

Imitation Learning for Sequential Manipulation Tasks: Leveraging Language and Perception

by

Dain Kim

S.B., Computer Science and Engineering, MIT (2020)

Submitted to the Department of Electrical Engineering and Computer
Science

in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2021

© Massachusetts Institute of Technology 2021. All rights reserved.

Author
Department of Electrical Engineering and Computer Science
May 31, 2021

Certified by.....
Julie A. Shah
Associate Professor of Aeronautics and Astronautics
Thesis Supervisor

Certified by.....
Nadia Figueroa
Postdoctoral Associate
Thesis Supervisor

Accepted by
Katrina LaCurts
Chair, Master of Engineering Thesis Committee

Imitation Learning for Sequential Manipulation Tasks: Leveraging Language and Perception

by

Dain Kim

Submitted to the Department of Electrical Engineering and Computer Science
on May 31, 2021, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

As robots are increasingly being utilized to perform automated tasks, effective methods for transferring task specifications to robots have become imperative. However, existing techniques for training robots to perform tasks often depend on rote mimicry of human demonstrations and do not generalize well to new tasks or contexts. In addition, learning an end-to-end policy for performing a sequence of operations for a high-level goal remains a challenge. Transferring sequential task specifications is a difficult objective, as it requires extensive human intervention to establish the structure of the task including the constraints, objects of interest, and control parameters.

In this thesis, we present an imitation learning framework for sequential manipulation tasks that enables humans to easily communicate abstract high-level task goals to the robot without explicit programming or robotics expertise. We introduce natural language input to the system to facilitate the learning of task specifications. During training, a human teacher provides demonstrations and a verbal description of the task being performed. The training process then learns a mapping from the multi-modal inputs to the low-level control policies. During execution, the high-level task instruction input is parsed into a list of sub-tasks that the robot has learned to perform.

The presented framework is evaluated in a simulated table-top scenario of a robotic arm performing sorting and kitting tasks from natural language commands. The approach developed in this thesis achieved an overall task completion rate of 91.16% on 600 novel task scenes, with a sub-task execution success rate of 96.44% on 1,712 individual “pick” and “place” tasks.

Thesis Supervisor: Julie A. Shah

Title: Associate Professor of Aeronautics and Astronautics

Thesis Supervisor: Nadia Figueroa

Title: Postdoctoral Associate

Acknowledgments

I would like to thank Professor Julie Shah for the opportunity to work in the Interactive Robotics Group for the past year. I also extend my sincerest gratitude to my supervisor, Nadia Figueroa, who provided tremendous feedback and guidance throughout this whole process. This work would not have been possible without both of their invaluable mentorship.

I have had the pleasure of teaching the MIT Interactive Music Systems course (6.809/21M.385) with Professor Eran Egozy in the past year. Working with him was an immensely rewarding experience. I would also like to acknowledge my academic advisor Adam Chlipala, my international student advisor Aurora Brule, and Ellen Reid from the Undergraduate EECS Office for their administrative support.

I was fortunate to have been surrounded by great friends and colleagues during my time at MIT. In particular, I want to thank my roommates Meital Hoffman and Andy Reyna for being wonderful company during a most unusual year. I thank the tEp community for making my undergraduate experience truly memorable and enjoyable. I also extend my thanks to Charlie Yeoh, who has been with me every step along the way and provided endless encouragement and advice.

Most importantly, I am incredibly grateful for my family back home in Korea. They have been my rock throughout my five years at MIT and more, and I would not be where I am today without their unconditional love and support.

Contents

1	Introduction	15
1.1	Motivation	15
1.2	Language-Conditioned Imitation Learning	18
1.3	Therbligs: Elemental Motions for Workplace Tasks	19
1.4	Experimental Validation and Results	20
2	Related Work	23
2.1	Representation Learning	23
2.2	Grounding	24
2.3	Task-Oriented Language Grounding	26
3	Approach	29
3.1	Modifications to Prior Work	29
3.2	Architecture Overview	32
3.2.1	Image Pre-processing	32
3.2.2	High-Level Semantic Command Parser	33
3.2.3	Language Pre-Processing	37
3.2.4	Language-Conditioned Attention Network	37
3.2.5	Sub-Task Sequencer	37
3.2.6	Control Model	39
4	Evaluation	41
4.1	Methods	41

4.1.1	Training	41
4.1.2	Test Data Collection	42
4.1.3	Performance Metrics	43
4.2	Results	46
4.2.1	Sub-Task Generation	46
4.2.2	Object Detection	47
4.2.3	Attention	49
4.2.4	Task Execution	51
4.3	Discussion and Limitations	55
4.3.1	Parser	55
4.3.2	Detection	55
4.3.3	Task Sequencing	56
4.3.4	Grasping	56
4.3.5	Motion Primitive	56
5	Conclusion	59

List of Figures

1-1	A simplified architecture of the proposed language-conditioned imitation learning framework.	18
1-2	The 18 workplace elemental motions (therbligs) and the standard symbols used to describe them. Illustration obtained from [16].	20
1-3	An illustration of the sorting and kitting task scenarios used in evaluation.	21
1-4	A bird’s-eye view of the simulation environment and the available objects.	22
3-1	Model architecture implemented in [46] for the imitation learning of <i>single</i> target-oriented robot manipulation tasks from goal-specific instructions.	31
3-2	Model architecture proposed in this work for the imitation learning of <i>sequential</i> robot manipulation tasks from high-level abstract instructions.	31
3-3	Visualization of the result of candidate object detection. The integer on the top left of each region indicates the detection model’s internal object index, and the score on the top right shows the detection confidence. The bounding boxes are color-coded for convenience. . . .	33
3-4	The semantic tree for a command with the structure V O	35
3-5	The semantic tree for a command with the structure V O L	35
3-6	The semantic tree for a command with the structure V O CC V O L .	36
3-7	The semantic tree for a command with the structure V O CC O L . .	36
3-8	Atomic sub-task commands generated by the semantic parser based on the detected task objects and the high-level instruction.	36
3-9	A detailed diagram of the language-conditioned attention network. . .	38

3-10	A finite state machine generated for the k sub-tasks $\{e_1 \dots e_k\}$ in Φ . . .	39
3-11	Details of the control model, which generates the robot control signals. This figure was taken from [46].	40
4-1	Examples of four randomly generated sorting task scenes and their corresponding task commands, roughly in the order of increasing task complexity.	43
4-2	Examples of four randomly generated kitting task scenes and their corresponding task commands, roughly in the order of increasing task complexity.	44
4-3	Sample task execution sequences for sorting (left) and kitting (right) tasks, with the original task command and the current sub-task com- mand being performed.	45
4-4	A confusion matrix for the object detection module. The module clas- sified the 3,600 task objects with 99.7% accuracy, with very few clas- sification failures.	48
4-5	A sorting task with a false positive produced in the object detection stage. The red cup on the right is detected twice by the detection module, for a total of three perceived red cups in the scene. The parser generates three pick-place sub-task pairs based on this information, which ultimately leads to the robot attempting three pick tasks in total.	49
4-6	Two examples of bin detection failures caused by occlusion. In the first scene, the robot arm at the end of the previous sub-task obscures the target bin. In the second scene, the red cup placed during the previous sub-task interferes with the detection of the bin. Since no target bin is identified, the model performs the subsequent place operation on a different bin with the incorrect color.	50
4-7	A confusion matrix for the attention network module. The module identified the 1,990 target objects with 81.1% accuracy.	52
4-8	Attention network accuracy for 1,148 pick and 842 place target objects.	52

4-9	Attention network accuracy for 261 generic and 1,451 specific task commands.	53
4-10	Examples of correctly executed place tasks.	54
4-11	Examples of place task inaccuracies during execution. These tasks were considered to be successful if the gripper was directly above the bin at release time.	55

List of Tables

3.1	A complete list of part-of-speech tags used in this work.	34
4.1	Model performance summary of the proposed language-conditioned sequential task learning framework.	47
4.2	Sub-task execution success rates for pick and place tasks.	54
4.3	Task completion rates for the sorting and kitting task scenarios. . . .	54

Chapter 1

Introduction

1.1 Motivation

As robots are increasingly being utilized in factory settings to perform automated tasks, effective methods for communicating task specifications to robots have become imperative. The most direct approach for achieving this is to manually program the robot to execute the desired behavior. This is a very time-consuming process, as it requires considerable effort and expertise to design, code, and test every step of the task. Moreover, as the task environment becomes more complex and variegated, explicitly programming the exact actions for each task quickly becomes impractical.

A widely accepted alternative is to use imitation learning (IL) [36] or learning from demonstration (LfD) [26, 8, 5], techniques that train the robot to mimic human behavior for a given task. The model observes demonstrations from a human teacher and is trained to perform the task by learning a mapping $\pi : \mathcal{X} \rightarrow \mathcal{Y}$ between input observations $\mathcal{X} \in \mathbb{R}^M$ and output actions $\mathcal{Y} \in \mathbb{R}^N$. The mapping function π is often referred to as the policy, which is learned from a training set of input-output $(\mathcal{X}, \mathcal{Y})$ pairs. The actions are formulated as motor signals that the robot must execute – normally desired positions, velocities, or torques, depending on the robot. The input observations may be in the form of:

- Perceptual inputs such as a raw image of the task scene $\pi : (\mathcal{X} = I) \rightarrow \mathcal{Y}$,

where $I \in \mathbb{R}^{P \times Q \times 3}$ with $P, Q \in \mathbb{N}$ denoting the size of the image.

- Processed image features or locations of the task objects $\pi : (\mathcal{X} = F) \rightarrow \mathcal{Y}$, where $F \in \mathbb{R}^{R \times K}$ with $R, K \in \mathbb{N}$ denoting the feature dimensionality and number of objects.
- The robot states $\pi : (\mathcal{X} = r) \rightarrow \mathcal{Y}$, where $r \in \mathbb{R}^N$ is the robot state which may or may not have the same dimensionality as the action space. For simplicity, in this work we assume that they are the same; i.e. $r, \mathcal{Y} \in \mathbb{R}^N$ where N is the number of degrees-of-freedom of the robot.

An overview of the different observation modalities is provided in [38]. Regardless of the observation modalities, IL approaches provide a more natural and intuitive way to describe tasks to the agent. It is much easier for the human teacher to demonstrate a task physically or kinesthetically (by moving the robot) than to articulate it in a highly specific language that only the robot and the programmer can understand.

Using the imitation learning framework, robots have been able to learn control policies for specific, singular goal-oriented tasks including pick/grasp, pour, and place with relative ease [39]. However, teaching robots to perform multiple atomic operations in sequence, guided by a desired metric or high-level task goal, is still an open challenge within the end-to-end imitation learning framework. This is difficult to achieve since it requires heavy guidance from a human expert to manually specify the set of rules, constraints, objects of interest, and control parameters for the high-level task in order to ultimately execute the low-level control primitives. Thus, imitation learning only partially addresses the issue of tedious programming when it comes to learning complex sequential tasks. Most existing works that tackle the sequential task learning problem forgo this extensive level of human input required for low-level control and instead focus on learning only the high-level task specifications (i.e., high-level goal and sequence of actions or sub-tasks) [44, 18]. These approaches rely on pre-defined or pre-learned control policies to define the low-level behavior and do not provide a human-understandable communication interface to define or adapt the desired task specifications other than by providing another demonstration. This, again,

becomes cumbersome for the user. Further, since imitation learning schemes learn from a specific set of training data, it is often difficult to generalize well to tasks that the robot has not been trained on. If the robot is simply imitating human behavior without a shared conceptual understanding of the task, it cannot perform tasks that are more abstract in nature and involve reasoning about the task environment.

Recent efforts to address the aforementioned limitations of imitation learning have benefited from introducing another form of input: *language* [46, 42]. In addition to demonstrating the task, the human also provides a natural-language task description or command, which introduces context about the properties of the task objects and the environment. Hence, the learned mapping function becomes $\pi : (\mathcal{X} = [I, v]) \rightarrow \mathcal{Y}$, where $I \in \mathbb{R}^{P \times Q \times 3}$ indicates a raw image of the scene and v is an unstructured natural language command. With the help of this additional language element, the model is able to develop a more comprehensive understanding of the underlying structure of the world that cannot be attained simply from demonstrations, which allows it to generalize better to new environments. The shared conceptual language also serves to facilitate communication between the robot and human.

With the addition of language, the robot is able to interpret the task environment in a more semantically meaningful manner grounded in real-world concepts. But the rigidity of the communication language limits the complexities of the tasks that can be performed. For example, in existing works with table-top object manipulation scenarios [46, 33, 6], the robot must be given explicit instructions regarding which action to perform and objects to manipulate (e.g. “grab the red cup,” “place the cup into the yellow bin”, etc.), rather than being able to deduce the task from a more abstract or natural command (e.g. “put all the red and blue objects in the yellow bin”). To enable a more intuitive transferral of abstract tasks, the robot should be able to parse the complex task command into a series of simpler sub-tasks that achieves the final goal. Endowing robots with such capabilities, by leveraging perception and language, is the main focus of this thesis.

1.2 Language-Conditioned Imitation Learning

We expand upon a state-of-the-art language-conditioned imitation learning approach [46] that learns to perform single, discretized manipulation tasks such as pick and pour from a goal-oriented natural language instruction (e.g., “pick up the blue cup”). While we maintain the general structure of the learned policy $\pi : (\mathcal{X} = [I, v]) \rightarrow \mathcal{Y}$, our contribution extends it with the ability to decompose abstract instructions (e.g., “place all the blue cups in the red bin”) into elemental operations that the model jointly learns to perform. We accomplish this by introducing a high-level semantic model that parses the abstract task command into a sequence of goal-oriented commands.

