room task (similar to ALFRED). (Majumdar et al., 2020) improved VLN performance by using a visiolinguistic transformer based model that scores the compatibility between an instruction and a particular visual scene. Pretraining on the image-text pairs from the web improved the performance of the VLN.

2.2.2 Embodied Question Answering

Embodied Question Answering (Das et al., 2018a) (EmbodiedQA) is a related task in which an agent spawns at a random location in a 3D environment and is asked a question about an object. To correctly answer, the agent must navigate the environment and gather information through egocentric vision about the object and its surroundings. This challenging task requires many of the skills needed in the ALFRED benchmark, including active perception, commonsense reasoning, goal-driven navigation, and grounding language to vision and actions.

(Das et al., 2018a) proposed an approach with a two-step navigation module: a planner that selects actions and a controller that executes those actions a variable number of times. Their agent is initialized via imitation learning and then fine-tuned via reinforcement learning. (Das et al., 2018b) improved upon this approach by introducing a high-level policy that proposes compositional sub-goals to be executed by sub-policies. They train their model via imitation learning in a bottom-up fashion, first training the sub-policies before training the high-level policy. They then fine-tune their model via reinforcement learning in a similar bottom-up fashion, allowing the high-level policy to adapt to the behavior of the sub-policies.

(Yu et al., 2019) generalized the EmbodiedQA task to multiple targets; instead of a question asking only about a single object, it may ask about several objects and require comparative reasoning. They propose a novel architecture for the task consisting of four modules: a program generator that converts the question to executable sub-programs, a navigator that executes these sub-programs to guide the agent to relevant locations, a controller that selects relevant observations along the agent's path, and a visual question answering module that uses the observations from the controller to predict the final answer.

2.3 Relevant ML Methods

2.3.1 Multimodal Alignment

In ALFRED, the agent receives all natural language instructions at the beginning of the episode, but receives visual observations at each time-step. It is imperative for the agent to align the natural language directives with its current visual observation so that it can spot objects of interest described in natural language in the current visual scene. In this section, we summarize some research in multimodal machine learning focused on this problem of learning such soft alignment (Baltrušaitis et al., 2018).

(Yu and Ballard, 2004) used a graphical model to align objects in (egocentric) images with spoken words. (Mei et al., 2015) uses a bi-LSTM with a multi-level aligner to map instructions with navigational actions. (Ma et al., 2019) proposed a visual textual co-grounding alignment mechanism and a corresponding progress monitor. They used the hidden state from the previous timestep of their LSTM to generate textual and visual grounding, which helps their agent decide which action to take next. Similarly, (Wang et al., 2019) used an LSTM to predict actions and included an attention mechanism on visual and textual input based on the current hidden state of the LSTM. (Ke et al., 2019) used attention mechanism over language to compute how the previous action aligned with the description. (Wang et al., 2018; Shridhar et al., 2020a) both used modules which perform attention over textual input using the hidden state of the LSTM, so that the agent knows which words in the input text to focus on.

2.3.2 Generalization in Multimodal Settings

The ALFRED dataset has unseen splits of validation and test data which measure generalization of the learned policy, but multimodal models are more prone to overfitting due to their increased capacity (Wang et al., 2020). To this end, (Wang et al., 2020) also proposed a gradient blending approach, which computes optimal blends of modalities based on overfitting behavior. This approach achieves SOTA results on egocentric action task similar to ALFRED. (Alet et al., 2019) presented a meta-learning strategy where they separately trained each modular component on related tasks and then combined them to create a more general model that scales across tasks. More specifically to ALFRED, (Nguyen and Okatani, 2018) proposed multi-task

learning approach that enables visual-language representations that can be generalized to other tasks. Their algorithm used representation encoders that learn hierarchical features by fusing visual and semantic representations and task specific decoders that decode those features however they see best fit for the given task.

2.3.3 Reinforcement and Imitation Learning

All known SOTA approaches (Singh et al., 2020; Corona et al., 2020; Storks et al., 2021) for ALFRED use imitation learning (IL) (Hussein et al., 2017), despite IL having several known limitations because the standard i.i.d assumptions are not met (Ross and Bagnell, 2010). Methods like DAgger (Ross et al., 2011) that attempt to mitigate the limitations of IL cannot be applied directly because new data cannot be generated on the fly in ALFRED (Shridhar et al., 2020a). For these reasons, in this section, we describe methods that use a combination of IL and reinforcement learning (RL) techniques in addition to the ones that were covered in Section 2.2.1 (Wang et al., 2019, 2018).

Many methods have been proposed to use RL methods when expert data is present. (Ho and Ermon, 2016) proposed a method to directly learn a policy from expert data that optimizes a reward function that would be obtained by running inverse RL (Abbeel and Ng, 2010) on expert data. Notably, their method directly outputs the policy and does not involve running inverse RL to extract the reward function first, which could be very costly. They experimentally show that their method outperforms other IL methods and often achieves expert level performance. (Reddy et al., 2019) proposed a much simpler alternative to (Ho and Ermon, 2016) which still achieves competitive performance. They simply give the agent a positive reward when it matches the expert action and no reward otherwise. This simple idea is theoretically motivated and forces the agent to return to states seen by the expert. (Salimans and Chen, 2018) proposed a RL-based method to solve the challenging Atari game, Montezuma's Revenge, using a single demonstration. Their main contribution was to train the agent using a curriculum: they trained the RL agent to reach the goal by starting from states in the demonstration in reverse order. In other words, they first trained an RL agent to reach the goal state from second-to-last state in the demonstration, then from third-to-last, and so on. The main insight was that the RL agent had to learn only a sub-task at each step, overcoming any hard exploration. (Hester et al., 2018) proposed a method to do Deep Q-Learning (Mnih et al., 2013) from demonstrations by adding a term to the loss function that forces the Q-value of the expert action to be at least a margin higher than other actions. This term adds a trade-off between following the expert action and the optimal action as predicted by the Q-values. (Rajeswaran et al., 2017) proposed a method to learn complex, non-trivial manipulation tasks using RL and IL. They use IL to warm start the policy, and then train it using RL with a modified gradient update which forces the policy to stay close to expert actions. (Garmulewicz et al., 2018) proposed a simple modification to the loss function used in actor-critic methods to account for expert data and showed that their simple modification can achieve satisfactory results on challenging tasks like Montezuma's Revenge. (Nair et al., 2018) also proposed a method that involves a modification to the loss function to account for IL, but they only add this extra loss term when the learned critic believes that the expert actions are indeed better than the policy action. In other words, their modification accounts for cases when expert data may not be perfect.

3 Task Setup and Data

3.1 Task Definition

We plan to work with the ALFRED dataset (Shridhar et al., 2020a) with the goal of learning a set of actions in an indoor household setting which will help an agent complete a task described by natural language. The tasks require navigation and interaction with multiple objects in the scene. Each interaction action requires a pixel-wise interaction mask to specify the object of interest. The agent receives high-level and low-level natural language instructions at the beginning of the episode, and can use egocentric visual observation (i.e. access to current RGB image, depth map, and instance segmentation map) at each time step as input. The agent produces one or two outputs at each time step: the current action to take, and, if the action involves interaction, the interaction mask of an object of interest.

