Neuro-Symbolic Reinforcement Learning: A Robotic Manipulator Platform & Multi-Task Agent

Hunter W. Ellis

Thesis submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Engineering

Thinh T. Doan, PhD, Chair

Michael S. Hsiao, PhD, Co-chair

Ryan K. Williams, PhD

March 23, 2016

Blacksburg, Virginia

Keywords: Reinforcement Learning, Neuro-symbolic, Neuro-symbolic Concept Learner, Multi-task Reinforcement Learning, Robotics

Neuro-Symbolic Reinforcement Learning: A Robotic Manipulator Platform & Multi-Task Agent

Hunter W. Ellis

(ABSTRACT)

In recent years, neuro-symbolic learning methods have demonstrated promise in tasks requiring a semantic understanding that can often be missed by traditional deep learning techniques. By integrating symbolic reasoning with deep learning architectures the interpretability of the model's reasoning becomes more evident and can provide more control during deployment. This thesis aims to apply neuro-symbolic learning to the domain of reinforcement learning. First, a simulation environment for robotic manipulation tasks based on the Gazebo Harmonic physics simulator and ROS2 middleware suite is presented. In this environment an analysis of policy-gradient based reinforcement learning algorithm is given. Then, by leveraging the performance of deep learning with the semantic reasoning and interpretability of symbolically defined programming, a novel neuro-symbolic learning method is proposed to generalize tasks and motion planning for robotics applications using natural language. This novel neuro-symbolic can be seen as an adaptation of the Neuro-Symbolic Concept Learner (Mao et. al) developed by IBM Watson, in which images and natural language are first processed by convolutional and residual neural networks, respectively and then parsed by a symbolically reasoned program. Where the architecture proposed in this paper differs, is in its use of the Neuro-Symbolic Concept Learner for preprocessing of a given input task, to then inform a reinforcement learning agent of how to act in a given environment. Finally, the novel adaptation of the Neuro-Symbolic Concept Learner is introduced as a method of controlling multi-task agents.

Neuro-Symbolic Reinforcement Learning: A Robotic Manipulator Platform & Multi-Task Agent

Hunter W. Ellis

(GENERAL AUDIENCE ABSTRACT)

Neuro-symbolic learning is an area in machine learning that leverages user defined symbolic programming in addition to deep learning. This method goes against the typical approach of end-to-end training of models and instead hopes to benefit from the introduction of symbolic programs.

Dedication

This is where you put your dedications.

Acknowledgments

This is where you put your acknowledgement.

Contents

List of Figures

List of Tables

List of Abbreviations

AI Artificial Intelligence

DDPG Deep Deterministic Policy Gradient

HER Hindsight Experience Replay

NeSy Neuro-Symbolic

NLP Natural Language Processing

NS-CL Neuro-Symbolic Concept Learner

NSRL Neuro-Symbolic Reinforcement Learning

RAMPS RepRap Arduino Mega Pololu Shield

RL Reinforcement Learning

UART Universal Asynchronous Receiver/Transmitter

USB Universal Serial Bus

Chapter 1

Introduction

Neuro-symbolic learning methods and concepts have been used in recent years to achieve results that standalone deep learning and/or symbolic programming methods have not been able to achieve. These hybrid systems combine the pattern recognition and generalization capabilities of neural networks with the structured, rule-based reasoning of symbolic systems. For example, neuro-symbolic approaches have been successfully applied in visual question answering, knowledge base completion, and mathematical reasoning—areas where pure neural methods may struggle with logical consistency, and pure symbolic methods may lack adaptability (Besold et al., 2017; Mao et al., 2019). This synergy provides a framework for intelligent systems that are both robust and explainable.

The emerging developments in the field of neuro-symbolic learning have created opportunities to explore a wide range of applications and adaptations. Researchers have begun to integrate symbolic structures into neural models to improve generalization, transfer learning, and sample efficiency (Garcez et al., 2019). Conversely, symbolic reasoning systems are now being augmented with learned components to better handle ambiguity and noisy inputs. Such combinations have shown promise in domains like natural language understanding, program synthesis, and robotics, where both structured knowledge and perceptual learning are crucial (Evans & Grefenstette, 2018). These developments suggest that hybrid approaches can fill critical gaps left by each paradigm alone.