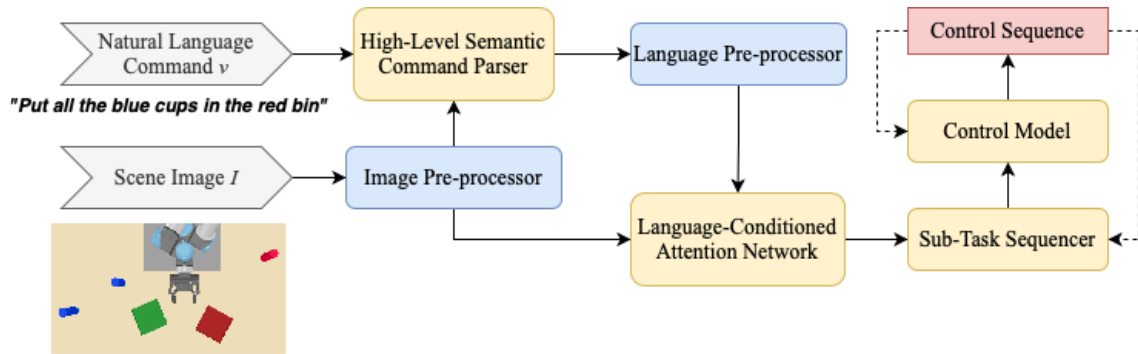


Figure 1-1: A simplified architecture of the proposed language-conditioned imitation learning framework.

Our framework builds upon the semantic-to-control dual model structure proposed by [46] and takes inspiration from the architecture proposed in [42] for conditional driving from natural language instructions. The proposed architecture is composed of four main modules: high-level semantic command parser, language-conditioned attention network, sub-task sequencer, and low-level control model. The semantic command parser assesses the different linguistic components of the natural language input instruction and generates a sequence of simpler sub-task commands. Each sub-task command focuses on a single elemental action to be taken by the robot. The attention network combines the reduced commands and the image of the task scene to generate a task embedding for each sub-task. The task embeddings are then executed in order by the sub-task sequencer via an automatically generated FSM, which calls on the lower-level control model to generate the action controls required to achieve

each sub-task. An overview of the proposed architecture is illustrated in Figure 1-1. A more granular depiction and description of all components in the architecture is detailed in Section 3.2. Using our framework, the robot is able to handle commands it previously could not understand and exhibits improved performance compared to [46] due to the introduction of semantic linguistic structure.

1.3 Therbligs: Elemental Motions for Workplace Tasks

In a pick-and-place setup, the tasks that we expect the robot to perform are comparable to the basic object manipulation motions used by human workers in factories. Thus, in this work, we define the atomic units of actions of our robot in the context of *therblig* motions: 18 elemental motion elements that make up a set of fundamental motions required for a worker to perform a manual operation or task [3]. Therbligs, as depicted in Figure 1-2, are frequently used in the study of motion economy in the workplace. A workplace task is analyzed by recording each of the therblig units for a process, and the results are used for the optimization of manual labor by eliminating unneeded movements. We assume that the instruction coming from the human will contain implicit requirements to perform a combination of these motions. As a proof-of-concept implementation, we focus on only a subset of the therbligs which constitute the “pick” and “place” tasks.

1. Pick = Search + Find + Transport Empty + Grasp
2. Place = Search + Find + Transport Loaded + Position + Release Load



















 Search	 Use
 Find	 Disassemble
 Select	 Inspect
 Grasp	 Preposition
 Hold	 Release Load
 Transport Loaded	 Unavoidable Delay
 Transport Empty	 Avoidable Delay
 Position	 Plan
 Assemble	 Rest

Figure 1-2: The 18 workplace elemental motions (therbligs) and the standard symbols used to describe them. Illustration obtained from [16].

1.4 Experimental Validation and Results

We establish our framework in a table-top environment with a seven-degree-of-freedom robotic arm anchored to a flat workspace. The robot is instructed in natural language commands to arrange objects from a cluttered formation into distinct bins. We use pre-trained weights from [46] to map the generated task embeddings to the attention network and the low-level control model. The performance of the proposed imitation learning architecture was evaluated on two industry use cases: *sorting* and *kitting*, depicted in Figure 1-3. In *sorting*, the robot is instructed to place all objects matching the given description in a bin. Some examples of *sorting* natural language commands include:

- “Put all the cups in the bin.”
- “Put all the red cups in the red bin.”

In *kitting*, the robot is given a pick-and-place task involving a list of objects

described in the instruction, and expected to arrange those objects into a bin.

- “Put two blue cups in the yellow bin.”
- “Put one red cup and one blue cup in the green bin.”

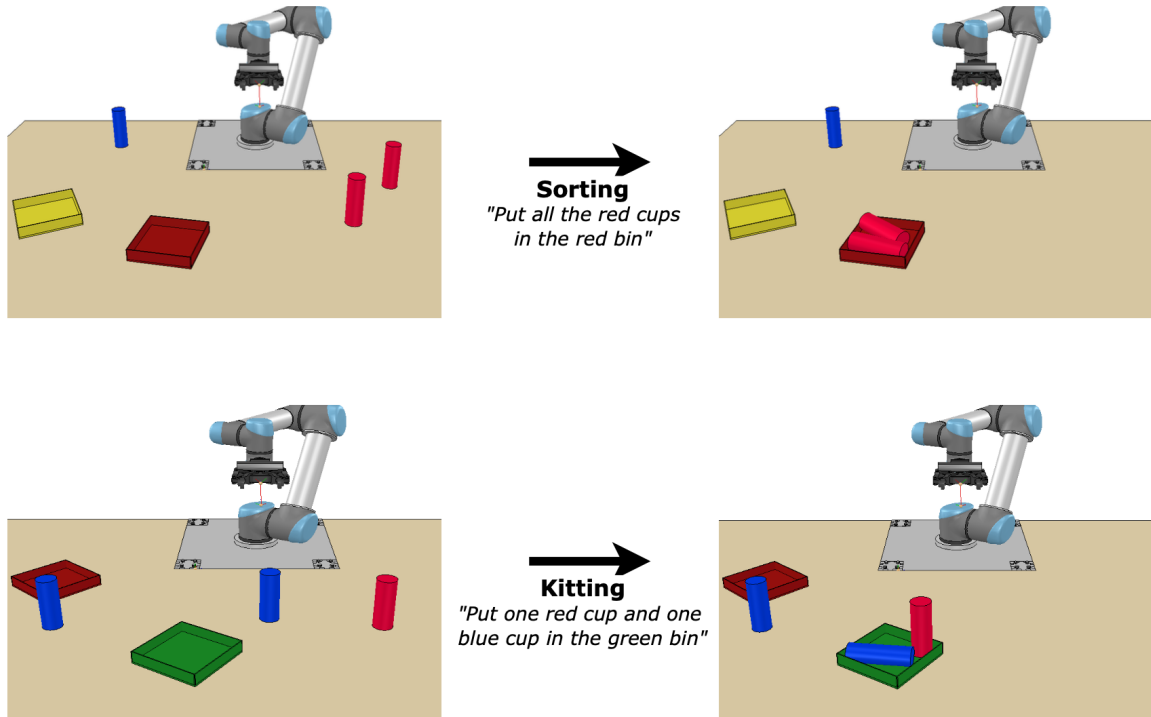


Figure 1-3: An illustration of the sorting and kitting task scenarios used in evaluation.

Both experimental scenarios were run in a simulated table-top environment using the CoppeliaSim simulator [43, 23]. Figure 1-4 depicts the simulation environment and the different objects used. The proposed framework was evaluated on 600 unseen task scenarios and achieved an overall task completion rate of 91.16%. 1,712 individual “pick” and “place” tasks were performed, with a sub-task execution success rate of 96.44%. The evaluation methodology and results are discussed in further detail in Chapter 4.

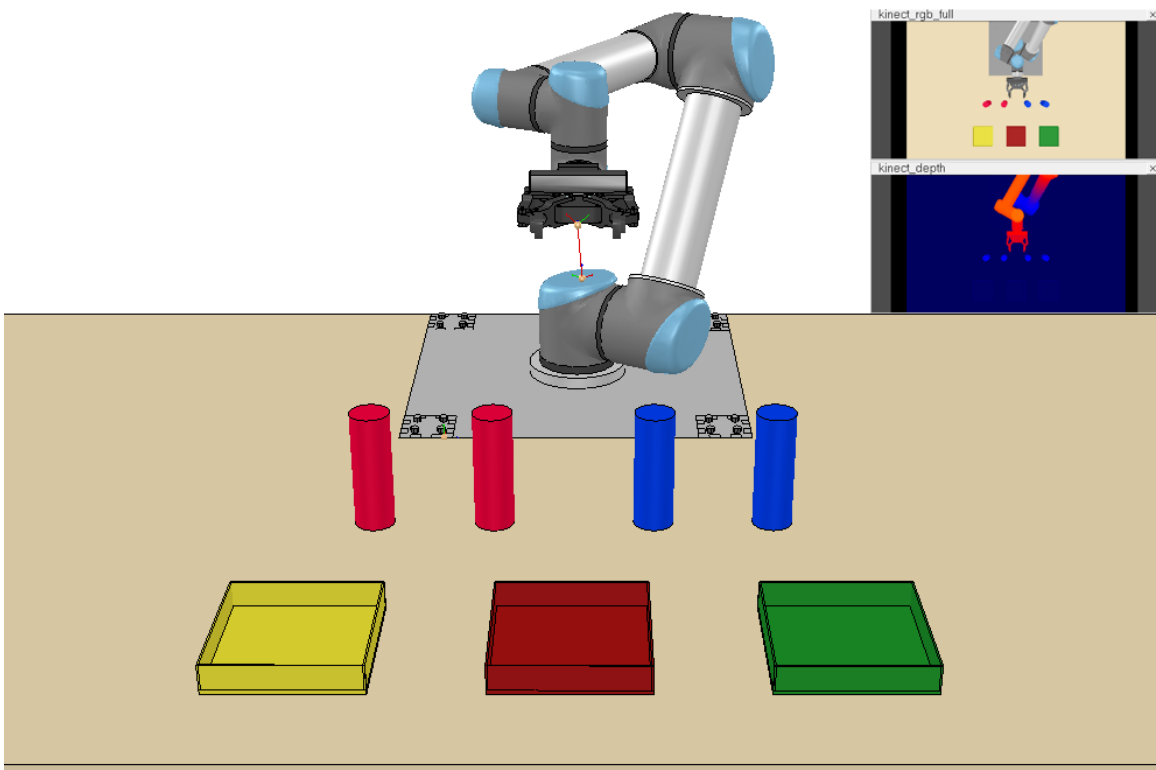


Figure 1-4: A bird's-eye view of the simulation environment and the available objects.

Chapter 2

Related Work

The key challenge that we tackle in this work is creating task representations that integrate perception and language to control a robot with natural language commands. This chapter presents an overview of previous literature and key concepts that are related to this framework of language-conditioned imitation learning. Section 2.1 presents a brief introduction to representation learning techniques in domains with structured inputs such as ours. Section 2.2 introduces the concept of grounding and reviews prior work in robotics that utilizes grounding to improve performance. A special emphasis is placed on reviewing existing works on task-oriented language grounding in Section 2.3.

2.1 Representation Learning

An agent needs to have a solid understanding of its environment in order to make intelligent and sound decisions. Building and maintaining representations of the input data, or representation learning, is therefore an important and widely studied problem in artificial intelligence [7]. Representation learning is often employed in a variety of problem setups to yield an interpretable representation of the data and the output. [15] introduces Information Maximizing Generative Adversarial Networks (InfoGAN), an unsupervised representation learning algorithm that learns interpretable and disentangled representations from unlabeled image datasets. The algorithm is able to

identify distinct writing styles from the MNIST dataset and visual concepts such as hair styles from the CelebA face dataset. Similarly, [2] learns interpretable data representations from image datasets using both generative and discriminative models, and shows that the enhanced interpretability improves the generalizability of the models.

In the imitation learning domain, some effort has been made to go beyond surface-level mimicry and instead learn meaningful representations about the task environment. [35] implements a general functional gradient boosting approach to imitation learning in relational domains. Given a set of traces from the human teacher, the system learns a policy in the form of a set of relational regression trees that additively approximate the functional gradients. The algorithm is applied to four relational domains and is shown to perform better than comparable propositional models. In [28], the agent is given input-output image examples and learns the task goal without any explicit instructions. Concepts are derived as programs that will produce the correct output image when executed with the given input image. Using this architecture, most concepts are learned correctly with just a few examples and generalize well to markedly different images depicting the same concepts.

2.2 Grounding

While representation learning is most often discussed in the context of deep learning algorithms [15, 2], in which the intermediate representation does not necessarily have semantic significance, it can also be applied to domains with clear underlying structures to achieve grounding. In the context of our work, we define grounding as creating connections between abstract expressions and real-world referents such as object color, size, location, and arrangement. [51] provides a detailed survey of the grounding problem and introduces a general framework for evaluating the quality of a grounded representation.

Compositionality is a useful tool for building and evaluating a grounded representation when inputs exhibit some structure (i.e. geometric, spatial, and relational features). A number of works utilize this concept to learn compositional image rep-

representations and construct an image-to-concept mapping. [48] introduces a simple regularization technique that allows the learned representation to be decomposable into parts, using attribute annotations to disentangle the feature space of a network into subspaces corresponding to the attributes. The Tree Reconstruction Error (TRE) algorithm [4] is one evaluation method for assessing compositional structure in representation learning problems where the structure of the observations is understood.