We intend to predict the interaction mask pixel-wise instead of using any other coarser representations like bounding boxes. We also intend to use inputs in their rawest representation (e.g. raw image data instead of extracted features) for maximum generality and flexibility of downstream methods. Furthermore, we plan to develop a method to solve the full task of navigation and interaction in the ALFRED dataset. We clarify this to convey that we are not working with a small sub-task or a sub-problem of the dataset. Since current methods (Corona et al., 2020; Singh et al., 2020; Shridhar et al., 2020a) struggle with generalization to novel objects and environments, we will attempt to primarily focus on improving generalization performance, which is measured by an "unseen" split of the test set which contains new environments and objects.

3.2 Dataset Statistics

In this section, we present a subset of the analysis we performed. **We encourage the reader to see the Jupyter notebook stored in the *Analysis* folder for full list of figures pertaining to the analysis since this report only includes a subset.**

3.2.1 Metadata Analysis

In this section, we analyze various metadata in ALFRED. Table 1 shows average values for the quantitative metrics we chose to measure on ALFRED. We can see that the averages are fairly consistent across splits. The navigation-interaction ratio indicates that for every interaction action in a demonstration, there are roughly 9 navigation actions. The mask coverage indicates that on average, the ground-truth interaction mask covers a rather small (15-17%) proportion of the image. The step-object coverage of nearly 1 indicates that for almost all interactions, the name of the object of interest is mentioned in the corresponding language directive. By manually inspecting examples with low interaction step coverage, we find that the object's name are usually substituted with a synonym (e.g."rag" for "cloth" and "scoop" for "ladle").

	Train	Valid (seen)	Valid (unseen)
Steps per directive	6.68	6.64	6.27
Tokens per step	12.39	12.18	12.63
Task desc. tokens	10.02	10.09	10.04
Images	286.75	287.24	277.72
Actions	49.78	50.12	46.98
Images per action	6.08	6.02	6.12
Actions per step	7.6	7.72	7.72
Nav-interact ratio	9.19	9.25	8.13
Total objects	33.2	32.84	38.44
Mask coverage	0.17	0.17	0.15
Step-object coverage	0.86	0.85	0.88

Table 1: Average values of various quantitative aspects of ALFRED by split. See appendix for definitions.

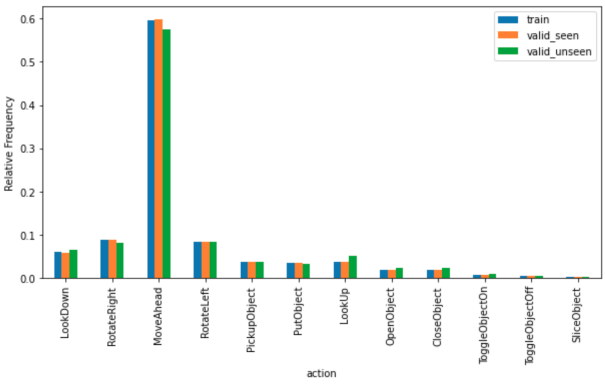

Figure 1: Relative frequency of each action type by split

Furthermore, Figure 1 shows the frequency of the 12 different actions (5 navigation actions + 7 interaction actions). Note that all splits have fairly equal frequency for all actions. Also, note that 60% of actions are "move ahead" actions which signifies the importance of a navigation module we may need in our model.

3.2.2 Textual Analysis

For textual analysis, we first identified out of vocabulary (OOV) words in the training and validation set using a vocabulary of 685k words defined by

spaCy. We found that less than 0.002% of all words were OOV in any split, indicating the dataset is already quite clean. Most of the OOV words were just misspelled (eg: "stovve"), indicating that it would be important to preprocess text using a simple spell checker before using it for downstream tasks.

We also found the top few synonyms used to describe objects in the dataset. This was performed by comparing the similarity of word vectors of all objects in the dataset with all common nouns identified in all task descriptions. The complete results are in the Jupyter notebook, but a few results are shown in Table 2. This reveals that our model will need to be robust enough to recognize synonyms of different words in order to be successful.

Furthermore, we analyzed how many objects present in the scene are directly referred to in the step-by-step instructions. The results are plotted in a histogram in Figure 2, which reveals that much less than 40% of objects in the scene are actually referred to in the instructions.

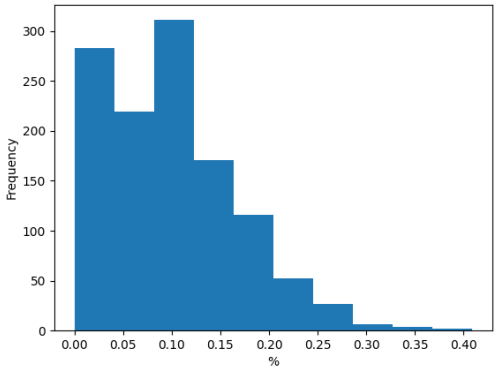

Figure 2: Percentage of objects referred to in instructions compared to objects in the scene

3.2.3 Visual Analysis

Due to compute constraints, most visual analysis was performed qualitatively by inspecting visual quality of different types of images. Figure 3 shows some sample RGB, depth and instance segmentation images. All images retrieved during simulation are of size 300 x 300.

Object Name	Synonyms used in task descriptions
Coffee Machine	Espresso Machine, Beverage Machine
Chair	Couch Chair, Sofa Chair
CD	DVD
Side Table	Corner Table
Butter Knife	Bread knife
Ottomon	Loveseat, Recliner
Fridge	Kitchen Fridge, Refrigerator
Poster	Wall Photo, Picture
Safe	Safety Box
Soap Bottle	Lotion Bottle

Table 2: Synonyms (i.e. closest words in embedding space) used in task descriptions of some objects in the dataset

3.2.4 Other Qualitative Analysis

In addition to the quantitative analysis, we also analyzed the solvability of the task. Our analysis in Figures 7 and 8 (in Appendix) shows that the unseen split of the validation set contains objects from the same classes as the training data, but could contain novel instances of those objects in novel environments. Since classes remain the same during training and testing times, the training data contains full information to solve the task, meaning a sufficiently intelligent agent should be able to solve the task, given training data.

3.3 Metrics

There are two primary metrics for evaluation. The first is "task success", which is a binary value indicating if the object positions and state changes correspond correctly to the goal-conditions of the task at the end of the action sequence. The second is "goal-condition success", which is the fraction of required goal-conditions that were completed at the end of the episode. Note that "task success" is true only if "goal-condition success" is 100%. Additionally, there exists a path-weighted version of these two metrics that considers the length of the episode, penalizing longer action sequences. For example, in the path-weighted version, an agent would receive half the score for taking twice as long as an expert to complete the task.