Meanwhile, the field of Reinforcement Learning (RL) has made significant strides, demon-

strating the paradigm's effectiveness in creating agents that can autonomously perform complex tasks, such as playing strategic games, controlling robotic limbs, or navigating real-world environments (Mnih et al., 2015; Silver et al., 2017). RL agents learn through trial and error, developing sophisticated behavior through interactions with their environment. However, these agents often lack transparency in their decision-making processes, making them difficult to interpret and trust. This lack of interpretability limits their use in high-stakes or regulated domains where understanding and explaining behavior is essential.

The intersection of neuro-symbolic learning and reinforcement learning has opened new avenues for developing agents that are both high-performing and interpretable. By integrating symbolic reasoning into RL agents, it is possible to create systems that not only learn effective policies but also provide insights into their decision-making processes (Zambaldi et al., 2019). Ideally, a neuro-symbolic RL agent should be flexible in its ability to capture complex non-linear relationships while remaining interpretable in its reasoning and behavior. This blend can enable more reliable, transparent, and generalizable intelligent systems—paving the way for safer deployment in real-world environments, including healthcare, autonomous vehicles, and scientific discovery.

1.1 Objectives

The primary objective of this thesis is to develop and evaluate a neuro-symbolic reinforcement learning (NSRL) framework that combines the perceptual capabilities of deep learning with the interpretability and structure of symbolic reasoning. The goal is to demonstrate that such a hybrid approach can yield agents that are both effective in complex task environments and more transparent in their decision-making processes. By integrating symbolic abstraction into the RL observation space, this work aims to improve the explainability, modularity,

and control of learning agents—key properties that are often lacking in traditional deep reinforcement learning approaches.

This thesis proposes an adaptation of the Neuro-Symbolic Concept Learner (NS-CL) architecture, originally designed for visual question answering, for use within a reinforcement learning context. Rather than employing NS-CL purely as an output interpreter, the architecture is repurposed to act as a semantic filter that preprocesses raw sensory input into symbolic representations. These symbolic augmentations are then used to guide the learning process of the RL agent, allowing it to reason about tasks using interpretable, structured representations rather than unstructured pixel-level inputs.

Additionally, a secondary objective is to develop a robotic manipulation platform and simulation environment tailored to the evaluation of multi-task RL agents. This environment is designed to support symbolic task definitions and natural language prompts, providing a testbed for validating the generalization and flexibility of the proposed neuro-symbolic reinforcement learning framework. This enables the agent to perform real-world inspired tasks such as object sorting, placement, and navigation based on semantic goals.

Ultimately, this research aims to contribute a practical demonstration of neuro-symbolic integration within reinforcement learning and robotics. The outcome should highlight how symbolic reasoning can complement learned behavior, and offer insights into how structured knowledge can be used to improve safety, generalization, and usability in autonomous agents.

1.2 Applications

Neuro-symbolic reinforcement learning has a wide range of potential applications, particularly in domains that require both perceptual learning and structured, interpretable decision-

making. The integration of symbolic reasoning into reinforcement learning agents enhances their capacity to generalize across tasks, follow abstract instructions, and explain their actions—capabilities that are increasingly important in real-world systems. This section outlines key application areas, with a focus on robotic manipulation tasks in industrial settings.

1.2.1 Assembly Line Automation

One of the most promising applications of this work is in industrial automation, specifically in robotic assembly lines. Traditional automation systems rely on rigid, pre-programmed rules that often lack adaptability to dynamic environments. Neuro-symbolic RL agents, by contrast, can adapt to changing contexts while maintaining a symbolic understanding of tasks, objects, and goals. For example, an agent might be instructed to "place all red cubes in the bin," and using symbolic augmentation, it can reason about object properties and execute appropriate actions—even when encountering variations in object placement or appearance. This level of semantic reasoning is crucial for enabling multi-task, reconfigurable robotic systems that reduce downtime and human oversight in manufacturing processes.

1.2.2 Mulit-Task Robotic Control

The ability to perform multiple tasks within a single environment is another major benefit of neuro-symbolic reinforcement learning. In traditional RL, agents are often trained for narrow, single-task performance. By contrast, a neuro-symbolic architecture allows for abstract task representations that can be shared and reused across different behaviors. For instance, an agent trained to "move the object to the left of the blue cylinder" can generalize that reasoning to a different object or spatial relationship using symbolic structures.

This capability has implications for service robotics, warehouse automation, and assistive technologies, where flexible task understanding is vital.