Several works focus on grounding concepts through human-robot interaction. Lemaignan et al. envisions a natural dialogue between a human and a robot helper [29]. When the human asks for help in vague terms such as “Give me that,” the robot should be able to understand both the verbal and non-verbal context: understand the semantics of the sentence, correctly interpret the request in the spatial context of the speaker, and transform this into an appropriate response. To address the language processing component, they implement a semantic parser that extracts the grammatical structure from the input sentence and translates it into an RDF statement that can be used to build upon an existing knowledge base. This work demonstrates that extracting and representing symbolic knowledge from the real world can improve the robot’s ability to perform high-level tasks. However, it ultimately addresses human-robot interactions of a more social nature, focusing on identifying and handling different types of utterances in natural dialogue such as declarative statements and questions. [11] presents a more task-oriented scenario of a robot that learns to perform everyday tasks from human demonstration. The robot has no prior knowledge of high-level concepts, and learns them solely from observing demonstrations by a human teacher. For instance, with enough demonstrations of moving all markers to the left side of the table, the robot is able to ground visual and spatial concepts such as “markers” and “left side of the table” and can perform the same task with a different configuration. While this work shows the generalizability of grounded concepts in the task goal learning problem, it does not utilize any linguistic input to guide the grounding process, which is the point of interest in this thesis.

2.3 Task-Oriented Language Grounding

Language grounding is a well-studied problem across many fields that involve task learning. In computer vision, many works explore the use of natural language instructions to automate image annotation [53], generation [52, 40], and manipulation [10, 45, 30]. Zhang et al. learn visual representations of medical images from the paired annotations [53]. Since medical image understanding often requires representations of visual features that are much more fine-grained than those required for identifying objects in natural images, an emphasis is placed on the image encoding mechanism and making efficient use of the small medical image dataset. [52] proposes an attentional generative adversarial network (AttnGAN) for generating a detailed image based on the text description. The description is encoded using a bi-directional LSTM unit to generate a global sentence vector, which is used to generate a low-resolution image in the first stage. In the following stages, each image sub-region is further refined using a word-context vector produced by an attention layer. This architecture effectively yields a higher-resolution image with more details at each stage. [40] proposes a similar architecture for the image generation, but leaves the specific text encoder up to choice. [30] presents a model for manipulating specific visual attributes of a given image based on a text description of the desired attributes. The text is encoded using a pre-trained RNN text encoder designed in [34], and the text-image affine combination module combines the encoded text and image features to select text-relevant regions that need to be modified. [45] utilizes LSTM to encode the text instruction that specifies how the given image should be manipulated. Similarly to the language encoding mechanism proposed in this paper, [45] utilizes Global Vectors for Word Representation (GloVe) [37] to map the input image edit request to vectors. While impressive results are presented using the text-image inputs, these works focus primarily on GAN-driven image generation and manipulation rather than the text embedding process itself.

More relevant to the human-robot interactive component of our scenario is the language-guided navigation task, in which a mobile agent is provided with natural

language directions and a view of the task scene. The agent learns to transform the instruction into a navigation plan that can be executed to reach the desired destination. Considerable work has been done on this area [12, 24, 47, 13, 17, 42], with varying approaches for formulating the task. [12] develops a gated attention mechanism for mapping the language to the visual attributes. The proposed model consists of a convolutional network to process the input image, a Gated Recurrent Unit (GRU) network to process the instruction, and a multi-modal fusion unit that combines the two representations. The fusion unit concatenates the embedded language and image features and applies a fully connected linear layer with a sigmoid activation to produce the attention vector. This output vector is used by the policy learning module to estimate the policy function. In [24], the task is modeled as a Markov Decision Process (MDP), where each action yields a corresponding reward to the agent. The text instruction is passed through an LSTM network to obtain a continuous vector representation, and the output vector is reshaped into a kernel to perform a convolution operation on the 2D image embedding and produce a language-conditioned state representation. [47] dynamically instantiates a probabilistic graphical model for the natural language command. The input command is decomposed into Spatial Description Clauses [25], and a “Generalized Grounding Graph” is constructed according to the command’s hierarchical and compositional linguistic structure inferred from the SDCs. [17] and [42] take a step further towards generalizability and include a higher-level master policy that proposes subgoals to be executed by specialized sub-policies. [13] utilizes both the natural language input and repeated observations of human demonstrations to accomplish the navigation task. The agent first infers a navigation plan for the instruction based on the observed actions. Using this as supervision, it then learns a semantic parser that can map novel instructions into executable navigation plans. Because these works aim to translate navigational instructions to the robot, they mainly focus on grounding navigational verbs like “go” and “follow” and inferring the relative spatial attributes of the objects in the scene such as “the westernmost rock.”

A number of approaches have been proposed for employing natural language to

guide task learning in a table-top pick-and-place setup. [20] proposes a probabilistic model to ground abstract concepts in natural language instructions. They formulate the grounding problem as estimating the likely set of groundings for an input instruction. The model incorporates notions of cardinality (“one”, “two”) and ordinality (“first”, “second”), as well as spatial references (“nearest”, “farthest”). Given an instruction such as “pick up the second block from the row of blocks,” the model identifies the keywords and reduces the search space of abstract concepts by pruning away the unlikely portions. The architecture of [1] is motivated by the fact that humans exhibit selective attention when performing tasks: when observing a scene with a particular task in mind, the features of the scene that are relevant to the task are given more attention, while others are de-emphasized or even ignored. Thus, their proposed vision system learns to pay attention to only the relevant regions of each frame regarding the task at hand. [33] presents a model that takes into account the variations in natural language, and ambiguities in grounding them to robotic instructions with appropriate environment context and task constraints. The environment and task context is encoded into an energy function over a conditional random field, and the language input is reduced to a more formal structure based on clausal decomposition. [21] more explicitly limits the form of the input instruction as $\langle \text{target relations referent} \rangle$, and extracts the relevant object information from the corresponding parts of the input. This approach to achieving the language embedding is similar to that of the semantic command parser proposed in this paper. Taking advantage of the linguistic structure inherent in natural language can facilitate the decomposition of the input instruction and generation of sub-tasks.

Chapter 3

Approach

In this chapter, we describe the various components our framework that enable the robot to derive elemental sub-tasks from an abstract natural language command, and execute them in sequence to accomplish the final task goal. A schematic of the framework is shown in Figure 3-2.

3.1 Modifications to Prior Work

Our framework is presented as an extension to [46], which implements a language-conditioned imitation learning model that controls a robotic arm to perform pick-and-pour tasks in a table-top setup. See Figure 3-1 for a detailed illustration of their architecture. The model receives a top-down image of the workspace of the robot, as well as a natural language command involving either a “pick” or “pour” action. The input features are pre-processed and passed into the model’s attention network, and a mapping between the input features and the task embedding is learned in the trainable weights of the attention network. The task embedding is then converted to dynamic movement primitive (DMP) parameters [22] for the robot control via a second set of trained weights, and the trajectory is executed until the task is complete.

The raw image of the task scene and natural language instructions, paired with the task’s corresponding DMP trajectories, are provided to the model at training time. The training process generates the mappings from the image and instruction inputs to

the task embedding, and from the task embedding to the motion primitive parameter output. These correlations are encoded in the learned weights in the semantic and control models.

While [46] successfully translates high-level natural-language instructions to low-level actions, it is only capable of performing a specific “pick” or “pour” operation for each instruction. Commands it can handle reliably are limited to the format “pick up the X cup” and “pour it into the Y dish.” Compound commands that combine these two operations, such as “pick up the blue cup and pour it into the yellow dish,” will fail, even though these two types of commands are semantically identical and should result in the same behavior from the robot. It follows that the model also cannot handle commands involving multiple objects in the scene, such as “pour all cups into the dish.” Our contribution expands the model’s scope to be able to handle compound and multi-object commands.

The original model implements a control policy that generates a *single* goal-oriented motion primitive that achieves an elemental action (“pick” or “pour”). In our approach, the policy maps the input $\mathcal{X} = [I, v]$ to an automatically generated *state machine* of motion primitives that sequences individual elemental actions to achieve the final goal. Hence, our architecture produces an end-to-end language-conditioned high-level control policy of the following form:

1. $\pi : (\mathcal{X} = [I, v]) \rightarrow (\mathcal{Y} = R)$ for the “open-loop” case, where $R = [r_0, \dots, r_t]$ is the full trajectory of the robot joint configurations that achieves the high-level goal when executed in sequence.
2. $\pi : (\mathcal{X} = [I, v, r_t]) \rightarrow (\mathcal{Y} = r_{t+1})$ for the “closed-loop” case, where the parameters of the motion primitive are recomputed at each time step to account for execution noise.

We assume the latter approach in the subsequent sections.

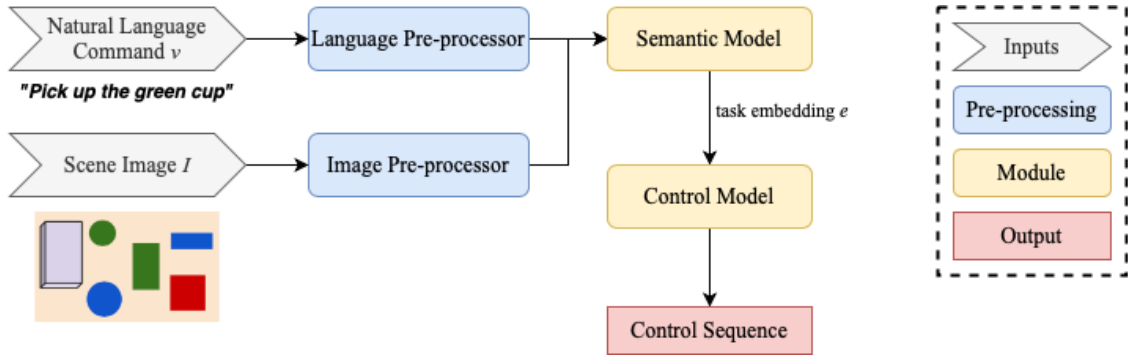


Figure 3-1: Model architecture implemented in [46] for the imitation learning of *single* target-oriented robot manipulation tasks from goal-specific instructions.

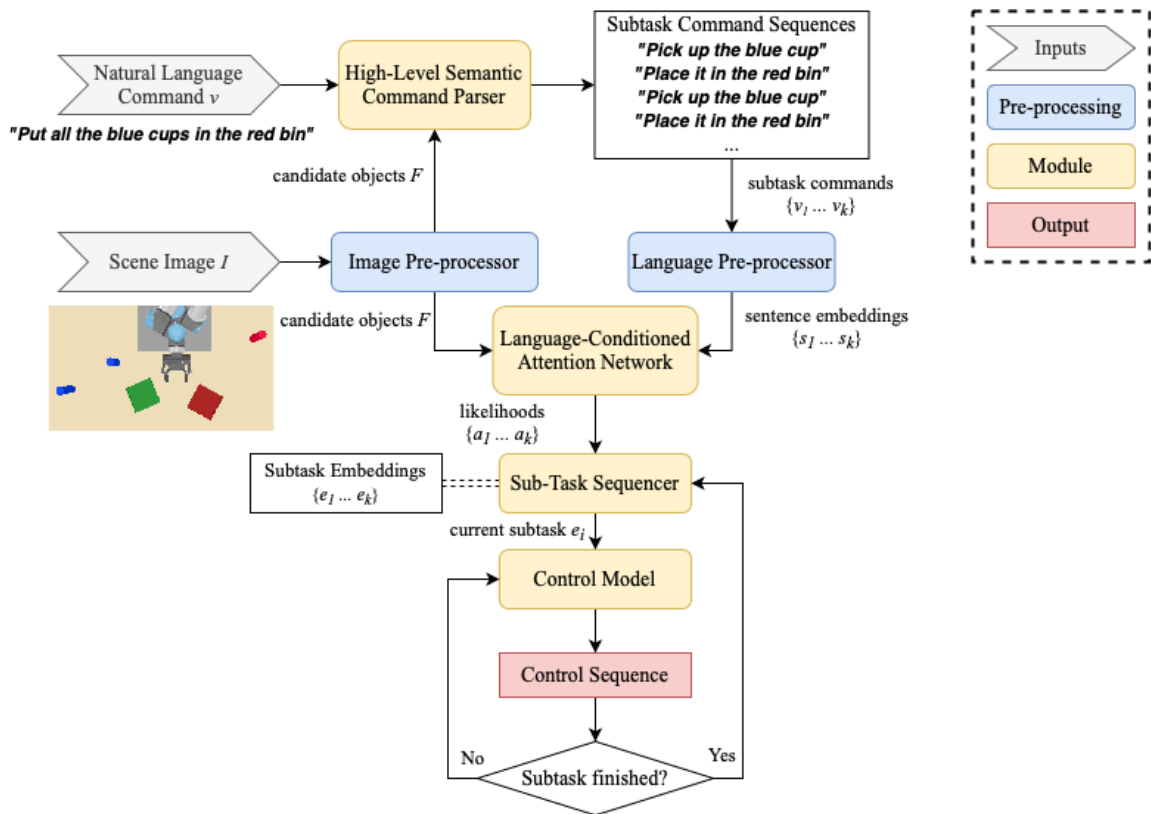


Figure 3-2: Model architecture proposed in this work for the imitation learning of *sequential* robot manipulation tasks from high-level abstract instructions.

3.2 Architecture Overview

The proposed language-conditioned imitation learning framework, illustrated in detail in Figure 3-2, is composed of the image and language pre-processing units as well as four major modules: the high-level semantic command parser, the language-conditioned attention network, the sub-task sequencer, and the low-level control model. First, the raw scene image I is encoded by the image pre-processor to produce a list of candidate objects F . The semantic command parser receives the high-level natural language instruction v and the parsed task scene information F to generate a list of atomic sub-task commands $\{v_1 \dots v_k\}$ described in natural language. Each sub-task command is encoded by the language pre-processor to produce the corresponding sentence embeddings $\{s_1 \dots s_k\}$. The language-conditioned attention network combines the two encoded input modes F, s to produce an intermediate representation for each sub-task: $\{a_1 \dots a_k\}$. The sub-task sequencer determines the order of the sub-tasks to be performed and the target object for each sub-task, and instantiates a state machine based on the resulting list of sub-task embeddings $\{e_1 \dots e_k\}$. Finally, the low-level control model is called to generate the control parameters for each of sub-task e_i .