(a) (b) (c)

Figure 3: Sample RGB, depth, and instance segmentation images retrieved from AI2Thor simulator

4 Models (2 pages)

4.1 Baselines

4.1.1 Baselines and Other Methods

We plan to use the following methods for comparison purposes, which are cited below and explained in Section 2.

- **Seq-2-Seq PM** (Shridhar et al., 2020a)
- **MOCA** (Singh et al., 2020)

4.1.2 Proposed Ablations

Shridhar et al. (2020a) and MOCA (Singh et al., 2020) perform similar input ablation studies for their approaches. They individually remove four inputs from the model: language, vision, the step-by-step instructions, and the goal statement, and evaluate the model's performance in each case. We also plan to conduct a similar ablation study on our model.

In addition, we also plan to perform an ablation study over the training method. While our modeling plans have not been finalized, we currently are interested in exploring an RL-based approach. To empirically verify if RL performs better than IL, we plan to train two variants of our model: one trained using RL, and the other trained using simple IL like other approaches.

4.1.3 Metrics

We plan to use the following existing metrics to evaluate our approach.

1. **Task Success**: Task success is simply a binary value indicating whether the final states and positions of objects of interest in the trajectory align with expected states (i.e. if task is completed).

2. **Goal-Condition Success**: This is the ratio of goal-conditions completed at the end of the episode to the total number of goal-conditions required for task success.

3. **Path Weighted Metrics**: We can also compute the path weighted versions of the above metrics. So, if the model takes twice as many actions has the expert, the original task success and goal-conditioned success scores would decrease by half.

Moreover, to better identify performance bottlenecks in ours and other approaches, we propose the following other metrics:

1. **Navigation Object Success (Nav-Obj)** and **First Navigation Object Success (First-Nav-Obj)**: Of all the objects the expert interacted with, Navigation Object Success is the proportion of objects that the agent approached at some point in the episode. Here we define "approached" as coming within a certain distance and facing the correct orientation. We also record whether the agent could approach just the first object that the expert interacted with, denoted by First Object Navigation Success. This metric is important because it more clearly shows whether the agent has at least learned to navigate correctly, which can be measured in a more disentangled manner by only considering navigation to the first object. This is because there will be likely interaction actions before approaching the later objects, making it difficult to reliably attribute credit of future success/failure to navigation alone.

2. **Interaction Success (Int-Succ)**: Interaction Success is the proportion of attempted interactions by the agent that were successful. Note that "successful" here simply means that the interaction action was executed in the simulator without an API failure. A high value for this metric indicates that the agent is able to at least successfully predict an appropriate action and interaction mask to interact with objects, even if they may not necessarily be objects that are relevant to the goal.

3. **Bad Mask Failure Rate (Bad-Mask)**: Bad Mask Failure Rate is the proportion of interaction failures that were due to a bad interaction mask. A high value indicates that the primary obstacle to an agent successfully interacting with objects is the interaction mask, while a low value indicates that most of the interaction failures are due to a different reason, such as incorrect action prediction (see item 5: Interaction Action Prediction Success).

4. **Unnecessary Interaction Ratio (Unnec-Int)**: Unnecessary Interaction Ratio is defined as the number of objects the agent interacted with that the expert did not, divided by the number of objects the expert interacted with. A high value will reveal that the agent is interacting with proportionally many objects not

relevant to the task, while a low value indicates that the agent is focusing on the objects that are relevant to completing the task.

5. **Interaction Action Prediction Success (Int-Act-Pred)**: Interaction Action Prediction Success is the fraction of times the model predicts the right interaction action given that it has identified the instance segmentation mask of the correct object. A high value for this metric indicates that once the agent can produce a correct mask, the task of predicting which action to execute is not a bottleneck. A low value suggests that even when the agent is able to produce a correct mask, predicting which action to take is still a significant obstacle to successful interaction.

Note that a recurring theme in most proposed metrics is to measure performance over each aspect needed to solve the task in as much isolation as possible.

In addition to the above base metrics, we also plan to group all metric calculations by task type to understand models at a more fine-grained level.

4.1.4 Results table

As mentioned in 4.1.1, we compare our approach to two other approaches. We report our overall results in Table 3. Additionally, we report results for our proposed intrinsic metrics on the validation set and compare to the baseline model in Table 4, and analyze them in Section 4.3.

4.2 Qualitative Analysis

We also ran the baseline and MOCA policy on a few examples to qualitatively analyze the agent's performance. The screen recordings of these examples are uploaded on Google drive, and linked below.

In example 1, titled Base-Ex1, the agent has to take a spray bottle from the basin counter and place it on the toilet. Instead, the agent picks up a cloth (placed near the spray bottle) from the basin counter and just moves on the other end of the counter. Note that even though the agent moved correctly to the spray bottle and made only a small mistake (i.e. picked up cloth instead of spray bottle), it could not recover from the mistake, and ended up doing something very different at the end. This highlights a common problem with vanilla imitation learning that the agent cannot recover from small mistakes because that are out of training distribution.

In the next example, titled Base-Ex3, the agent is supposed to microwave a potato and put it in the fridge. Instead, the agent starts a length trajectory without every picking up the potato: It opens, operates, and closes the microwave repeatedly (sometimes placing nothing inside of it) and then repeatedly opens and closes the fridge (placing nothing inside of it). This example (and many others we have seen) highlight the primitiveness of the agent, and show that the agent is not really learning anything useful, but just operating on statistical co-occurrences: for example, whenever it sees a fridge/microwave, it just repeats a particular action sequence (i.e. open, put object inside, close for fridge or open, put object inside, close, turn on, pick object up, turn off).

The MOCA policy does perform relatively better than the baseline, but the same problems mentioned are noticed. In example Moca-Ex4 the agent is asked to slice an apple in the pan with a knife; instead it picks up a fork and travels back and forth in the kitchen. As with baseline, the agent makes a mistake by picking up the fork but is not able to recover from the mistake made early on.

In the next example, Moca-Ex5 the agent is asked to take a mug, microwave it, and place it on the table across the room again. Unfortunately, the agent simply walks in loops inside the kitchen without even trying to pick up the mug. Similar to the baseline policy, this example reveals that the policy is not really learning anything intuitive, and is not able to attend over the correct objects in long, complex trajectories.

4.3 Inferences, Interpretations, and Insights

In this section, we analyze the implications of both our quantitative and qualitative results.

Intrinsic Metrics (Table 4)

1. Navigation remains a challenge on unseen data. On the unseen split, even MOCA is unable to navigate to the first relevant object one-third of the times, and only navigates to just over half of the relevant objects throughout the episode. However, the fact that MOCA performs significantly better than the baseline model on the unseen split suggests that the language-guided dynamic filters and obstruction detection that MOCA adds could be beneficial to generalizing navigation to new

environments.