1.2.3 Natural Language Grounding

Another application enabled by this architecture is the grounding of natural language commands into symbolic forms that inform reinforcement learning policies. This thesis demonstrates how natural language prompts can be interpreted by a symbolic module and used to condition the agent's behavior. This has broad implications in human-robot interaction, where instructing a robot through high-level language—rather than low-level programming—is desirable. Applications could range from domestic robotics to collaborative manufacturing, where non-expert users need to communicate tasks to autonomous systems in an intuitive and interpretable way.

1.3 Scope

1.4 Contributions

Chapter 2

Background

This chapter presents the foundational concepts that support the methods and systems used in this thesis. This thesis is at the intersection of robotics, simulation, reinforcement learning, and neuro-symbolic AI. Each of these areas is described below in the context of this work to provide background on the development of the robotic test platform and proposed neuro-symbolic model.

2.1 Robotic Manipulators

Robotic manipulators are commonly used in industrial and research applications due to their precision and versatility. These systems typically consist of multiple joints (degrees of freedom) that allow an end effector to move and manipulate objects in 3D space. This thesis focuses on a 6-axis robotic manipulator, capable of performing complex tasks such as grasping, moving, and orienting objects. The flexibility of these robots makes them ideal for testing reinforcement learning algorithms, especially in multi-task settings.

2.1.1 6-Axis Robotic Manipulators

Six-axis arms offer a rich action space and are commonly used in industrial automation. Their joints—usually revolute—enable motion in three-dimensional space, with pitch, yaw, and roll

control over the end effector. These properties make them suitable for fine-grained motion planning and manipulation tasks, especially when paired with a learning-based controller.

2.2 Control Software & Simulation

To safely develop and test control algorithms, robotic systems are often simulated before being deployed in real-world settings. This thesis leverages the Robot Operating System (ROS2) middleware and the Gazebo Harmonic physics simulator to create a modular, testable robotic platform.

2.2.1 Robot Operating System (ROS2)

ROS2 is a widely adopted middleware framework that supports distributed communication, modular design, and hardware abstraction for robotic systems. It allows for flexible integration between control algorithms, sensors, and actuators, making it an ideal tool for reinforcement learning research.

2.2.2 Gazebo Simulation Environment

Gazebo provides realistic physics-based simulation of robots and their environments. It supports a variety of sensors, actuators, and physics engines, making it a powerful tool for testing learning-based control in a safe and controlled environment. In this work, Gazebo Harmonic is used to simulate the physical properties of the 6-axis robot and its interaction with objects.

2.3 Reinforcement Learning

Reinforcement learning (RL) is a machine learning paradigm that emphasizes goal-oriented learning from interactions in an environment [**?**]. In the RL paradigm agents act in an environment and recieve rewards from the environment. A high reward informs the agent that it's actions are beneficial to the objective, where as a low reward tells the agent that it is acting poorly in the environment, with respect to the objective. These agents attempt to recieve a high reward by taking an input observation state and developing an optimal policy to predict actions that will return a high reward.

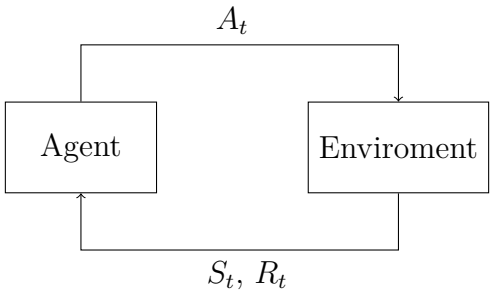

Figure 2.1: Agent-environment interaction loop

In practice, this means that at each timestep, an agent observes a state, performs an action, and receives a reward based on the outcome. Over time, a well designed agent will learn a policy that finds or closely approximates the optimal policy [**?**].

2.3.1 Markov Decision Process

Environments are typically defined as an Markov Decision Process (MDP). There are four main components that make up an MDP, states S, actions A, transition probability P, and reward R.

2.3.2 Finding an Optimal Policy

To *solve* an MDP we attempt to find an optimal policy π^*.

2.3.3 Deep Reinforcement Learning

Deep RL leverages deep neural networks to approximate policies or value functions in high-dimensional state spaces. It allows agents to learn directly from raw inputs like images or sensor data. Algorithms like DQN and PPO have been successful in a variety of domains, but often lack interpretability.

2.3.4 Deep Deterministic Policy Gradient

DDPG is an off-policy actor-critic algorithm designed for continuous action spaces [**?**]. It combines value-based and policy-based methods by training a deterministic policy alongside a Q-function. In this work, DDPG is used as the primary reinforcement learning method for robotic control tasks.

