Combining Non-monotonic Logical Reasoning with Data-driven Learning for Scene Understanding and Transparent Decision Making

Tiago Andrade Mota

A thesis submitted in partial fulfilment of the requirements for the degree of
Doctor of Philosophy in Electrical and Computer Engineering,
The University of Auckland, New Zealand.

May 2020

**This thesis is for examination purposes only,
and is confidential to the examination process.**

School of Graduate Studies
AskAuckland Central
Alfred Nathan House
The University of Auckland
Tel: +64 9 373 7599 ext 81321
Email: postgradinfo@auckland.ac.nz

Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 3 of this thesis is based on our published: Tiago Mota and Mohan Sridharan. "Incrementally Grounding Expressions for Spatial Relations between Objects". International Joint Conference on Artificial Intelligence (IJCAI 2018).	
Nature of contribution by PhD candidate	Design and implement algorithms/architecture, experimentally evaluate hypotheses, and write the paper.
Extent of contribution by PhD candidate (%)	65

CO-AUTHORS

Name	Nature of Contribution
Mohan Sridharan	Help design algorithms/architecture, identify hypotheses for evaluation, set up experiments, and write paper.

Certification by Co-Authors

The undersigned hereby certify that:

- ❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- ❖ that the candidate wrote all or the majority of the text.

Name	Signature	Date
Mohan Sridharan	S. Sridharan	March 15, 2020

School of Graduate Studies
AskAuckland Central
Alfred Nathan House
The University of Auckland
Tel: +64 9 373 7599 ext 81321
Email: postgradinfo@auckland.ac.nz

Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 4 of this thesis is based on our published: Tiago Mota and Mohan Sridharan. "Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots". Robotics Science and Systems Conference (RSS 2019).	
Nature of contribution by PhD candidate	Design and implement algorithms/architecture, experimentally evaluate hypotheses, and write the paper.
Extent of contribution by PhD candidate (%)	65

CO-AUTHORS

Name	Nature of Contribution
Mohan Sridharan	Help design algorithms/architecture, identify hypotheses for evaluation, set up experiments, and write paper.

Certification by Co-Authors

The undersigned hereby certify that:

- ❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- ❖ that the candidate wrote all or the majority of the text.

Name	Signature	Date
Mohan Sridharan	S. Mohan	March 15, 2020

THE UNIVERSITY OF AUCKLAND
Te Whare Wānanga o Tāmaki Makaurau
NEW ZEALAND

School of Graduate Studies
AskAuckland Central
Alfred Nathan House
The University of Auckland
Tel: +64 9 373 7599 ext 81321
Email: postgradinfo@auckland.ac.nz

Co-Authorship Form

This form is to accompany the submission of any PhD that contains published or unpublished co-authored work. **Please include one copy of this form for each co-authored work**. Completed forms should be included in all copies of your thesis submitted for examination and library deposit (including digital deposit), following your thesis Acknowledgements. Co-authored works may be included in a thesis if the candidate has written all or the majority of the text and had their contribution confirmed by all co-authors as not less than 65%.

Please indicate the chapter/section/pages of this thesis that are extracted from a co-authored work and give the title and publication details or details of submission of the co-authored work. Chapter 5 of this thesis is based on our published: Tiago Mota, Mohan Sridharan and Ales Leonardis. "Integrated Commonsense Reasoning and Deep Learning for Transparent Decision Making in Robotics". 17th European Conference on Multi-agent Systems (EUMAS 2020).	
Nature of contribution by PhD candidate	Lead the design of algorithms/architecture, experimental evaluation of hypotheses, and writing of the paper..
Extent of contribution by PhD candidate (%)	65

CO-AUTHORS

Name	Nature of Contribution
Mohan Sridharan	Help design algorithms/architecture, identify hypotheses for evaluation, set up experiments, and write paper.
Ales Leonardis	Provide feedback to improve the paper.

Certification by Co-Authors

The undersigned hereby certify that:

- ❖ the above statement correctly reflects the nature and extent of the PhD candidate's contribution to this work, and the nature of the contribution of each of the co-authors; and
- ❖ that the candidate wrote all or the majority of the text.

Name	Signature	Date
Mohan Sridharan	S. Mohan	April 22, 2020
Ales Leonardis	Ales Leonardis	April 24, 2020