2. Failed interactions are a bottleneck to overall performance on unseen data. We see that even MOCA has just a 24.5% interacton success rate on the unseen split, while the baseline has a lower 14.8% success rate. In the case of MOCA, most of these failures (73.7%) are due to incorrect masks, while just under half are for the baseline. Given that one of MOCA's key contributions is a novel module for object-centric mask prediction, this is somewhat surprising. This could however be explained if the baseline model first encounters other failure modes that MOCA is able to overcome, so the baseline is not able to reach the point where the interaction mask is the bottleneck. Looking further into the data, we find that MOCA attempted nearly 2.5× more interactions than the baseline did, suggesting that perhaps other bottlenecks such as navigation are limiting the baseline's potential for interacting with objects (see item 1).

3. These models do not focus well on the objects relevant to the goal, interacting with roughly the same number of irrelevant objects as there are relevant objects in the trajectory (shown by the Unnecessary Interaction Ratio of roughly 1 for both models). This suggests that these models are not successfully integrating the language directives with their perceptual inputs to select the objects specified by the directives.

4. Action prediction remains a significant bottleneck to successful interaction, especially on unseen data. Even after producing a correct interaction mask, MOCA still only predicts the correct interaction action 14% of the time, and the baseline only 0.8% of the time, on the unseen split. This suggests that while mask prediction is a significant challenge, an equally (if not more) important task is predicting the correct action to take to interact with the object.

Task-Level Metrics by Task Type We group the two task-level metrics, Task Success and Goal-Condition Success, by the 7 task types in ALFRED and plot the results in Figures 4 and 5. While the baseline model struggles with all task types, MOCA demonstrates a substantial improvement on all tasks in the seen split except on the **Stack & Place** task. This could be due to the fact that **Stack & Place** is the only task with a movable receptacle, presenting a unique challenge that is not found in other tasks.

The task that both models perform best on is the **Clean & Place** task. In the unseen split, MOCA interestingly performs much better on this task than on any other task type. Further exploration of the unique attributes of this task may be needed to determine why MOCA's success rate is so much higher on the unseen split than other tasks. Perhaps it is because this task has less change from seen to unseen than the other tasks, or perhaps it is simply due to variability in the evaluation: the raw number of successes for MOCA on this task in the unseen split was just 18, and was 1-5 for the other tasks.

The key insight from these plots is that for many task types, a substantial improvement in seen performance does not necessarily imply the same improvement in unseen performance. Learning how to successfully generalize to new environments and objects is a distinctive challenge and must be addressed explicitly when designing an approach.

We also did a similar task level analysis on metrics we designed ourselves (not discussed due to space constraints), and the results of the same are provided in a Jupyter notebook added in the submission folder.

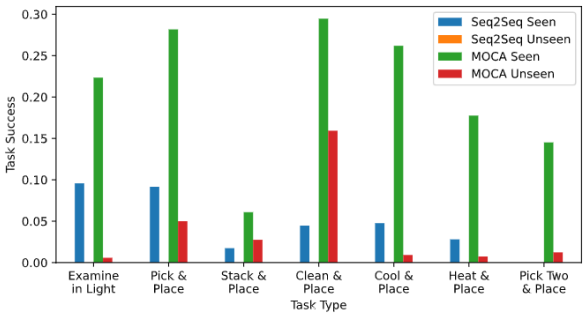

Figure 4: Task Success by Task Type

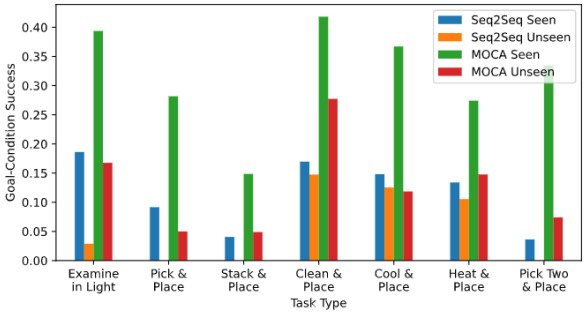

Figure 5: Goal-Condition Success by Task Type

Need for RL. The most important lesson we learned from our analysis was just how important RL could be for this task. Below we highlight specific reasons we think that RL is crucial for solving a multimodal navigation and interaction task like ALFRED:

1. Our qualitative analysis highlights cases where the agent makes one mistake and cannot recover from it because it has deviated too much from the expert trajectory and does not have enough information to get back on the right course. This is a common problem that imitation learning agents face (Ross et al., 2011), and something RL agents typically won't suffer from because they are trying to maximize a reward instead of rote learn a trajectory.

2. Since supervision in RL comes from a self designed reward function, we can force the agent to learn much more task specific knowledge compared to the vanilla imitation learning. For example, Table 4 shows that in most cases agents trained with both MOCA and baseline agents interact with more unnecessary objects than the total objects needed to complete the task (because Unnec-Int is often greater than 1). This is of course an undesirable behavior, but one that cannot be controlled by imitation learning directly. In case of RL, we could simply give a negative reward to the agent for all unnecessary interactions, making the agent explicitly learn that such behavior is undesired.

4.4 Other noteworthy details

We want to highlight that there were certain baseline models and metrics that we had planned to do in this submission, but could not due to various reasons such as implementation feasibility, compute limits, etc. We explain specific reasons why those baselines and metrics could not be computed in the appendix Section 7.2.

Moreover, we want to highlight that we built a full end to end pipeline to perform our analysis, which contains clean, refactored and generalizable versions of the baseline Seq2Seq model (Shridhar et al., 2020a) and MOCA (Singh et al., 2020). The current implementation of the baseline Seq2Seq model (Shridhar et al., 2020a) is quite complicated and very tightly intertwined with specific other code in their library. These factors implicitly force all other ALFRED models to be coded up in a very specific manner, so they can use the evaluation scripts provided by the original ALFRED authors. For example, the MOCA model is also implemented using same ALFRED code because of tight coupling within different parts in that code base (such as dependence of models on specific styles of data preprocessing). We feel this is suboptimal because such coding styles hinder generality. For example, it is not (easily) possible in the original code base to train the same proposed baseline model as a policy network using reinforcement learning. Therefore, we designed our pipeline that can take any model and potentially train it using both reinforcement learning and/or imitation learning. We want to highlight that the actual training code has not be written because that was not the goal of this milestone, but the interface is generic enough to support it. We highlight important features of our pipeline in appendix Section 7.3.

4.5 Proposed Approach

4.5.1 Motivation

We noticed that in all proposed ALFRED models, there is a significant reliance on phenomena that the authors expect would automatically emerge without explicit enforcement, and little focus on grounding of different modalities. In the following, we identify concrete instances of such over-reliance on emergent phenomena and little focus on grounding.

- **Too much reliance on LSTM hidden state.** In the original baseline model (Shridhar et al., 2020a) (see Figure 10), the LSTM's (shown in green in the figure) hidden state's role is to keep track of current context (as it is used to attend over instructions). However, to perform well in the task, a lot of information needs to be included in the "context". For example, "context" needs to include some action history, interaction history, instruction understanding, goal understanding, and image understanding with respect to the goal and instruction, among other things. It seems quite unlikely that such a rich representation will automatically emerge in the hidden state, especially without any enforcement. Note that such an argument can be made even for the LSTMs used in the MOCA (Singh et al., 2020) architecture.