2.3.5 Hindsight Experience Replay

HER enhances sample efficiency in sparse reward environments by relabeling failed experiences as successful ones based on alternative goals [**?**]. This allows agents to learn from episodes that would otherwise provide little learning signal. HER is particularly useful in goal-conditioned tasks such as pick-and-place.

2.4 Neuro-symbolic AI

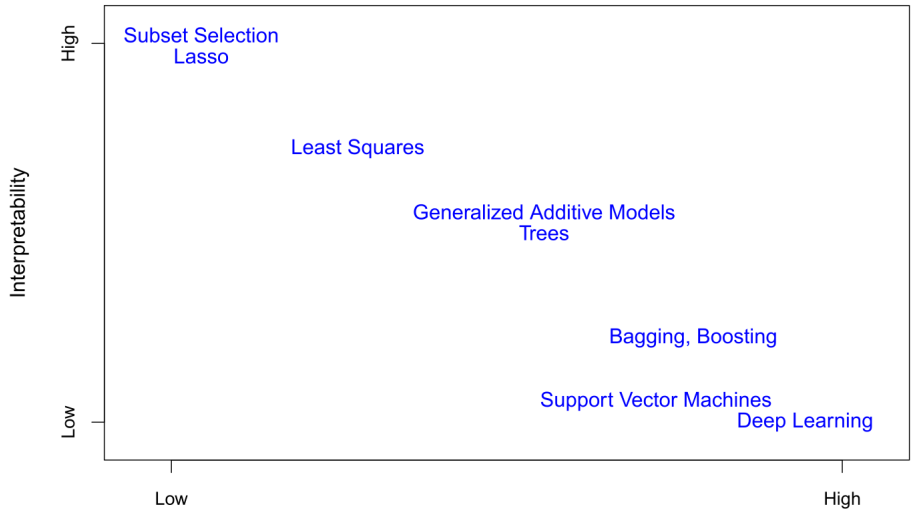

Figure 2.2: Trade-off between interpretabilty and model flexibility [**?**]

Neuro-symbolic models can be broadly defined as models which focus on merging both *neural* and *symbolic* AI approaches to add value to the system. The term *nerual* refers to the use of artificial neural networks and the term *symbolic* typically refers to approaches based on explicit symbol manipulation. (CITATION taxonomy.pdf) The fundamental desire of neuro-symbolic models is to leverage both neural and symbolic approaches in a way that is favorable to the strengths of each approach and unfavorable to their weaknesses. The strengths neural models would include the ability to leverage raw data that may be too dificult/complex to semantically reason about, while the weaknesses may include the challenges faced when attempting to reason neural models. Conversely, the strengths of symbolic models include their ability to be highly explainable and verifiable, but they may have weaknesses in their reliance human input/understanding of a system. Thus, the promise of a neuro-symbolic architecture is a system which would be robust from training data, symbolically explainable, and be able to leverage human expert knowledge in its design.

In recent years many different neuro-symbolic models have been designed and deployed. Because the intersection of these two approaches may be quite broad and models can take many different forms be it is important to make distinctions between the various incarnations of neuro-symbolic architectures. Henry Kautz, in a 2005 article presents "six possible designs" patterns classifying each method in reference to their neural and symbolic interactions:(CITATION KAUTZ)

2.4.1 Classification of Neuro-symbolic Architectures

Neuro-symbolic Design Pattern	**Definition**	**Example**
Symbolic Neural symbolic	A symbolic input is fed into a neural network producing a symbolc output.	GPT
Symbolic[Neural]	A neural subroutine is evoked by a symbolic strategy.	AlphaGo
Neural[Symbolic]	A symbolic subroutine is evoked by a neural strategy.	ChatGPT accessing Wolfram for computations
Neural|Symbolic	A neural network converts a non-symbolic input into symbolic data to be symbolically processed.	Neuro-Symbolic Concept Learner
Neural:Symbolic → Neural	A symbolically represented dataset is used to train an neural network to predict a symbolic output	ANN-MPC
Neural_{Symbolic}	Symbolic rules are used to define the structures making up the neural network.	Logic Tensor Networks

2.4.2 Neuro-symbolic Concept Learner

The Neuro-Symbolic Concept Learner (NS-CL), developed by Mao et al., is a representative example of the Neural|Symbolic pattern. It combines convolutional and recurrent networks to extract structured representations from images and language, which are then processed by a symbolic reasoning module. This thesis adapts NS-CL to preprocess environment

observations and task instructions for reinforcement learning.