3.2.1 Image Pre-processing

The image pre-processing stage is analogous to the first therblig motion, “Search.” The object detection module identifies all candidate objects in the scene, which are later used perform the subsequent “Find” operation. The module receives a top-down image of the task scene $I \in \mathbb{R}^{569 \times 320 \times 3}$. A pre-trained object detection network, Faster R-CNN [41], is applied to I to generate a set of candidate objects in the scene:

$$F = \{[f_c^o, f_c^b]\}_{c=1:C} \quad (3.1)$$

where each of the C detected objects is represented by a feature vector, comprised of the detected object class f_c^o and the bounding box of the object $f_c^b \in \mathbb{R}^4$. The FRCNN

model used for the image processing was pre-trained from ResNet-101 on the COCO dataset. Figure 3-3 shows a visualization of the candidate objects identified by the detection module in a scene with three bins and two cups, with their respective internal object indices and bounding boxes.

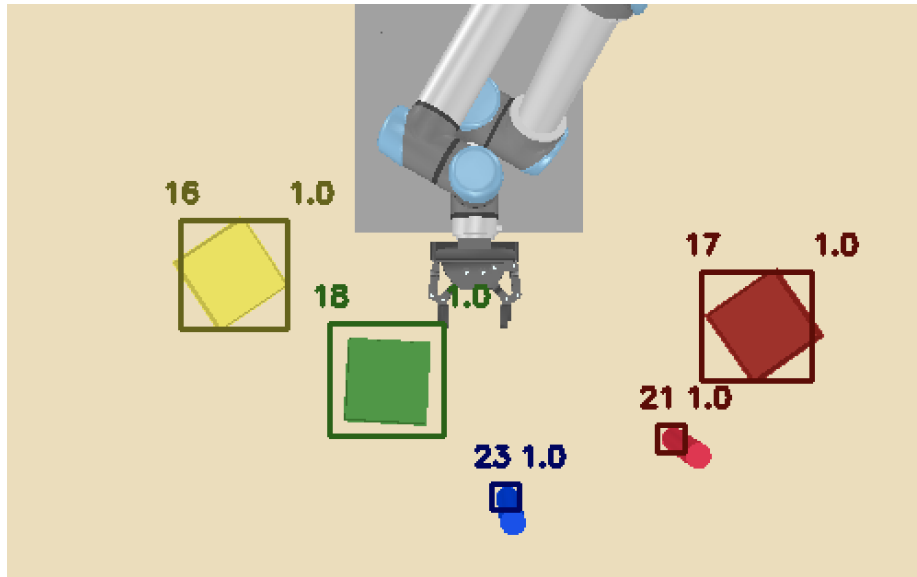


Figure 3-3: Visualization of the result of candidate object detection. The integer on the top left of each region indicates the detection model’s internal object index, and the score on the top right shows the detection confidence. The bounding boxes are color-coded for convenience.

3.2.2 High-Level Semantic Command Parser

The semantic command parser serves two functions: check that the input command is well-formulated (i.e. follows the expected grammar and syntax rules), and decompose the command into a series of instructions for atomic sub-tasks. The initial command is split into individual words and tagged using the part-of-speech (POS) tagging convention. POS tagging identifies each word as a grammatical category such as noun or verb. Table 3.1 shows a comprehensive list of the tags used in this work, as well as their definitions and examples.

POS tagging is a nontrivial task, with extensive work being done in the NLP domain to develop accurate and efficient techniques for identifying parts of speech

[27, 32, 49]. However, as this is not the main focus of our work and our natural language input is relatively simple and well-defined, we use the python nltk library [9] to tag the instruction and assume that POS tagging is done correctly.

POS Tag	Definition	Example
VB	verb (base form)	<i>place</i>
NN	singular noun	<i>cup</i>
NNS	plural noun	<i>cups</i>
DT	determiner	<i>the</i>
PDT	pre-determiner	<i>all</i> in “all the cups”
JJ	adjective	<i>red</i>
IN	preposition	<i>in</i>
RP	particle	<i>up</i> in “pick up”
PRP	personal pronoun	<i>it</i>
CC	coordinating conjunction	<i>and</i>
CD	cardinal digit	<i>two</i>

Table 3.1: A complete list of part-of-speech tags used in this work.

The standardized POS tags allow for a pattern-matching approach to segment the input sentence. Using these tags, we define three patterns of sentence segments at the highest level: command phrase **V**, objects **O**, and location **L**. These are conceptually analogous to the linguistic units: verb phrase, noun phrase, and prepositional phrase, respectively. We use the regex syntax to describe these patterns below.

1. Command phrase **V**: VB RP?
2. Objects **O**: (DT JJ? NN | PDT DT? JJ? NNS | CD JJ? (NN | NNS) | PRP)
3. Location **L**: IN DT JJ? NN

A command phrase **V** simply consists of a verb and optionally a particle associated with the verb. The objects **O** can be specified in a few different patterns, from the simplest form *it* to a phrase describing the number of objects and a reference to the type of object such as *two green cups* or *all the red cups*. The location **L** specifies the task goal location for performing the “place” operation.

We define a “valid” manipulation command as one that matches the following syntax pattern:

$$\mathbf{V O}(((\mathbf{C C O}) * | (\mathbf{C C V O})) ? \mathbf{L}) ? \quad (3.2)$$

For instance, the simplest command structure **V O** contains only the command phrase and the target object. More complex commands may contain multiple command phrases or target objects. Figures 3-4 through 3-7 show sample commands following the four major patterns of increasing complexity and their semantic trees. The rule is specific for our table-top pick-and-place scenario, but the parser can use any set of pre-defined grammar rules and thus can be adapted to any environment that uses semi-structured natural language input.

Once the input command is determined to be valid, the parser generates the natural language sub-task instructions. Based on the action in the command phrase and the available objects in the task scene, a series of “pick” or “place” commands is generated that accomplishes the initial goal when executed in sequence. Figure 3-8 shows two examples of the task scene and instruction input pairs and the corresponding sub-task instructions generated by the parser.

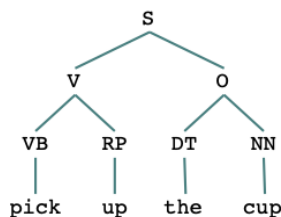


Figure 3-4: The semantic tree for a command with the structure **V O**.

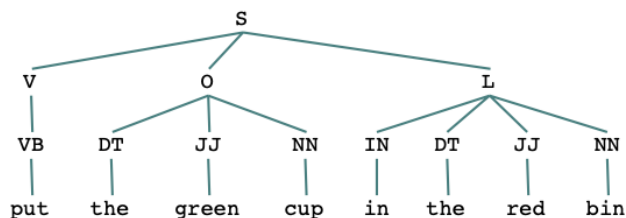


Figure 3-5: The semantic tree for a command with the structure **V O L**.

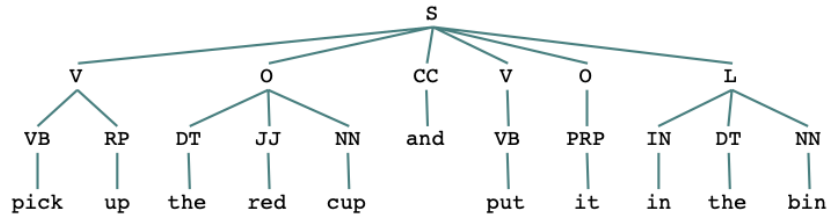


Figure 3-6: The semantic tree for a command with the structure **V O CC V O L**.

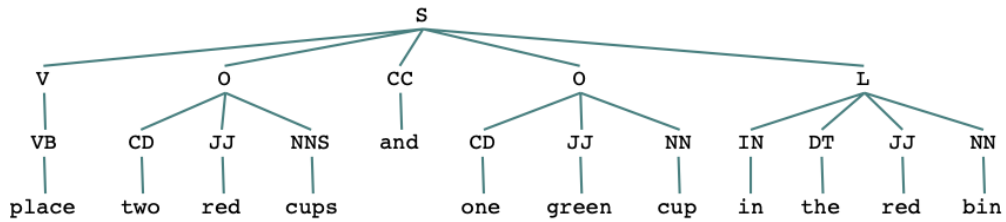


Figure 3-7: The semantic tree for a command with the structure **V O CC O L**.

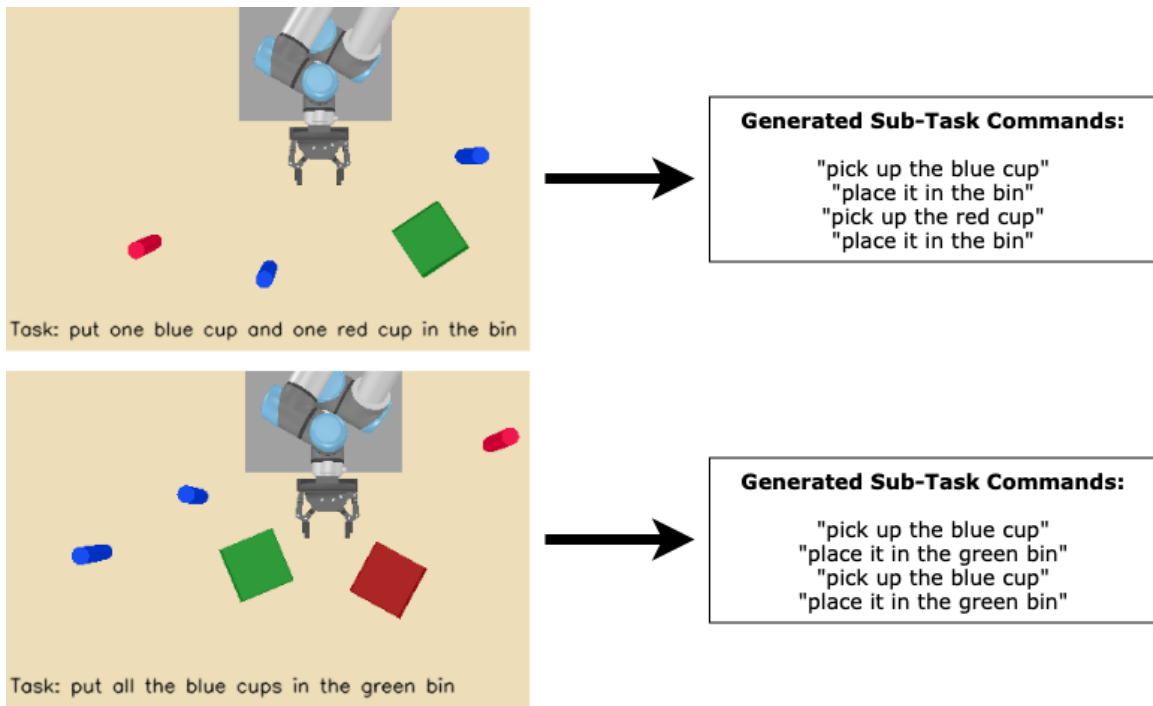


Figure 3-8: Atomic sub-task commands generated by the semantic parser based on the detected task objects and the high-level instruction.

3.2.3 Language Pre-Processing

The language pre-processing stage captures the information represented in the natural language command. The sub-task instructions generated by the semantic command parser are tokenized using the GloVe vector representation for words [37]. Each word in the sub-task instruction v_i is mapped to a row index of $G \in \mathbb{R}^{30000 \times 50}$, consisting of 30,000 most frequently used English words represented as GloVe word embeddings. The tokenized sentence is then converted into a fixed-size matrix $V \in \mathbb{R}^{15 \times 50}$ that encodes up to 15 words, where each row is a 50-dimensional word embedding. Finally, a GRU layer is applied to V to produce a sentence embedding, $s_i \in \mathbb{R}^{32}$.

3.2.4 Language-Conditioned Attention Network

The attention network combines the sub-task sentence embedding s_i and the image embedding F to identify the target object referred to by the command.

$$a_i = w_a^T f_a([f_c, s_i]) = \{[a_i^c]\}_{c=1:C} \quad (3.3)$$

For each of the C candidate objects in F , a likelihood a_i is calculated by concatenating the sentence embedding s_i with the object’s feature vector f_c and applying the attention network $f_a : \mathbb{R}^{37} \rightarrow \mathbb{R}^{64}$. The result is multiplied with a trainable weight $w_a \in \mathbb{R}^{64}$ to be converted into a scalar. The attention network f_a is defined as follows:

$$f_a(x) = \tanh(Wx + b) \odot \sigma(W'x + b') \quad (3.4)$$

where W, W' and b, b' are trainable weights and biases, respectively.

3.2.5 Sub-Task Sequencer

The sub-task sequencer contains two components: the task object selector $\Pi(\cdot)$ and the high-level policy generator. The task object selector first applies a sigmoid function to the likelihoods produced by the attention network: $a = \sigma([a_i^1 \dots a_i^C])$. The sigmoid function serves to polarize the likelihoods so that the object with the highest

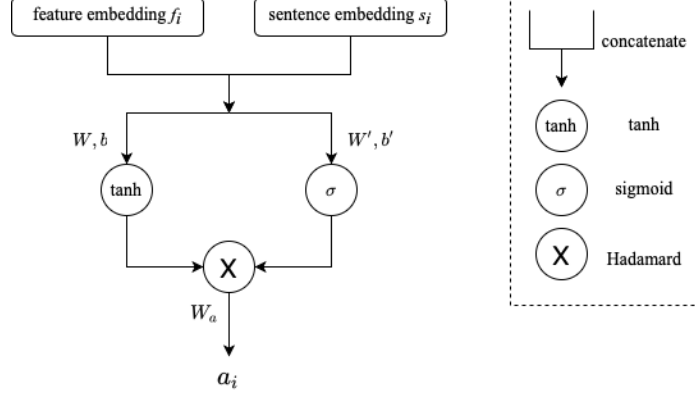


Figure 3-9: A detailed diagram of the language-conditioned attention network.

likelihood is accentuated. In the case where there are multiple objects in the task scene that meets the object description, all such objects will have a likelihood close to 1. The selector identifies all the objects that match the description given by the sub-task command and chooses a target object $\Pi(a)$. Thus, the selector effectively performs the second therbilig operation, “Find”. As a simple proof-of-concept implementation, we use an algorithm that randomly selects the task object. However, more complex task object selectors can be used here in place of the random selector. Such possible extensions will be discussed later in this section.