	Validation				Test			
	Seen		*Unseen*		*Seen*		*Unseen*	
Model	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond	Task	Goal-Cond
Seq2Seq + PM Both (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)
Modular (Corona et al., 2020)	-	-	-	-	-	8.80 (6.30)	-	7.20 (5.70)
MOCA (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)
Ours								
Human	-	-	-	-	-	-	91.00 (85.80)	94.50 (87.60)
Ablations								
No Language								
No Vision								
Goal-Only								
Instructions-Only								
Trained using IL only								

Table 3: **Task and Goal-Condition Success Rate.** Corresponding path-weighted metrics are given in parentheses.

	Seen						*Unseen*					
Model	Nav-Obj	First-Nav-Obj	Int-Succ	Bad-Mask	Unnec-Int	Int-Act-Pred	Nav-Obj	First-Nav-Obj	Int-Succ	Bad-Mask	Unnec-Int	Int-Act-Pred
Seq2Seq + PM Both	0.680	0.735	0.494	0.250	1.298	0.131	0.283	0.322	0.148	0.437	0.809	0.008
MOCA	0.778	0.838	0.470	0.617	1.103	0.279	0.586	0.669	0.245	0.737	1.375	0.140
Ours												
Ablations												
No Language												
No Vision												
Goal-Only												
Instructions-Only												
Trained using IL only												

Table 4: Intrinsic Metrics - Validation.

- **Little benefit from previous action.** Both MOCA (Singh et al., 2020) and the baseline model (Shridhar et al., 2020a) (see Figures 11 & 10) use the previous action in their model as input to better provide context to the model. Based on our data analysis, using only the most recent action seems to be ineffective since 60% of actions are just "MoveAhead"; in other words, 60% of the time the model cannot extract very meaningful information from the most recent action alone. Furthermore, for interaction actions, keeping track of the action alone does not provide any information about the object that the agent interacted with. Due to these reasons, the burden of the LSTM to maintain an informative context increases even further because all extra information is expected to be stored in its hidden state.

- **Language not grounded in vision.** Both the baseline model and MOCA do not ground language in vision (MOCA grounds vision in language using dynamic filters, but not vice versa). It seems that grounding language in vision (in addition to grounding language in the current context) is essential because visual input directly can tell the model what is visible in the scene, and what to focus on in language accordingly.

As we will describe, we attempt to solve all above issues in our proposed model.

4.5.2 Notation

Before we discuss our model, we introduce some notation that would make describing the model easier:

- G denotes the natural language description of the high level goal of the task. So, $G = [g_1, \ldots, g_{|G|}]$ where g_i denotes the i^{th} word.

- L denotes the list of step-by-step natural language instructions. So, $L = [L_1, \ldots, L_{|L|}]$ such that each L_i is the i^{th} instruction, and $L_i = [L_{i1}, \ldots, L_{in}]$ where L_{ij} denotes the j^{th} word of the i^{th} instruction.

- I_t is the RGB image from the simulator at the t^{th} time step.

- (a_t, O_t) is a tuple such that a_t denotes the action taken at time step t, and O_t denotes the class of the object upon which the action was applied in case the action was an interaction action.

- M_t is the interaction mask corresponding to particular instance of object from class O_t.

- O_{I_t} denotes the list of object names visible in image I_t. So, $O_{I_t} = \{O^1_{I_t}, \ldots, O^k_{I_t}\}$ where $O^i_{I_t}$ is the i^{th} object in the scene (with no particular order). As will be described later, O_{I_t} is retrieved by running instance segmentation (He et al., 2017) on I_t and getting object class names.

4.5.3 Proposed Idea

Using the described notation, Figure 6 shows the complete architecture, and we explain each element and modelling choice below.

- **Disentangling Explicit & Implicit Context.** An important novel contribution of our method is to significantly lighten the load of each LSTM. As described in Section 4.5.1, current architectures expect too much from their LSTMs. To this end, we provide some easily deducible history directly to our model, so the LSTM can only focus on learning a more latent and abstract form of context. This lets us essentially disentangle the explicit and implicit forms of context, which should make learning much simpler.

  Specifically, we intend to pass the last k actions and the last k' objects the agent interacted with as input. As shown in the context module of Figure 6, we will use these two inputs to construct a representation of "explicit context". We can combine this "explicit context" with the hidden states of our two LSTMs (which together form a representation of "implicit context") that will be explained below to form an overall context vector. We can then use this overall context vector to perform attention over natural language instructions and goal as shown in the figure. The intuition behind passing the last few actions and interaction objects is that any agent would need to keep track of its last few actions and interactions to have a good understanding of what its doing, so it makes more sense to just directly provide this history.

- **Grounding Language in Vision.** We propose to use a novel multimodal method to ground language in vision. Our insight is that it would be much harder for the model to align a raw visual representation (obtained from running a simple CNN on the input image) with a language representation because vision and language features are inherently different. If we can represent the image well using language, it would be much easier to align it with the instructions because both would be sequences of linguistic features. Our solution is to run an instance segmentation model on the input image I_t and identify all objects in the image (represented by $O^1_{I_t}, \ldots, O^k_{I_t}$ in the vision module of Figure 6). Since these objects are just words, we can embed them using pre-trained word embeddings and then run these embeddings through a transformer (without positional encodings because there is no inherent order) to get an overall representation (called "object embedding" in Figure 6). As shown in the figure, we use these embeddings and raw vision features to get "overall vision features", which are later used to attend over the natural language instructions as shown in the Action Module. We feel that language grounding would be much easier if we directly identify all objects in the image by their names because a) we are aligning language features with language features instead of vision features with language features, b) we are directly providing all objects in the scene as input, thereby significantly reducing the load of the CNN that outputs the "raw vision features" in Figure 6. While we thought of this idea independently, we would like to point out that the idea of using object labels has been used in a previous ALFRED model (Storks et al., 2021), but in a different way. While Storks et al. (2021) used this idea in a small module which was used to input additional features to the original baseline model (Shridhar et al., 2020a), our idea has no dependence on the baseline model.

- **Two LSTMs.** Instead of using only one LSTM as in the baseline model (Shridhar et al., 2020a), we propose to use two different LSTMs: one to keep track of instructions, and another to keep track of the high level goal description as shown in the Action Module of Figure 6. Having two LSTMs again significantly reduces the responsibility of each because a clear disentanglement can be achieved between low level instruction following and high level goal completion. As shown in the figure, the low level "instruction LSTM" is responsible for action and interaction object

prediction based on overall context, vision features, and attended low level natural language instructions. The "goal LSTM" is then updated using the predicted action and object, and the natural language goal description is attended based on the overall context. Note that MOCA also uses two LSTMs as shown in Figure 11, but the LSTMs do not interact with each other, and they serve a very different intuitive purpose (keep track of perception and policy) than the two LSTMs in our model (keep track of overall goal and low level instructions).