2.5 Neuro-symbolic Reinforcement Learning

Neuro-symbolic reinforcement learning (NSRL) is an emerging field that merges symbolic abstraction with RL policies. The aim is to produce agents that can learn effectively from data while providing semantic interpretability. NSRL allows for modular design, generalization across tasks, and the integration of external knowledge sources (e.g., task graphs or logic rules). In this thesis, NSRL is explored through a novel adaptation of NS-CL to guide robotic agents using natural language prompts and symbolic scene representations.

Chapter 3

Robotic Manipulator Robotic Manipulator Platform

Given the problem space, a real-world and cooresponding simulated environment was created as a platform to train and test reinforcement learning algorithms for the neuro-symbolic architecture described in this paper and general reinforcement learning algorithms involving robotic manipulators. The physical hardware used for this robotic manipulator is based on an open-source hardware design[CITATION], with modifications made to reduce cost while maintaining performance. The cooresponding simulation and control of the manipulator was developed using the ROS2 middleware suite and nd Gazebo simulation enviroment enabling a flexible and reproducable setup for training, validating, and deploying algorithms.

3.1 Robotic Manipulator Hardware

The physical hardware used to test the algorithms is made up of a 6-axis robot arm communicating over serial (UART) with a PC running the algorithms in a ROS2 Jazzy workspace. The rigid components of the arm including gears and structural pieces are made of 3D-printed Polylactic Acid (PLA). The robotic arm uses stepper motors, belts, and pulleys to articluate the 6 joints. The first 5 joints (J1 to J5) use bipolar NEMA 17 stepper motors, while the last joint J6, responsible for manipulating the end effector, uses a bipolar NEMA

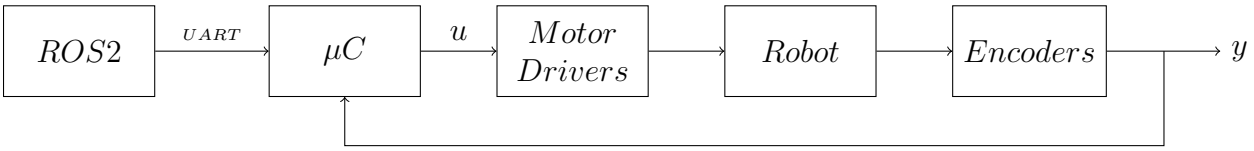

Figure 3.1: Robot servo feedback system.

8 stepper motor. The belts and pulleys are "off-the-shelf" GT2 timing belts and pulleys of varying sizes, used to the torque applied to each joint (belts and pulleys connect every motor to a joint with the exception of J6). Ball and shunt bearings of verious sizes are also used to reduce friction in the joints.

The electronic hardware is controlled by an ATmega328p microcontoller and 6 A4988 stepper motor drivers originally setup for controlling the stepper motors of a 3D-Printer. Custom firmware was written for the microcontroller to run the 6 stepper motor drivers simultaineously with a serial (UART) inturrupt to recieve control commands from the host PC.

3.1.1 Inverse Kinematics

3.1.2 Motor Control

Bipolar stepper motors are used to move the joints of the robot, choosen for their low cost and wide availability. Stepper motors have

A hobbiest 3D-printer stepper motor controller with custom firmware was used to reduce cost.

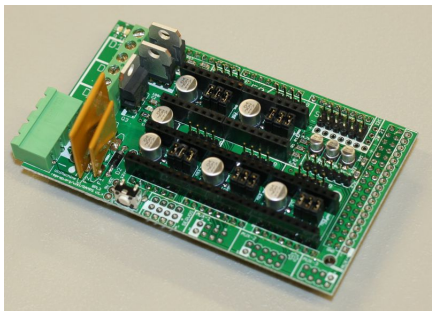

Figure 3.2: RAMPS board

3.1.3 Communication

Communication is done over a USB-UART connection, sending ROS2 position commands to a ATmega328p microcontroller used to process the position requests and update the joint positions.

3.2 Robotic Manipulator Software

The environment consists of the same six axis robot arm described in the previous section set up in the Gazebo Robotic Simulator (using a ROS Universal Robot Description File and Gazebo Simulation Description File). Within the arm's working envelop various basic 3-D shapes (i.e. spheres, cubes, cylinders, etc.) are present.