A sub-task embedding $e_i \in \mathbb{R}^{32}$ is computed by multiplying $\Pi(a)$ with the image features F , concatenating it with the sentence embedding s_i , and applying ReLU, with trainable weight W and bias b .

$$e_i = \text{ReLU}(W[F \cdot \Pi(a), s_i] + b) \quad (3.5)$$

The high-level policy generator Φ then determines the order of the sub-tasks to be performed in sequence. A finite state machine (FSM) is constructed with the ordered list of sub-task embeddings $\{e_1 \dots e_k\}$. Each state in the FSM represents the sub-task embedding to be passed to the low-level control model. At each time step, after the control sequence for the current sub-task embedding is executed, the phase value returned by the low-level control model is used to determine if the sub-task

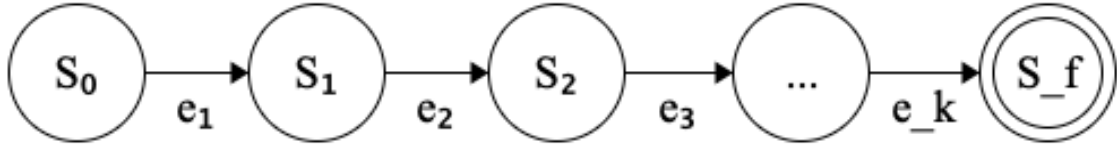


Figure 3-10: A finite state machine generated for the k sub-tasks $\{e_1 \dots e_k\}$ in Φ .

is complete. If the finish condition is met, Φ is called to move onto the next sub-task embedding to execute. This process repeats until all the sub-tasks have been completed.

3.2.6 Control Model

We directly adopt the control model used in [46], which was in turn inspired by the approach in [22]. The control model maps the sub-task embedding e_i and the current robot state r_t to control signals for the next time step, and produces a trajectory of dynamic motor primitive (DMP) parameters for the robot’s seven joints (see Figure 3-11). The trajectory can be executed in an open-loop fashion, but to better account for the non-deterministic nature of control tasks such as physical perturbations and execution noise, the trajectory is recalculated at each time step.

To keep track of the robot’s current and previous movements, the robot state r_t at each time step is encoded by a GRU cell that is initialized with the robot’s start configuration r_0 . This produces the latent robot state $h_r \in \mathbb{R}^7$. The motor primitive is learned in the trainable weights $w \in \mathbb{R}^{B \times 7}$, where B is the number of kernels for each DOF of the robot. The estimated current progress to achieving the goal is reflected in the phase variable $0 \leq \phi \leq 1$, where 0 indicates that the control sequence has not yet started and 1 indicates that the execution is complete. The phase progression for each time step is specified by Δ_ϕ . Based on the sub-task embedding e_i and the robot state h_r , the model generates a full set of motor primitive parameters for the current time step:

$$(w_t, \phi_t, \Delta_\phi) = (f_w([h_t, e_i]), f_\phi([h_t, e_i]), f_\Delta(e_i)) \quad (3.6)$$

where $f_w : \mathbb{R}^{39} \rightarrow \mathbb{R}^{B \times 7}$, $f_\phi : \mathbb{R}^{39} \rightarrow \mathbb{R}^1$, and $f_\Delta : \mathbb{R}^{32} \rightarrow \mathbb{R}^1$ are multilayer per-

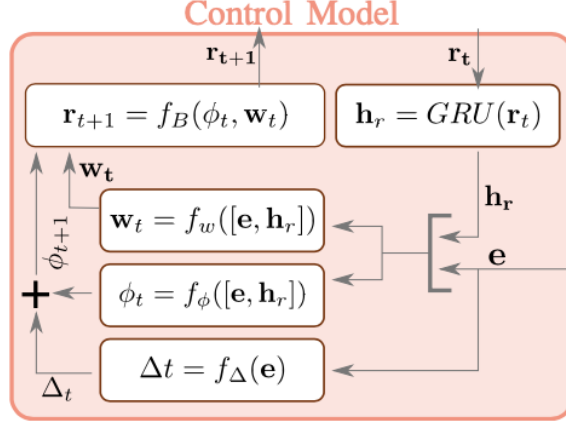


Figure 3-11: Details of the control model, which generates the robot control signals. This figure was taken from [46].

ceptrons. The motor primitive $(w_t, \phi_t, \Delta_\phi)$ is evaluated at each phase $\phi_t + \Delta_\phi$ to generate the next control signals r_{t+1} to be executed. We define a sparse linear map $H_{\phi_t} \in \mathbb{R}^{7 \times (B \times 7)}$ that contains the basis function vectors for each DOF. The control signal generator f_B performs a simple multiplication operation on $H_{\phi_t + \Delta_\phi}$ to generate the next control signal:

$$r_{t+1} = f_B(\phi_t + \Delta_\phi, w_t) = H_{\phi_t + \Delta_\phi} w_t^T \quad (3.7)$$

The output of the control model accomplishes the final series of therblig motions: “Transport Empty + Grasp” for the pick operation, and “Transport Loaded + Position + Release Load” for the place operation.

Chapter 4

Evaluation

In this chapter, we evaluate the performance of our language-conditioned sequential task learning framework in an end-to-end fashion. Two evaluation types were performed: sorting and kitting. The sorting task involves arranging all objects with a common feature into a designated location on the table. The kitting task compiles a collection of objects as specified in the task instruction. Section 4.1 describes the training and data collection processes and the evaluation metrics. Section 4.2 presents the overall test results and performance details of the different components of the pipeline. Section 4.3 presents a discussion of the results and examines potential improvements to the framework.

4.1 Methods

4.1.1 Training

The proposed framework was implemented using the pre-trained weights provided by [46] for mapping the embedded language instruction to the attention and control features. The control model was trained to perform pick-and-pour tasks, which we have modified to perform pick-and-place operations instead. As such, no re-training of the model was necessary for the evaluation.

The original model was trained using five auxiliary losses to optimize the learning

process. The attention network was trained using $\mathcal{L}_a = -\sum_i^c x_i \log(y_i)$, defined as the cross-entropy loss for a multi-class classification problem over c classes. The training label was a one-hot vector created in the image pre-processing stage and indicated the object being referred to by the task description. The control model was trained using the combined mean-squared-error losses of four parameters: phase estimation $\mathcal{L}_\phi = \text{MSE}(\phi_t, \phi_t^*)$, phase progression $\mathcal{L}_\Delta = \text{MSE}(\Delta_\phi, \Delta_\phi^*)$, weights $\mathcal{L}_w = \text{MSE}(W_t, W_{t+1})$, and trajectory $\mathcal{L}_t = \text{MSE}(R, R^*)$. The expected phase estimation and progression ϕ_t^*, Δ_ϕ^* were inferred from the number of steps in the given demonstration. For the trajectory loss, the generated trajectory $R = [r_{\phi=0} \dots r_{\phi=1}]$ was compared against the trajectory of the demonstration R^* . A weighted sum of the five losses was used as the overall loss: $\mathcal{L} = \alpha_a \mathcal{L}_a + \alpha_\phi \mathcal{L}_\phi + \alpha_\Delta \mathcal{L}_\Delta + \alpha_w \mathcal{L}_w + \alpha_t \mathcal{L}_t$. Values $\alpha_a = 1, \alpha_\phi = 1, \alpha_\Delta = 14, \alpha_w = 50, \alpha_t = 5$ were empirically chosen as the hyper-parameters for \mathcal{L} , and the model training was supervised by minimizing \mathcal{L} with an Adam optimizer using a learning rate of 0.0001. More details on the training process can be found in [46].

4.1.2 Test Data Collection

We tested our framework in a simulated table-top environment with a fixed 7-DOF robot arm. To emulate realistic pick-and-place task scenarios, we devised the task types for the evaluation based on two factory use cases: sorting and kitting. 600 distinct task scenes were generated for the evaluation: 300 for the sorting task and 300 for kitting. The attention network was not trained on these scenes, so the performance of the attention network in the testing phase is a direct reflection of the network’s generalizability. Each scene contained at most six objects from the available objects introduced in Figure 1-4, with at least one and up to four graspable *cups* with two color variations and at least one and up to three *bins* of distinct colors for the cups to be placed in. The positions and orientations of the objects were randomized, with a positional constraint to prevent any collisions.

High-level task commands for the sorting and kitting scenarios were automatically generated using the sentence templates: “put all the X cups in the Z bin” for the

sorting task and “put n_X X cups (and n_Y Y cups) in the Z bin” for the kitting task, where X, Y, Z are optional color identifiers for the task objects and n_X, n_Y are cardinal values specifying the number of task objects. The task command generation was guided by the available objects in the scene and the valid operations that can be performed on them. For instance, the kitting instruction “put two blue cups in the bin” cannot be executed correctly if there is only one blue cup in the scene, as is the sorting instruction “put all the red cups in the bin” in a scene without any red cups. The final instruction for the test scene was randomly drawn from all the valid forms of commands generated in this manner.

Figures 4-1 and 4-2 showcase some examples of the randomly generated task scenes and commands for the sorting and kitting task types. Figure 4-3 catalogs sample task execution sequences performed by the robot for the two task types.

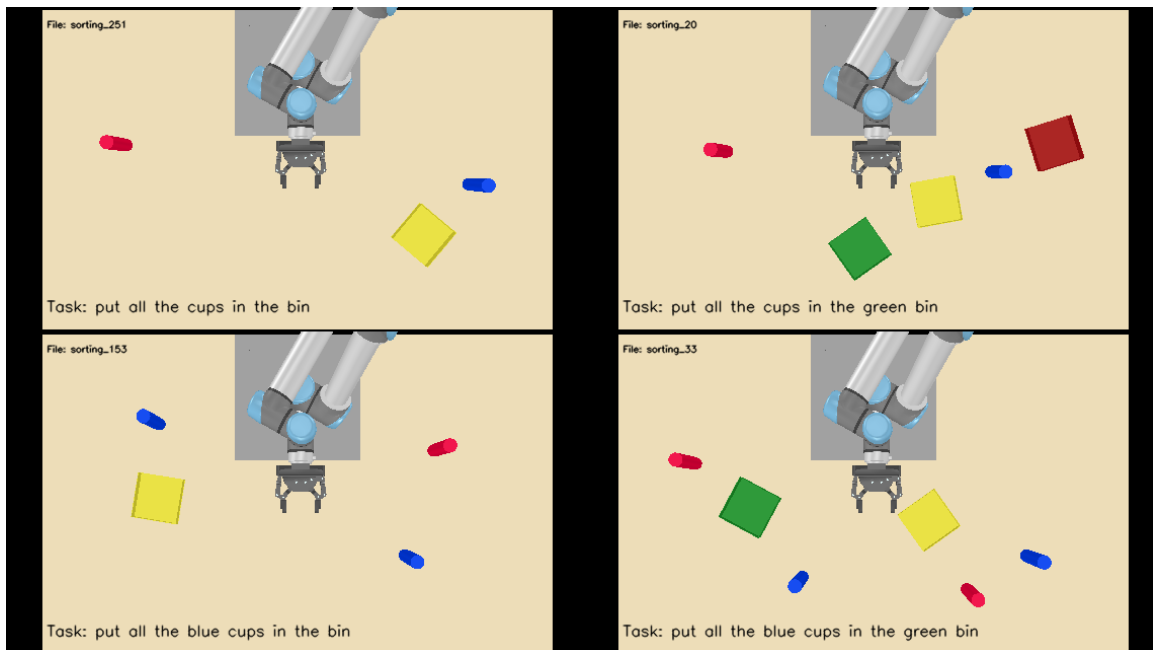


Figure 4-1: Examples of four randomly generated sorting task scenes and their corresponding task commands, roughly in the order of increasing task complexity.

4.1.3 Performance Metrics

The overall task completion rates were measured in each test scene, as well as more fine-grained metrics for evaluating each major module of the proposed framework. In

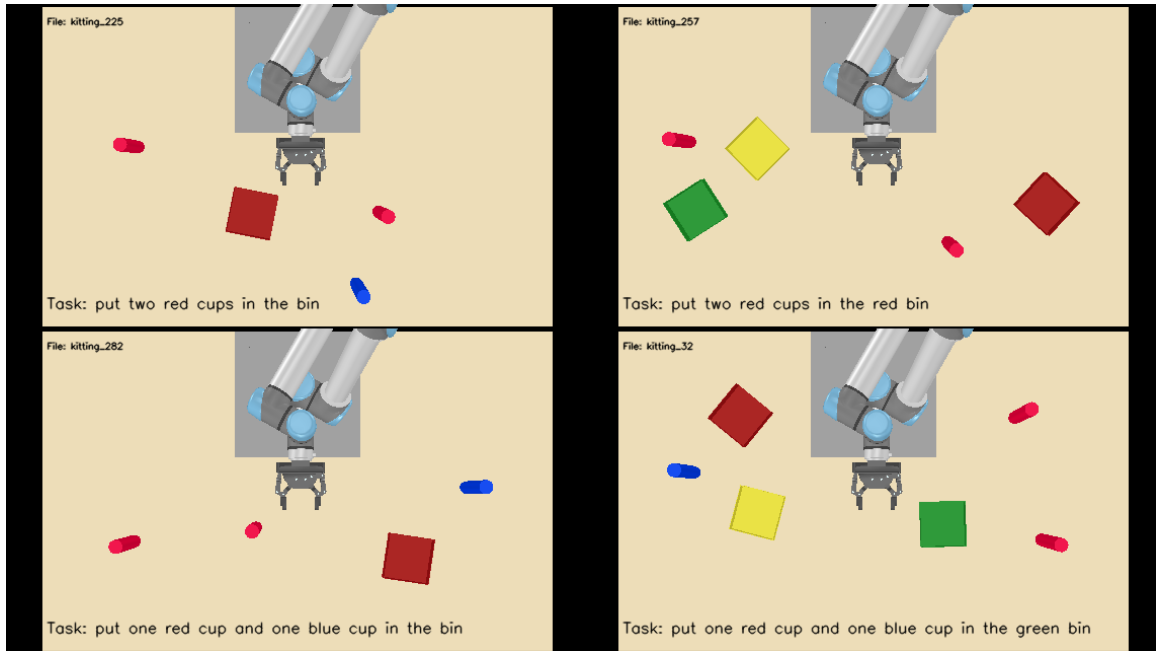


Figure 4-2: Examples of four randomly generated kitting task scenes and their corresponding task commands, roughly in the order of increasing task complexity.

particular, we measured the performances of the object detection module, attention network, and low-level control for the atomic sub-tasks.