- **Interaction Mask Prediction.** We follow MOCA's (Singh et al., 2020) approach of disentangling object class prediction and instance mask prediction. This makes the action predictor's job significantly easier because it only needs to identify the class of the object it needs to interact with (which often would be mentioned directly in natural language instructions) instead of identifying the object class, identifying the object in the current image, and then predicting a segmentation mask around the object. We plan to use MOCA's exact method of associating the actual object instance from the predicted object class using confidence and association-based measures.

- **Mini Transformer + Readout** At various places in our model, we use a mini transformer to obtain one cumulative representation out of a given set of representations. We use the word "mini" to denote that we just intend to use 2-3 layers of the transformer instead of all 12. As shown in Figure 6, we use an additional input (denoted by $\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3$) in all mini transformers. $\mathcal{R}_i$ is simply a learned vector added to each transformer, and serves the same role as a classification token does in BERT (Devlin et al., 2018). The additional token is just used to *readout* a vector (denoted by $\hat{\mathcal{R}}_i$) from the output of the transformer, which can then be used for downstream processing. This mini transformer + readout block is used whenever we want the input sequence to be processed with respect to other elements of the sequence. For example, it is important to process past actions with respect to each other so that the model knows exactly what the agent was trying to do in the past.

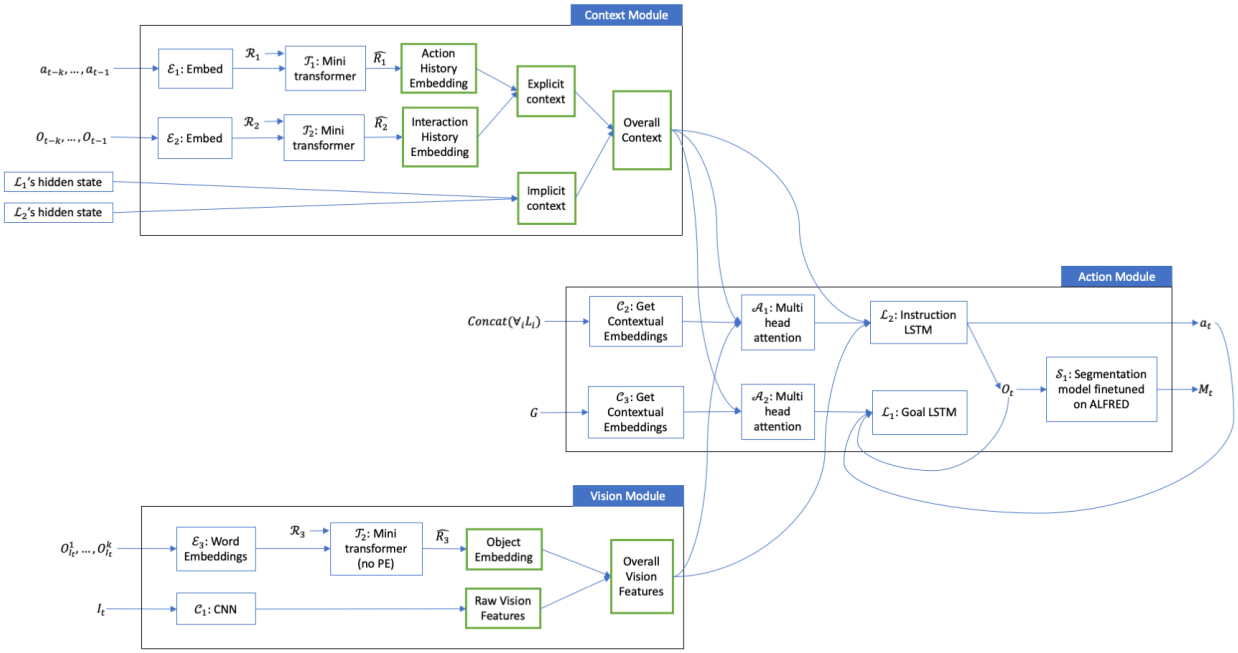

Figure 6: Our proposed model consists of three novel modules as shown which are intended to provide better overall structure, stronger inductive biases, and more prior information to make it easier for each part of the model to learn. The notation used here is described in Section 4.5.2, and the model is explained in detail in 4.5.3.

5 Results (1 page)

The columns above are just examples that should be expanded to include all metrics and baselines.

6 Analysis (2 pages)

This section should include at least two to three plots

6.1 Ablations and Their Implications

6.2 Qualitative Analysis and Examples

This section should likely contain a table of examples demonstrating how the current approach succeeds/fails.

References

Pieter Abbeel and Andrew Y Ng. 2010. Inverse reinforcement learning.

Ferran Alet, Tomás Lozano-Pérez, and Leslie P. Kaelbling. 2019. Modular meta-learning.

Tadas Baltrušaitis, Chaitanya Ahuja, and Louis-Philippe Morency. 2018. Multimodal machine learning: A survey and taxonomy. *IEEE transactions on pattern analysis and machine intelligence*, 41(2):423–443.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. 2016. Openai gym. *arXiv preprint arXiv:1606.01540*.

Rodolfo Corona, Daniel Fried, Coline Devin, Dan Klein, and Trevor Darrell. 2020. Modularity improves out-of-domain instruction following. *arXiv preprint arXiv:2010.12764*.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, J. Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *CGW@IJCAI*.

Abhishek Das, Samyak Datta, Georgia Gkioxari, Stefan Lee, D. Parikh, and Dhruv Batra. 2018a. Embodied question answering. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 2135–213509.

Abhishek Das, Georgia Gkioxari, Stefan Lee, D. Parikh, and Dhruv Batra. 2018b. Neural modular control for embodied question answering. *ArXiv*, abs/1810.11181.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

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. *arXiv preprint arXiv:1806.02724*.

Michał Garmulewicz, Henryk Michalewski, and Piotr Miłoś. 2018. Expert-augmented actor-critic for vizdoom and montezumas revenge. *arXiv preprint arXiv:1809.03447*.

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.

Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. 2018. Deep q-learning from demonstrations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. *arXiv preprint arXiv:1606.03476*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ahmed Hussein, Mohamed Medhat Gaber, Eyad Elyan, and Chrisina Jayne. 2017. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35.

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

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

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

Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. 2020. Improving vision-and-language navigation with image-text pairs from the web.

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

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Ashvin Nair, Bob McGrew, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. 2018. Overcoming exploration in reinforcement learning with demonstrations. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6292–6299. IEEE.

Duy-Kien Nguyen and Takayuki Okatani. 2018. Multi-task learning of hierarchical vision-language representation.

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*.

Siddharth Reddy, Anca D Dragan, and Sergey Levine. 2019. Sqil: Imitation learning via reinforcement learning with sparse rewards. *arXiv preprint arXiv:1905.11108*.

Stéphane Ross and Drew Bagnell. 2010. Efficient reductions for imitation learning. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 661–668. JMLR Workshop and Conference Proceedings.

Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings.

Homagni Saha, Fateme Fotouhif, Qisai Liu, and Soumik Sarkar. 2021. A modular vision language navigation and manipulation framework for long horizon compositional tasks in indoor environment. *ArXiv*, abs/2101.07891.

Tim Salimans and Richard Chen. 2018. Learning montezuma's revenge from a single demonstration. *arXiv preprint arXiv:1812.03381*.

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*, pages 10740–10749.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew J. Hausknecht. 2020b. Alfworld: Aligning text and embodied environments for interactive learning. *ArXiv*, abs/2010.03768.

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*.

Shane Storks, Qiaozi Gao, Govind Thattai, and G. Tür. 2021. Are we there yet? learning to localize in embodied instruction following. *ArXiv*, abs/2101.03431.

Weiyao Wang, Du Tran, and Matt Feiszli. 2020. What makes training multi-modal classification networks hard?

Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. 2019. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6629–6638.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 37–53.

Saim Wani, Shivansh Patel, Unnat Jain, Angel X Chang, and Manolis Savva. 2020. Multion: Benchmarking semantic map memory using multi-object navigation. *arXiv preprint arXiv:2012.03912*.

Chen Yu and Dana H. Ballard. 2004. On the integration of grounding language and learning objects.

Licheng Yu, Xinlei Chen, Georgia Gkioxari, Mohit Bansal, T. Berg, and Dhruv Batra. 2019. Multi-target embodied question answering. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6302–6311.

7 Appendix

7.1 Data Analysis Plots

Description of fields in Table 1:

1. Steps per directive: number of steps in each language directive
2. Tokens per step: number of words in each directive step
3. Task description tokens: number of words in directive task description
4. Images: number of images per demonstration
5. Actions: number of actions per demonstration
6. Images per action: number of images divided by number of actions per demonstration
7. Actions per step: number of actions divided by number of directive steps per demonstration
8. Nav-interact ratio: number of navigation actions divided by number of interaction actions per demonstration
9. Total objects: number of total objects in a scene per demonstration
10. Mask coverage: proportion of the image that is covered by the interaction mask per demonstration
11. Step-object coverage: proportion of interaction actions whose object of interest is mentioned in the step-by-step instructions, averaged over all interaction actions and language directives in the demonstration

In Figure 9, we see that ALFRED contains a roughly equal number of demonstrations for each type of task, and for the most part, a roughly equal proportion for each split. The validation data, especially the unseen portion, does have relatively less "Pick Two & Place" tasks than the training data. Additionally, the unseen portion has a significantly higher proportion of "Examine in Light" tasks than the other splits.

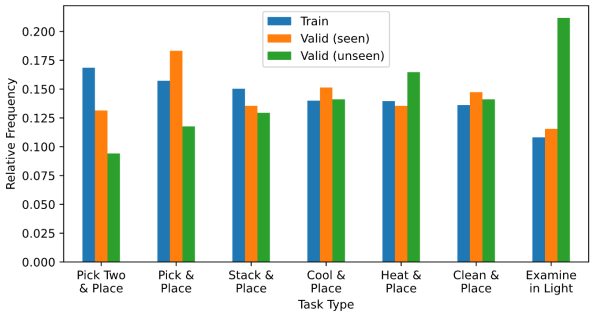

Figure 9: Relative frequency of each task type by split

7.2 Hard to compute baselines/metrics

In the following, we list out all baselines/metrics that we planned to run, but could not due to reasons listed below.

- **Seq-2-Seq (CNN-LSTM)** (Shridhar et al., 2020a) We did not evaluate the vanilla CNN-LSTM Seq2Seq architecture because a) we evaluated the progress monitoring variant that gets better performance, and b) the pre-trained model for this was not provided and we wanted to spend our limited compute on better performing models like MOCA (Singh et al., 2020).
- **Modular Seq-2-Seq** (Corona et al., 2020) We did not evaluate our internal metrics on the modular Seq2Seq architecture because their code was not released.
- **Seq-2-Seq RL (proposed):** Seq2Seq RL was an architecture we had originally proposed which basically used reinforcement learning to train the policy network which was same as the Seq2Seq PM architecture. However, we were not able to evaluate on this because of several reasons. One, we did some calculations on GPU compute and realized that training using RL would be very expensive which we would not be able to afford (at least not on models that we have not built). Two, we felt that the reward function would need to be heavily tuned to get any satisfactory performance, which would take a lot of time and probably not be considered a "baseline" model since we would have heavily modified it. Three, many libraries/implementations did not support recurrent policies which was just another challenge in implementing this.
- **Interaction Mask Prediction Performance (IMPP)**: We initially defined IMPP as the in-