The Robot Operating System is a middleware suite used for robot software development. ROS workspaces consist of packages that interface with ROS libraries.

3.2.1 Robot Description

The `arm_description` package is a ROS package that contains various files used to describe the physcial characteristics of the robot including its visual, collision, control, and forward

kinematics.

The six-axis robotic arm model deployed in this package is a slightly modified version of an open source six-axis robot design with modfications made to some of the pulleys and end-effector design. The robot arm is made up of six joints (J1-J6) all described as revolute joints in the `arm_description`'s Universal Robot Description File (`.urdf`). Meshes for rendering the robot imported as `.stl` files and the meshes for collision areas are described by COLLADA (`.dae`) files. These meshes are linked together in the URDF in a kinematic chain to form the robotic arm manipulator. Control of the robot is accomplished through the ROS Jazzy control package.

3.2.2 Controllers

3.2.3 Hardware Interface

3.3 Reinforcement Learning Environment

Chapter 4

Neuro-symbolic Reinforcement Learning Model

4.1 Achitecture Overview

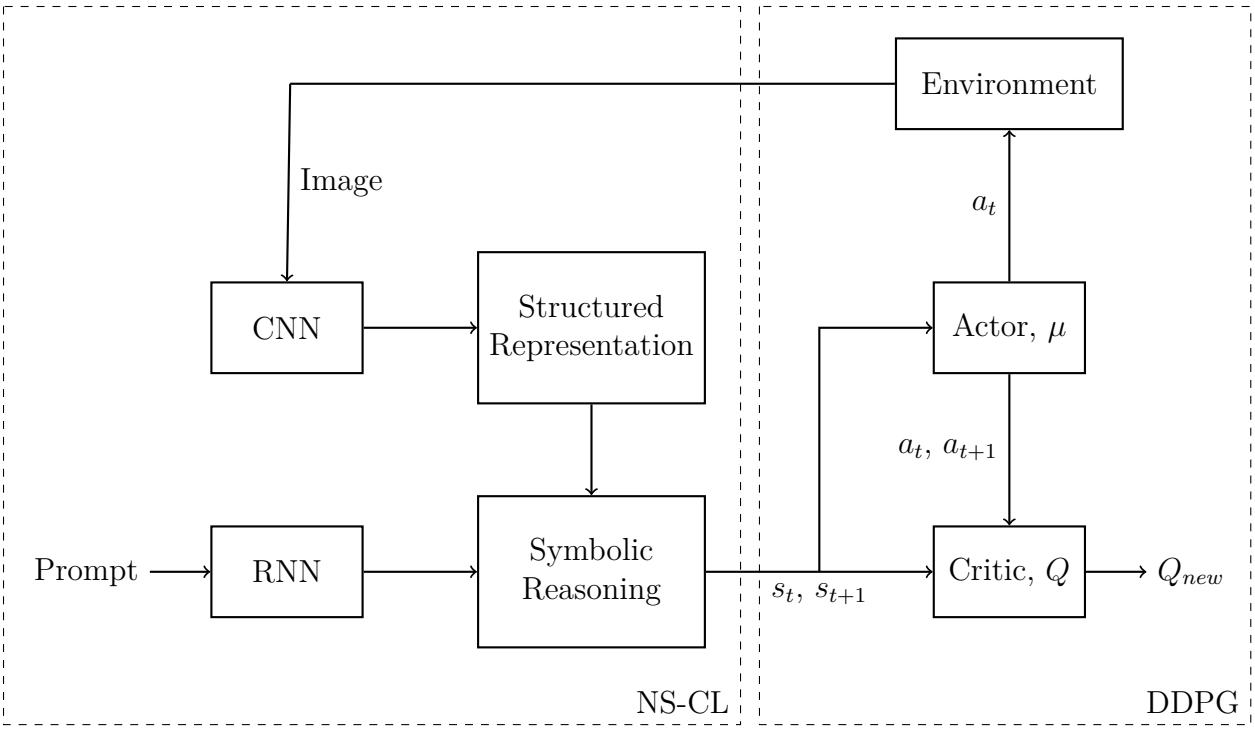

Figure 4.1: NS-CL with proposed RL adaptation.