We used the simulation input parameters used in generating the task scene to define ground truths for evaluating the object detection and attention network modules. This was represented as a zero-padded and sorted 1×6 array containing the internal object IDs. In a scene with two red cups (ID: 21) and a yellow bin (ID: 16), for instance, the detection ground truth for the objects in the scene is: $[0, 0, 0, 16, 21, 21]$. The labels for evaluating the attention network were derived from the current scene image at the beginning of each sub-task and filtered to include only the objects identified by the keyword in the sub-task command. For a sub-task instruction “pick up the red cup” in the example scene, the attention ground truth would be $[21, 21]$ for the two red cups in the scene. The following command, “place it in the yellow dish”, would indicate only the yellow dish in the scene: $[16]$. These ground truth labels were compared with the masked output of the attention network. A filtering threshold of 95% was used to determine whether the attention score was high enough relative to the entire output array of the attention network.

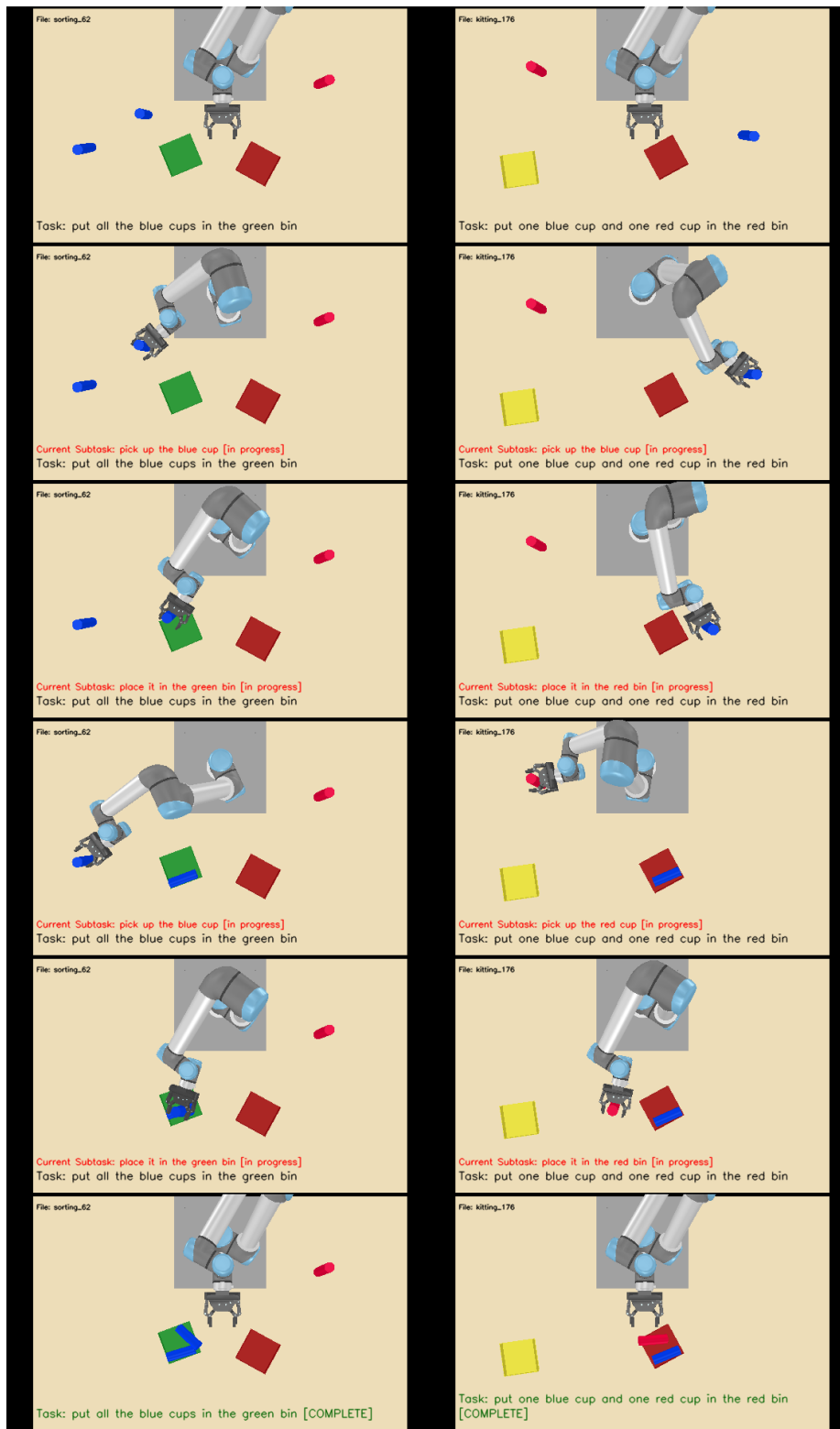


Figure 4-3: Sample task execution sequences for sorting (left) and kitting (right) tasks, with the original task command and the current sub-task command being performed.

The “pick” operation was defined by the robot’s gripper state, which ranges from 0 (open/released) to 1 (closed/gripped). If the correct object was inside the robot’s gripper when it was closed, this was counted as a successful “pick” operation. A “place” operation was considered successful if the gripper position was directly above the target bin at the time of release. This was decided in part to prevent the unpredictability of the simulation environment from affecting the measured task execution accuracy. The bins used for the training and testing in the simulation were quite small, which often led to cups being released in the correct location but falling out of the bin or rolling away after placement with the slightest perturbations in the environment. Evaluating the gripper location at the time of the “place” task execution was sufficient to assess the model’s accuracy for performing the task correctly.

4.2 Results

Table 4.1 summarizes the overall performance of the proposed approach and the success rates of each major component of the framework. A total of 547 tasks out of the 600 novel task scenes were correctly executed to completion (91.2%). The model achieved an object detection accuracy of 99.7% on 3,600 total task objects, attention accuracy of 81.1% on 1,990 target objects, and sub-task execution success rate of 96.4% for 1,712 individual “pick” and “place” sub-tasks. In the remainder of this section, we evaluate the performance of each of these modules in more detail.

4.2.1 Sub-Task Generation

The semantic parser produces the sub-task instructions based on the linguistic structure of the high-level task command and the objects in the task scene as detected by the image pre-processor. To generate the correct amount of sub-tasks for the sorting command “put all the X objects in the bin”, the parser identified all the possible targets from the objects in the scene and adjusted the number of actions accordingly. The command “put all the blue cups in the bin” in a task scene with two blue cups would generate two distinct “pick” tasks for the blue cup, while the same command

Model Performance Summary

	Sorting	Kitting	Overall
Sub-Task Generation	1.000	1.000	1.000
Object Detection	0.996	0.998	0.997
Attention	0.761	0.859	0.811
Pick	0.963	0.953	0.958
Place	0.972	0.970	0.971
Pick + Place	0.935	0.934	0.935
Final Task Completion	0.910	0.913	0.912

Table 4.1: Model performance summary of the proposed language-conditioned sequential task learning framework.

in a scene with one blue cup would result in only one “pick” task.

No explicit performance evaluation was performed on the sub-task generation, as the semantic parser strictly follows the grammar and task generation rules defined by the user. The grammar rules utilized in this framework is described in detail in Section 3.2.2.

4.2.2 Object Detection

The pre-trained FRCNN model was used for the object detection as discussed in 3.2.1. We briefly describe its performance here.

The detection accuracy on the input task scene image at the beginning of each simulation environment was used as the performance metric. As shown in the confusion matrix in Figure 4-4, all objects were detected and very few were mis-classified. Almost all classification failures resulted from the detection module identifying the same object twice, which was caused by the variation in the simulation’s lighting source and perspective distortions from the top-down camera. Figure 4-5 shows an example task scene in which a false positive in the detection module affects the subsequent generation and execution of the sub-tasks.

A more consequential type of failure occurred during the execution runtime. The top-down view of the task scene was often obstructed by the robot arm or the task objects during execution, which interfered with the detection of the occluded object

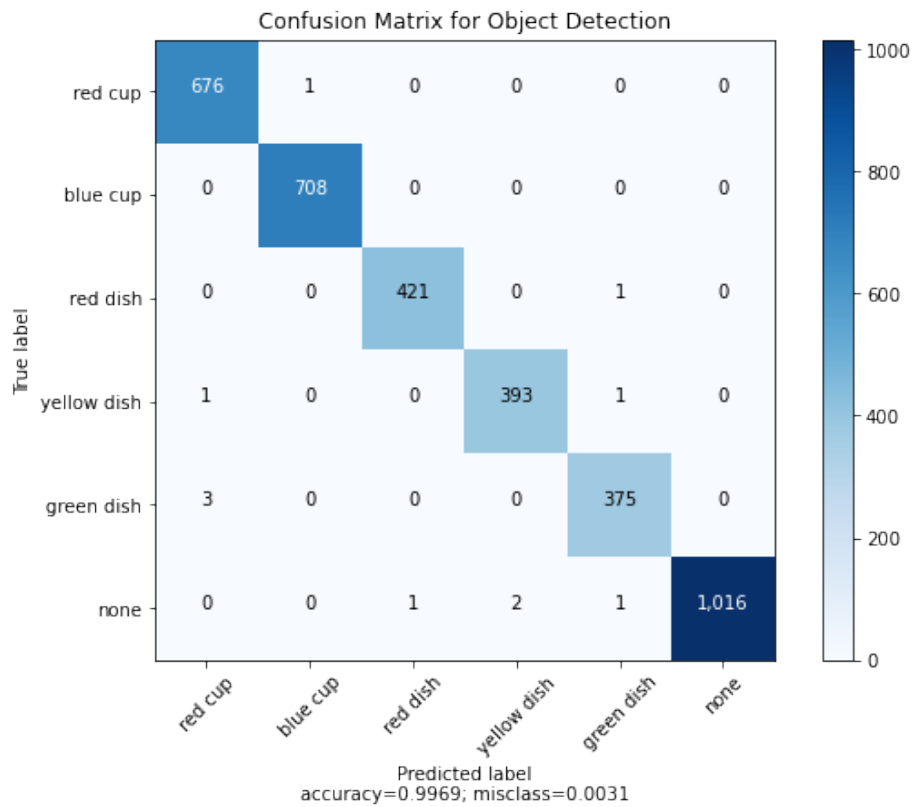


Figure 4-4: A confusion matrix for the object detection module. The module classified the 3,600 task objects with 99.7% accuracy, with very few classification failures.

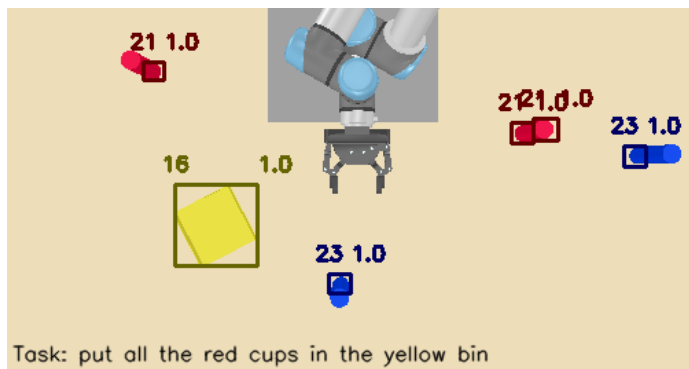


Figure 4-5: A sorting task with a false positive produced in the object detection stage. The red cup on the right is detected twice by the detection module, for a total of three perceived red cups in the scene. The parser generates three pick-place sub-task pairs based on this information, which ultimately leads to the robot attempting three pick tasks in total.

(Figure 4-6). The effect of this failure manifests in the result of the attention network, which is discussed in the next section. Because our model architecture was designed to build on previous components, a failure in an earlier module would cascade down the pipeline and ultimately affect the final task performance.