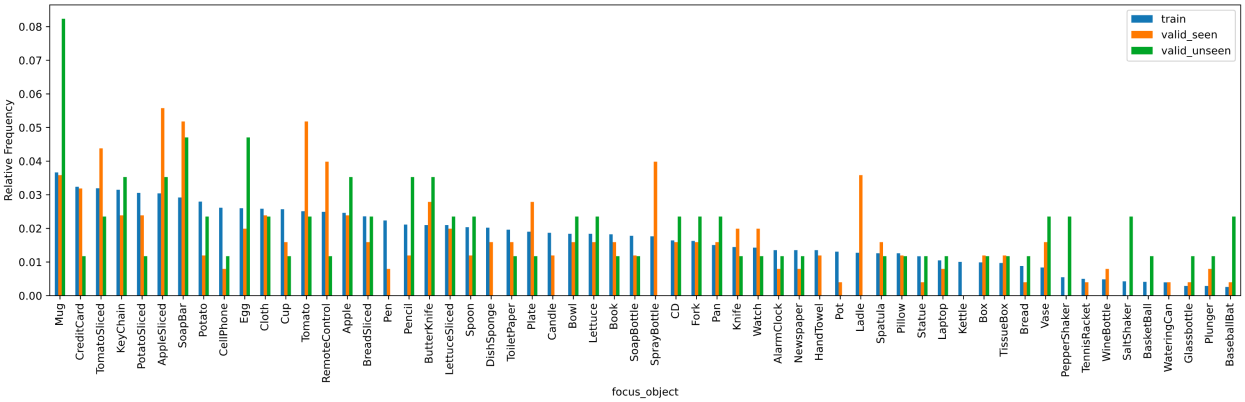

Figure 7: Relative frequency of focus objects used in different splits

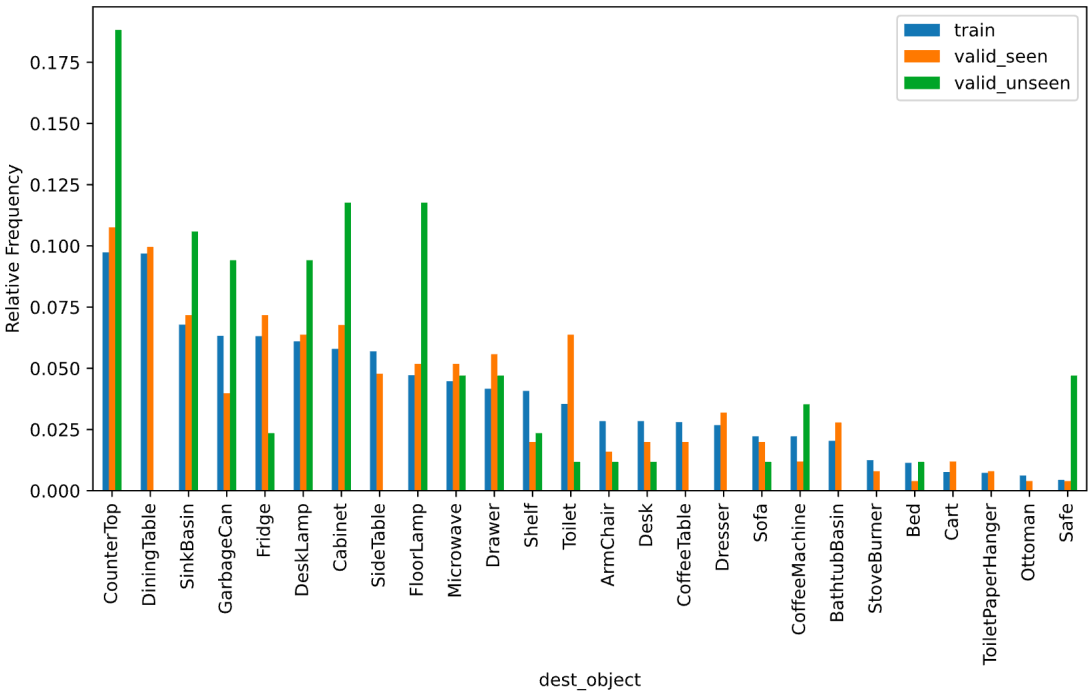

Figure 8: Relative frequency of destination objects used in different splits

tersection over union score of the predicted instance segmentation mask with respect to the ground truth mask when the interaction action was correct. The idea was that conditioning on correctness of interaction action would make IMPP only measure the performance of the component that predicts the interaction mask. However, we could not compute this because there was no objective, non-heurestic based way to identify which object the agent intended to interact with at which time. Just matching expert action and predicted at each time step would be too naive because it is unlikely that the agent would mimic the expert so closely.

- **Length-conditioned Success (LCS)**: We initially wanted to compute LCS, which was supposed to be the first n sub-goals successfully completed continuously. However, we could not compute this because the simulator does not store the ordering of the sub-goals, so we could not tell which sub-goals were first and which were last.

7.3 Pipeline features

1. Our pipeline uses very strong software engineering and object oriented design principles and strongly decouples all different parts that are logically distinct. Such decoupling allows models in our pipeline to be run without relying on any preprocessing or external dependencies.

2. Currently, our pipeline supports both the baseline Seq2Seq model and MOCA under a unified interface, which will not only help us compare our model with them, but also all future researchers working on ALFRED who want to compare their models to other SOTA models.

3. We also have built a Gym style (Brockman et al., 2016) environment for ALFRED, making it amenable to be trained using reinforcement learning easily.

4. A strong component in helping us keep our code very clean is our custom designed solution for configuration management, which keeps all configuration variables and utility functions separate from the main code, essentially helping us achieve disentanglement between logically separate parts.

7.4 Previous ALFRED Models

Figures 10 and 11 show previously proposed architectures.

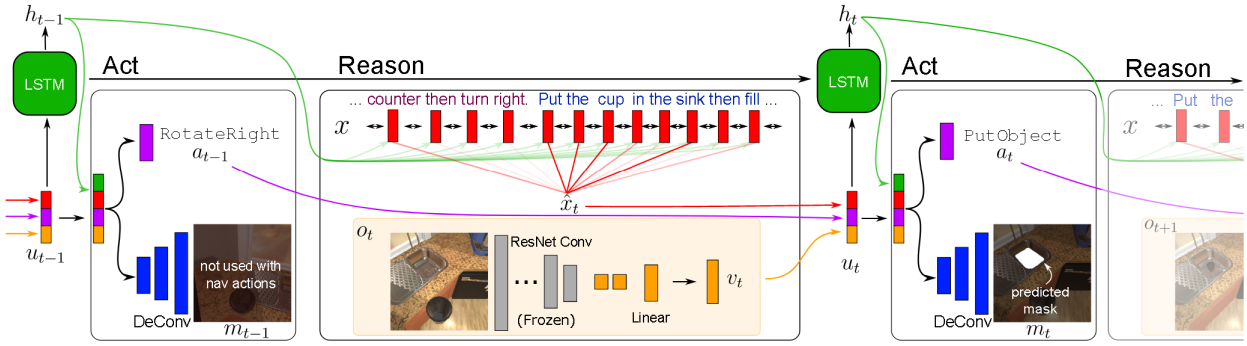

Figure 10: ALFRED Baseline Model (Shridhar et al., 2020a)

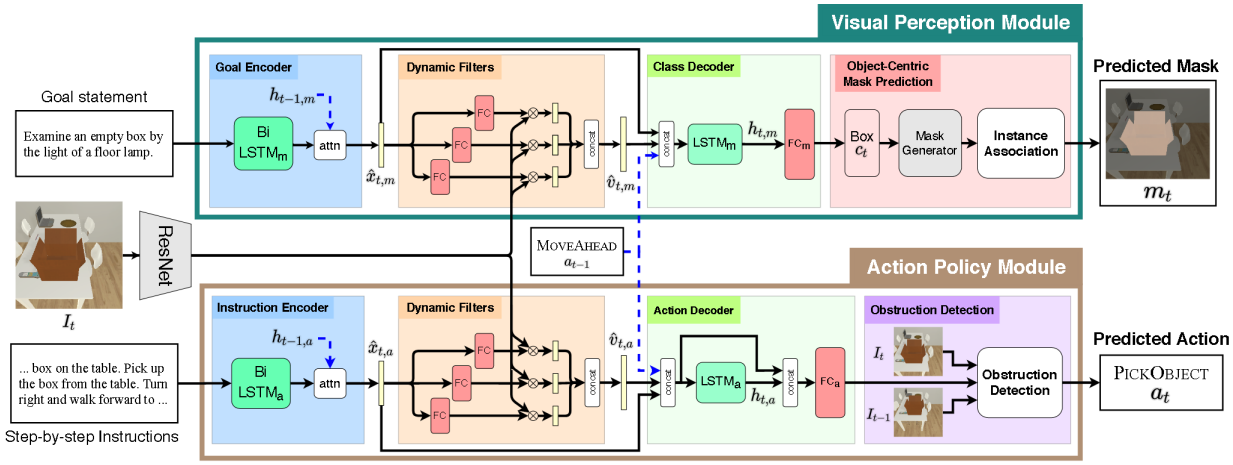

Figure 11: MOCA Model (Singh et al., 2020)