Shape	**Color** (r, g, b)	**Position** (x, y)	**Rotation** θ	**Velocity** (v_x, v_y, v_z, ω)
J1	(0, 0, 0)	(0, 0, 0)	0°	(0, 0, 0, 0)
J2	(0, 0, 0)	(0, 0.1, 0)	0°	(0, 0, 0, 0)
J3	(0, 0, 0)	(0, 0.2, 0)	0°	(0, 0, 0, 0)
J4	(0, 0, 0)	(0, 0.4, 0)	0°	(0, 0, 0, 0)
J5	(0, 0, 0)	(0, 0.4, 0)	0°	(0, 0, 0, 0)
J6	(0, 0, 0)	(0, 0.4, 0)	0°	(0, 0, 0, 0)
cube	(255, 0, 255)	(0.75, 0.75, 0)	0°	(0, 0, 0, 0)
cylinder	(255, 255, 0)	(-0.25, 1, 0)	0°	(0, 0, 0, 0)
rectangle	(0, 0, 255)	(0.5, 0.5, 0)	90°	(0, 0, 0, 0)
sphere	(0, 255, 0)	(0.6, -0.1, 0)	0°	(0, 0, 0, 0)

Table 4.1: Example of state observation before symbolic modification.

4.2 Computer Vision

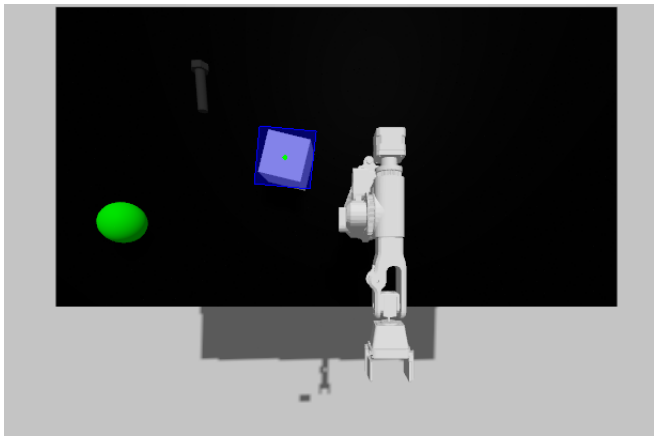

Figure 4.2: YOLOv8–OBB in Gazebo Simulation

Label	**Color** (r, g, b)	**Position** (x, y)	**Rotation** θ	**Velocity** (v_x, v_y, v_z, ω)	**Symbol**
J1	(0, 0, 0)	(0, 0, 0)	0°	(0, 0, 0, 0)	null
J2	(0, 0, 0)	(0, 0.1, 0)	0°	(0, 0, 0, 0)	null
J3	(0, 0, 0)	(0, 0.2, 0)	0°	(0, 0, 0, 0)	null
J4	(0, 0, 0)	(0, 0.4, 0)	0°	(0, 0, 0, 0)	null
J5	(0, 0, 0)	(0, 0.4, 0)	0°	(0, 0, 0, 0)	null
J6	(0, 0, 0)	(0, 0.4, 0)	0°	(0, 0, 0, 0)	null
cube	(255, 0, 255)	(0.75, 0.75, 0)	0°	(0, 0, 0, 0)	null
cylinder	(255, 255, 0)	(-0.25, 1, 0)	0°	(0, 0, 0, 0)	avoid
rectangle	(0, 0, 255)	(0.5, 0.5, 0)	90°	(0, 0, 0, 0)	move
sphere	(0, 255, 0)	(0.6, -0.1, 0)	0°	(0, 0, 0, 0)	null
goal	(0, 0, 0)	(-1, 1, 0)	0°	(0, 0, 0, 0)	goal

Table 4.2: Example of symbolically modified state.

4.2.1 YOLOv8–OBB

4.3 Natural Language Processing

4.4 State Representation

4.4.1 Symbolic Augmentations

4.5 Reinforcement Learning

4.6 Simulation Environment

Gazebo is a robotic physics simulator developed by Open Robotics which integrates the ODE physics engine, ORGE rendering engine, and support code for sensor and actuator control integration. This environment leverages Gazebo Harmonic (the latest release of Gazebo at the time of writing) to visually render and simulate the physics of the robotic manipulator and the objects in its environment.

4.6.1 Pick and Place Simulation

4.6.2 Dataset

The cooresponding dataset used to train the model that can be seen as an extension of the CLEVR dataset [CITATION]– which instead prompts the agent to act on the objects in the environment.

Chapter 5

Results

5.1 Ablation Study

Chapter 6

Discussion

Chapter 7

Conclusion