4.2.3 Attention

Some bin detection errors occurred in the detection module due to occlusion, which affected the performance of the attention network. In the initial implementation of our model, the embedding for the next sub-task would be generated at the end of the previous sub-task using the detected objects from the current scene. Occasionally, the position of the robot arm at the end of a “pick” task would obstruct the view of the target bin for the following “place” operation. In the same vein, objects placed in the bin during the previous sub-task could also interfere with the bin detection, affecting the execution accuracy of the subsequent “place” sub-tasks. This was more likely to occur when there were several objects in the target bin, or the object was of a different color than the bin. If the model could not identify the correct bin in the scene, it would default to a different, incorrect bin, or fail entirely if no other bin was in the scene (Figure 4-6). The first issue was addressed before evaluation by performing the object detection at the beginning of each “pick” task, when the robot arm is centered

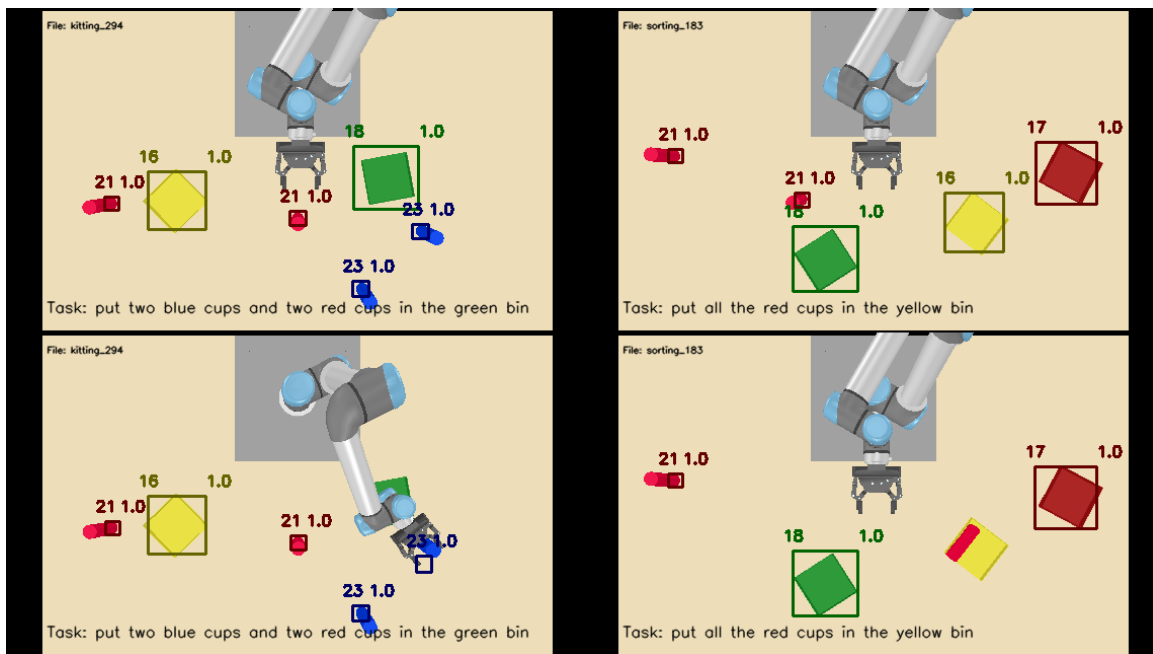


Figure 4-6: Two examples of bin detection failures caused by occlusion. In the first scene, the robot arm at the end of the previous sub-task obscures the target bin. In the second scene, the red cup placed during the previous sub-task interferes with the detection of the bin. Since no target bin is identified, the model performs the subsequent place operation on a different bin with the incorrect color.

and can be expected not to occlude any objects in the task scene. Addressing the second case is left for future work. The detection failures caused by the second issue largely explain the notably lower attention accuracy for the bins during the “place” operation compared to the “pick” operation, as shown in the side-by-side comparison of the two attention accuracies in Figure 4-8.

The attention network’s performance was also affected by the specificity of the task command. Generic commands such as “put the cup in the bin” in a task scene with several cups would result in multiple correct target objects. The attention network did produce high scores for all such objects, but the threshold of 0.95 used in the evaluation was quite stringent and sometimes resulted in objects with high attention scores getting filtered out. In particular, we note that in task scenes with generic “pick” commands and both red and blue cups present, the attention network always assigned a slightly higher score for the blue cup than the red. This anomaly accounts for the low attention rate on red cups for generic commands, which is displayed in Figure 4-9.

Another reason for the attention network’s lower accuracy is that the weights were trained on environments and commands that were less varied in the number of duplicate objects and more specific in instruction. A way to mitigate this is by augmenting the dataset of the previous model with our dataset and re-training the entire model. Alternatively, a more interesting solution would be to update an existing model to detect new commands and scenarios, as done in [14]. This is left for future work.

4.2.4 Task Execution

Table 4.2 summarizes the results of the execution performance on individual “pick” and “place” sub-tasks across all test scenes. A total of 856 pick-and-place sub-task pairs were executed, with an average of 2.85 sub-tasks per scene. 820 out of 856 pick tasks (95.8%) were correctly executed, while 831 out of 856 place tasks (97.1%) were correctly performed. The cumulative success rate (pick + place) describes the percentage of cases in which the cup was correctly picked up and then placed into the

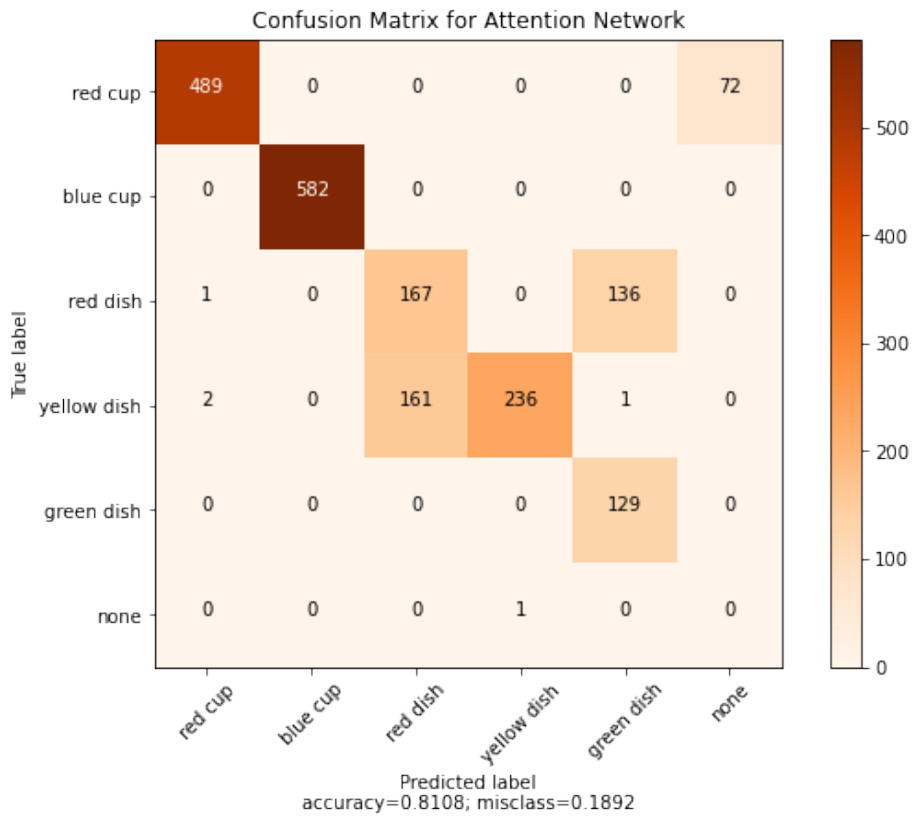


Figure 4-7: A confusion matrix for the attention network module. The module identified the 1,990 target objects with 81.1% accuracy.

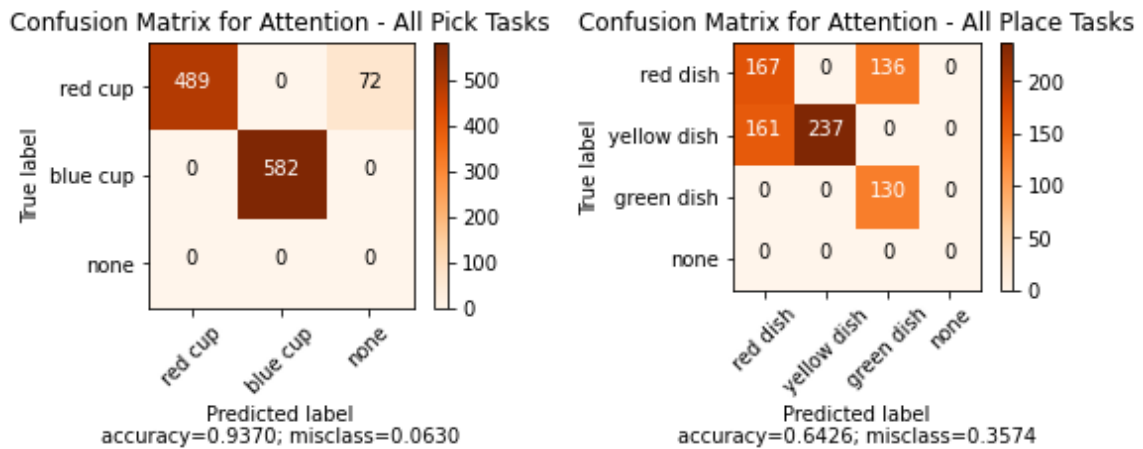


Figure 4-8: Attention network accuracy for 1,148 pick and 842 place target objects.

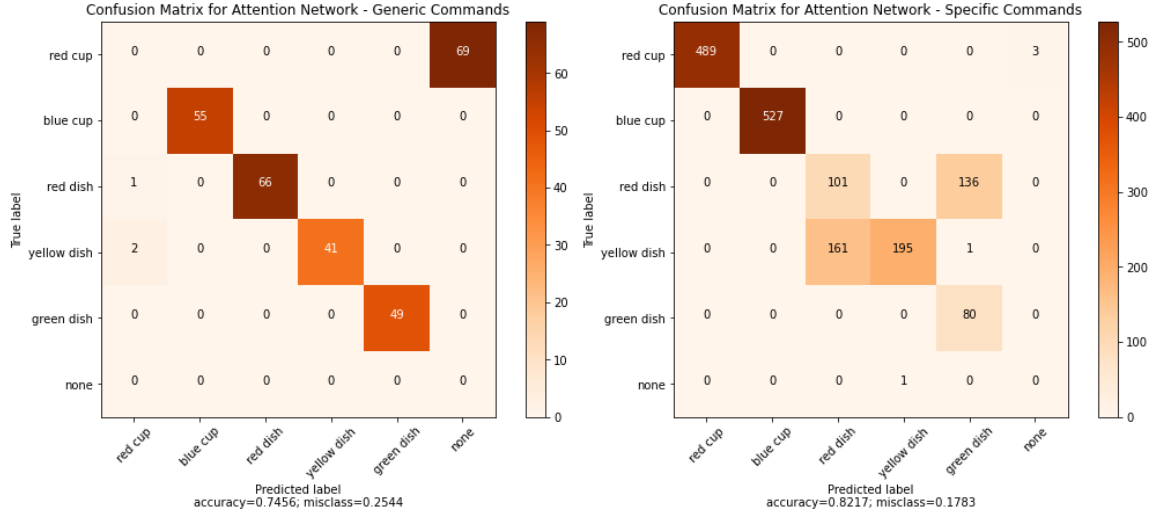


Figure 4-9: Attention network accuracy for 261 generic and 1,451 specific task commands.

correct bin. The pick-place sub-task pairs were executed successfully in 93.5% of all cases.

We further calculated the final task completion rates of the sorting and kitting scenarios (Table 4.3). The completion rate was measured based on whether the high-level task goal was successfully achieved, i.e. whether all sub-tasks were correctly performed. For a sorting task with the command “put all the X objects in the bin”, we calculate the intermediate pick completion rate n_{pick}/n_X , where n_{pick} indicates the number of objects correctly identified and picked up and n_X denotes the total number of pick operations to be performed. In the sorting scenario, n_X is the total number of objects in the scene with the feature X . The place completion rate is calculated as n_{place}/n_X , where n_{place} indicates the number of correct place operations performed on the target bin. Since the pick-place operations occur in pairs, the correct sub-task count is identical for both the pick and place completion rates. Similarly, for a kitting task with the command “put n_X X objects and n_Y Y objects in the bin”, the pick completion rate was calculated as $n_{pick}/(n_X + n_Y)$, based on the $n_X + n_Y$ pick tasks as specified by the command, and the place completion rate was $n_{place}/(n_X + n_Y)$. The final task completion rate calculates the ratio of the objects with correct execution of both sub-tasks, $n_{pick+place}$, over the total.

Sub-Task Execution Summary

	Sorting	Kitting	Overall
Pick	413/429 (0.963)	407/427 (0.953)	820/856 (0.958)
Place	417/429 (0.972)	414/427 (0.970)	831/856 (0.971)
Pick + Place	401/429 (0.935)	399/427 (0.934)	800/856 (0.935)

Table 4.2: Sub-task execution success rates for pick and place tasks.

Task Completion Summary

	Sorting	Kitting	Overall
Pick Sub-Task Completion Rate	285/300 (0.950)	282/300 (0.940)	567/600 (0.945)
Place Sub-Task Completion Rate	288/300 (0.960)	288/300 (0.960)	576/600 (0.960)
Task Completion Rate	273/300 (0.910)	274/300 (0.913)	547/600 (0.912)

Table 4.3: Task completion rates for the sorting and kitting task scenarios.

Because the model had been trained to perform “pour” tasks and was modified to perform “place” for the evaluation setup, it occasionally exhibited erratic or non-deterministic behaviors when executing the place task. A common calibration issue was regarding the release positions of the cups and the small size of the bins used to hold them. Slightest execution noises in the simulation had the potential to cause the placed cup to fall out of the bin or knock out the existing cups in the bin. As this is not a direct reflection of the model’s performance but rather an unfortunate artifact of the simulation environment, we designed the accuracy measurement to be based on the position of the robot gripper at the time of release, instead of whether the placed cup stayed within the bin at task completion. In future iterations of this framework, the simulation environment can be adjusted to be able to better handle multiple place operations within the same bin.

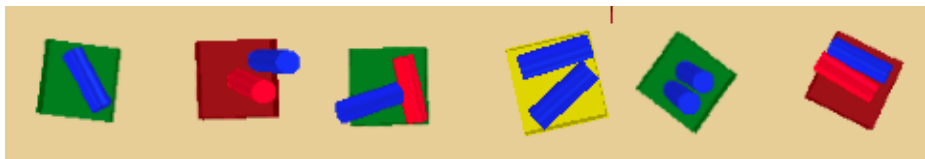


Figure 4-10: Examples of correctly executed place tasks.



Figure 4-11: Examples of place task inaccuracies during execution. These tasks were considered to be successful if the gripper was directly above the bin at release time.

4.3 Discussion and Limitations

We evaluated the model’s execution of two types of high-level pick-and-place tasks based on realistic use scenarios. Each module of the pipeline achieved high accuracy, and an overall task execution accuracy of 91.16% was achieved. While the proposed framework is shown to effectively communicate abstract sequential tasks to the robot in an end-to-end manner in a simulated environment, more improvements could be made to address some of its current limitations and facilitate the adaptation of this approach in real-life task scenarios.

4.3.1 Parser

The semantic command parser implemented in this work uses pre-defined grammar rules to parse the command into sub-tasks. As such, it cannot parse malformed or incomplete commands, which is common in natural language. The instruction parsing could instead be learned from a large number of unstructured verbal commands using an LTL formula [50, 31] to better handle noisy natural language inputs.

4.3.2 Detection

For simplicity, we constrained the features of the task objects in the testing environment to visual (color) and cardinal (count). However, the pre-trained object detection module is capable of identifying a greater variety of colors and numbers, as well as a number of different shapes and sizes. The model can be further expanded to handle spatial and relational features (e.g. leftmost cup, largest bin) [6].

4.3.3 Task Sequencing

If several possible target objects were present in the task scene, the model picked the one with the highest detection confidence to execute the task. This was an arbitrary metric we chose for the implementation and could be further constrained to perform the desired behavior for more specific task goals, such as picking the closest object. Similarly, the order of the sub-tasks to be performed was implicitly inferred from the input command: given the command “put the red cup and the blue cup in the dish”, the robot would first position the red cup, followed by the blue cup. This sub-task order selection process can also be further optimized for a particular metric, such as performing the task with minimum traveling distance or choosing the largest target object to position first for a “stacking” task. The task sequencing problem can be formulated as an MDP, or the constraints for the task goal could be learned directly from human demonstrations.

4.3.4 Grasping

In this work, object grasping was simplified to an open-or-close operation along the one-dimensional gripper state. In realistic scenarios, however, the task scene may contain objects of different shapes, sizes, and orientations, which would require inferring the correct grasping technique based on the target object’s features.

4.3.5 Motion Primitive

Sometimes the robot exhibited erratic behavior when the initial joint configuration and target object were not similar to any combination seen in the training data. One way to alleviate erratic behaviors with initial/final condition discrepancies in joint-space is to learn the motion primitive in task-space (Cartesian coordinates) and rely on inverse kinematics solvers to move the robot’s end-effector to the desired task-space locations. This could also improve the accuracy of the task-embedding \rightarrow control parameters mapping, as Cartesian coordinates are more interpretable than joint angles. Also, recall that the control model generates parameters for a time-

dependent DMP. If these are noisy, they may cause erratic behavior and make the robot move abruptly from one location to another or even exhibit unwanted cyclical motions. To address this issue, one could use a motion primitive that is solely state-dependent: $r_{t+1} = f_B(r_{t+1}, \theta_B)$, where θ_B are model parameters independent of time [19]. This would allow the robot to be robust to abrupt changes in the scene without needing to re-calculate phase variables or entire trajectories.

Chapter 5

Conclusion

Transferring task specifications to a robot is a difficult endeavor that often involves considerable supervision from human experts. In order for robots to be introduced to and operate efficiently in manufacturing environments, the task transfer process must be easily accessible for humans and robots alike. This requires an intuitive and robust mechanism for the human instructor to explain a task to the robot and have the instruction be correctly interpreted.

In this thesis, we designed and evaluated an end-to-end language-conditioned imitation learning framework for transferring sequential task instructions to the robot. The task specifications were provided in the form of image and natural language inputs, which were ultimately translated into low-level control parameters that the robot can execute in sequence. The natural language component precludes the need for an extensive knowledge of programming or robotics and enables the instructor to provide task instructions with relative ease. This approach was conceptually inspired by the therblig elemental motions, an analysis metric used in the study of motion economy in the (human) workplace.

The framework was implemented in part using pre-trained weights from a language-conditioned imitation learning framework designed to perform a single pick-and-pour task [46]. The evaluation of the framework was performed on task environments unseen during the training stage and produced comparable results to the original framework. This suggests that the proposed approach can be successfully adapted

to accommodate unseen task environments in order to perform more complex tasks than the ones achieved by the original model.

As discussed in Section 4.3, further adjustments can be made in the framework to improve its accuracy and generalizability to novel task environments. The instruction parsing could be learned from unstructured verbal commands to better handle noisy natural language inputs. The object detection module could be expanded so that the task scene can contain objects of different shapes, sizes, and orientations. This adaptation would also motivate the need to infer more complex grasping techniques based on the target object’s physical features. Finally, the sub-task sequencing problem can be optimized for a particular metric, such as achieving minimum traveling distance or performing a “stacking” task ordered by the object size.

Bibliography

- [1] Pooya Abolghasemi, Amir Mazaheri, Mubarak Shah, and Ladislau Bölöni. Pay attention! - robustifying a deep visuomotor policy through task-focused attention, 2018.
- [2] Tameem Adel, Zoubin Ghahramani, and Adrian Weller. Discovering interpretable representations for both deep generative and discriminative models. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 50–59. PMLR, 10–15 Jul 2018.
- [3] Lawrence S. Aft. *Work Measurement and Methods Improvement*. John Wiley & Sons, Inc., 2000.
- [4] Jacob Andreas. Measuring compositionality in representation learning. *CoRR*, abs/1902.07181, 2019.
- [5] Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.
- [6] Jacob Arkin, Thomas Howard, Rohan Paul, and Nicholas Roy. Efficient grounding of abstract spatial concepts for natural language interaction with robot manipulators. *The International Journal of Robotics Research*, 37, 01 2016.
- [7] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. *CoRR*, abs/1206.5538, 2012.
- [8] Aude Billard, Sylvain Calinon, Rüdiger Dillmann, and Stefan Schaal. *Robot Programming by Demonstration*, pages 1371–1394. 01 2008.
- [9] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit*. O’Reilly Media, Inc., 2009.
- [10] Jacqueline Brixey, Ramesh Manuvinakurike, Nham Le, Tuan Lai, Walter Chang, and Trung Bui. A system for automated image editing from natural language commands, 2018.

- [11] Crystal Chao, Maya Cakmak, and Andrea L. Thomaz. Towards grounding concepts for transfer in goal learning from demonstration. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, pages 1–6, 2011.
- [12] Devendra Singh Chaplot, Kanthashree Mysore Sathyendra, Rama Kumar Pasumarthi, Dheeraj Rajagopal, and Ruslan Salakhutdinov. Gated-attention architectures for task-oriented language grounding, 2018.
- [13] David Chen and Raymond Mooney. Learning to interpret natural language navigation instructions from observations. volume 1, 01 2011.
- [14] Lingzhen Chen and Alessandro Moschitti. Transfer learning for sequence labeling using source model and target data, 2019.
- [15] Xi Chen, Yan Duan, Rein Houthoofd, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *CoRR*, abs/1606.03657, 2016.
- [16] Wikimedia Commons. The 18 therbligs, 2012. File: Therblig (English).svg.
- [17] Abhishek Das, Georgia Gkioxari, Stefan Lee, Devi Parikh, and Dhruv Batra. Neural modular control for embodied question answering, 2019.
- [18] Staffan Ekvall and Danica Kragic. Robot learning from demonstration: A task-level planning approach. *International Journal of Advanced Robotic Systems*, 5(3):33, 2008.
- [19] Nadia Figueroa and Aude Billard. A physically-consistent bayesian non-parametric mixture model for dynamical system learning. In Aude Billard, Anca Dragan, Jan Peters, and Jun Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, volume 87 of *Proceedings of Machine Learning Research*, pages 927–946. PMLR, 29–31 Oct 2018.
- [20] Sergio Guadarrama, Lorenzo Riano, Dave Golland, Daniel Goehring, Yangqing Jia, Dan Klein, Pieter Abbeel, and Trevor Darrell. Grounding spatial relations for human-robot interaction. pages 1640–1647, 11 2013.
- [21] Yordan Hristov, Daniel Angelov, Michael Burke, Alex Lascarides, and Subramanian Ramamoorthy. Disentangled relational representations for explaining and learning from demonstration. *CoRR*, abs/1907.13627, 2019.
- [22] Auke Jan Ijspeert, Jun Nakanishi, Heiko Hoffmann, Peter Pastor, and Stefan Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, 25(2):328–373, 2013.
- [23] Stephen James, Marc Freese, and Andrew J. Davison. Pyrep: Bringing v-rep to deep robot learning, 2019.

- [24] Michael Janner, Karthik Narasimhan, and Regina Barzilay. Representation learning for grounded spatial reasoning, 2017.
- [25] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. pages 259–266, 03 2010.
- [26] Dana Kulic, Danica Kragic, and Volker Kr. Learning action primitives. pages 333–353, 01 2011.
- [27] Deepika Kumawat and Vinesh Jain. Pos tagging approaches: A comparison. *International Journal of Computer Applications*, 118(6), 2015.
- [28] Miguel Lázaro-Gredilla, Dianhuan Lin, J. Swaroop Guntupalli, and Dileep George. Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs. *CoRR*, abs/1812.02788, 2018.
- [29] Séverin Lemaignan, Raquel Ros, Emrah Akin Sisbot, Rachid Alami, and Michael Beetz. Grounding the Interaction: Anchoring Situated Discourse in Everyday Human-Robot Interaction. *International Journal of Social Robotics*, 4(2):pp.181–199, April 2012. 20 pages.
- [30] Bowen Li, Xiaojuan Qi, Thomas Lukasiewicz, and Philip H. S. Torr. Manigan: Text-guided image manipulation, 2020.
- [31] Shen Li, Daehyung Park, Yoonchang Sung, Julie A. Shah, and Nicholas Roy. Reactive task and motion planning under temporal logic specifications, 2021.
- [32] Lluís Marquez, Lluís Padro, and Horacio Rodríguez. A machine learning approach to pos tagging. *Machine Learning*, 39(1):59–91, 2000.
- [33] Dipendra Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35, 11 2015.
- [34] Seonghyeon Nam, Yunji Kim, and Seon Joo Kim. Text-adaptive generative adversarial networks: Manipulating images with natural language, 2018.
- [35] Sriraam Natarajan, Saket Joshi, Prasad Tadepalli, Kristian Kersting, and Jude Shavlik. Imitation learning in relational domains: A functional-gradient boosting approach. pages 1414–1420, 01 2011.
- [36] Takayuki Osa, Joni Pajarinen, Gerhard Neumann, J. Andrew Bagnell, Pieter Abbeel, and Jan Peters. An algorithmic perspective on imitation learning. *Foundations and Trends in Robotics*, 7(1-2):1–179, 2018.
- [37] Jeffrey Pennington, Richard Socher, and Christopher Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics.

- [38] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020.
- [39] Harish Ravichandar, Athanasios S. Polydoros, Sonia Chernova, and Aude Billard. Recent advances in robot learning from demonstration. *Annual Review of Control, Robotics, and Autonomous Systems*, 3(1):297–330, 2020.
- [40] Scott Reed, Zeynep Akata, Xinchen Yan, Lajanugen Logeswaran, Bernt Schiele, and Honglak Lee. Generative adversarial text to image synthesis, 2016.
- [41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.
- [42] Junha Roh, Chris Paxton, Andrzej Pronobis, Ali Farhadi, and Dieter Fox. Conditional driving from natural language instructions, 2019.
- [43] Eric Rohmer, Surya Singh, and Marc Freese. V-rep: A versatile and scalable robot simulation framework. pages 1321–1326, 11 2013.
- [44] Ankit Shah, Shen Li, and Julie Shah. Planning with uncertain specifications (PUnS). *IEEE Robotics and Automation Letters*, 2020.
- [45] Seitaro Shinagawa, Koichiro Yoshino, Sakriani Sakti, Yu Suzuki, and Satoshi Nakamura. Interactive image manipulation with natural language instruction commands, 2018.
- [46] Simon Stepputtis, Joseph Campbell, Mariano Phielipp, Stefan Lee, Chitta Baral, and Heni Ben Amor. Language-conditioned imitation learning for robot manipulation tasks, 2020.
- [47] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. volume 2, 01 2011.
- [48] Pavel Tokmakov, Yu-Xiong Wang, and Martial Hebert. Learning compositional representations for few-shot recognition. *CoRR*, abs/1812.09213, 2018.
- [49] Huihsin Tseng, Dan Jurafsky, and Christopher D Manning. Morphological features help pos tagging of unknown words across language varieties. In *Proceedings of the fourth SIGHAN workshop on Chinese language processing*, 2005.
- [50] Christopher Wang, Candace Ross, Yen-Ling Kuo, Boris Katz, and Andrei Barbu. Learning a natural-language to ltl executable semantic parser for grounded robotics, 2021.
- [51] Mary-Anne Williams, John McCarthy, Peter Gärdenfors, Christopher J. Stanton, and Alankar Karol. A grounding framework. *Auton. Agents Multi Agent Syst.*, 19(3):272–296, 2009.

- [52] Tao Xu, Pengchuan Zhang, Qiuyuan Huang, Han Zhang, Zhe Gan, Xiaolei Huang, and Xiaodong He. AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks, 2017.
- [53] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. Contrastive learning of medical visual representations from paired images and text, 2020.