Abstract

To truly assist humans in complex domains, robots need to address the challenges posed by dynamically changing environments. Reasoning with incomplete domain knowledge and uncertain data from sensors and actuators leads to incorrect decisions. Incremental learning and revision of this knowledge can help mitigate these problems. Deep networks represent the state of the art for many learning problems in robotics and AI. However, these networks require large training datasets, make it challenging to learn incrementally and interactively, and their internal representation and reasoning mechanisms are difficult to interpret. The architecture described in this thesis seeks to address these challenges by drawing inspiration from research in cognitive systems. It interleaves reasoning and learning, and combines the complementary strengths of non-monotonic logical reasoning with incomplete commonsense domain knowledge, deep learning, and decision-tree induction. As a motivating example, we consider a robot reasoning about the stability and occlusion of scene objects in images of particular scenes, and rearranging objects to reduce clutter. For any given scene, the architecture enables the robot to incrementally learn the grounding of spatial relations between objects in an image of the scene. The robot then attempts to complete the desired scene understanding task through non-monotonic logical reasoning with the extracted spatial relations and incomplete commonsense domain knowledge. When non-monotonic logical reasoning cannot complete the task, or provides an incorrect outcome, regions of interest relevant to the task at hand are automatically identified and used to train more targeted deep network models. Such regions and the corresponding labels (e.g., occlusion and stability labels) are also used to discover previously unknown domain axioms, which are merged with the existing knowledge and used for subsequent reasoning. Furthermore, the architecture enables the robot to explain its decisions and beliefs in the form of a description comprising relations between relevant objects, actions, and domain attributes. Experimental results using simulated images and images from a physical robot manipulating tabletop objects indicate the ability to: (i) improve the

reliability of decision making, and reduce the sample complexity and computational effort involved in training, in comparison with an architecture based just on deep networks; (ii) reliably acquire and merge new information about the domain in the form of constraints; and (iii) provide accurate relational descriptions as explanations in the presence of noisy sensing and actuation.

Contents

Contents **7**

List of Figures **10**

List of Tables **13**

1 Introduction **16**
- 1.1 Objectives and Contributions of Thesis 19
- 1.2 Thesis outline 20

2 Literature Review **22**
- 2.1 Visual Scene Understanding 22
- 2.2 Spatial Relations 26
- 2.3 Intuitive Physical Engines 28
- 2.4 Knowledge Representation and Reasoning 29
  - *2.4.1 First-Order Logic* 30
  - *2.4.2 Non-monotonic Logical Reasoning* 30
  - *2.4.3 Logic-based Knowledge Induction* 32
  - *2.4.4 Knowledge Representation and Reasoning with Deep learning* 33
- 2.5 Explainable Reasoning and Learning 35
- 2.6 Challenges in the literature 37

3 Learning the Ground of Spatial Relations between Objects **39**
- 3.1 Introduction 39
- 3.2 Architecture description 41

3.2.1 Domain Representation in ASP 42
3.2.2 Qualitative Spatial Representation 43
3.2.3 Metric Spatial Representation 44
3.2.4 Combined Model and Other Relations 47
3.3 Experimental Setup 48
3.4 Results and discussion 50
3.5 Conclusion 55

4 Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning **57**
4.1 Introduction 57
4.2 Proposed Architecture 58
4.2.1 Knowledge Representation with ASP 60
4.2.2 Attention Mechanism 62
4.2.3 Convolutional Neural Networks 64
4.2.4 Decision Tree Induction and Axiom Merging . . 66
4.3 Experimental setup 71
4.4 Results and discussion 73
4.5 Conclusion 82

5 Reasoning with Incomplete Commonsense Knowledge for Transparent Decision Making on Robots **83**
5.1 Introduction 83
5.2 Architecture 85
5.2.1 Interaction interface and control loop 87
5.2.2 Program Analyzer 88
5.2.3 Baxter platform 91
5.3 Experimental setup and results 91
5.3.1 Experimental setup 91
5.3.2 Execution Trace 93
5.3.3 Experimental results 100
5.4 Conclusions 104

6 Conclusion and Future Work **105**
6.1 Conclusion 105

6.2 Future work 107

Bibliography **109**

List of Figures

1.1 Outline of the thesis 21

3.1 (a) Illustrative image of scene with objects; and (b) segmented version with 3D point clouds of objects in different colors. . . . 40
3.2 Proposed architecture. 41
3.3 (a) Bounding box for point cloud of a particular object; and (b) Pyramids delimiting space around the bounding box. 44
3.4 (a) Example of 1D histogram grounding “not-touching”; (b) Example of 2D position histogram grounding “left”. 45
3.5 Examples of simulated scenes used in experiments. 53
3.6 (a) Image from TOSD dataset; (b) Histogram generated from the image using the smaller box as the reference object. 54
3.7 Histograms representing learned MSR groundings for: (a) *above*; and (b) *behind*. 54
3.8 (a) Image with one object *above* another; and (b) revised 2D histogram for *above*. 55

4.1 A simulated scene with toys. The robot has to reason about occlusion and stability of structures to reduce clutter. 58
4.2 Architecture combines the complementary strengths of non-monotonic logical reasoning, deep learning, and decision tree induction, in order to perform the scene understanding tasks reliably and efficiently. 59
4.3 Examples of ROIs highlighted by the attention mechanism for: (a) balance of structure; (b) occlusion of objects. 63
4.4 Lenet architecture. 64

4.5 Example of a decision tree constructed for stability estimation using some labeled examples. Highlighted branches are used to construct previously unknown axioms. 66
4.6 Example of a decision tree constructed for occlusion estimation using some labeled examples. Highlighted branch is used to construct previously unknown axiom. 67
4.7 Accuracy of Lenet and Alexnet with and without commonsense reasoning and the attention mechanism. The number of background images was fixed at 100. Our architecture improves accuracy in comparison with the baselines. 74
4.8 Examples of test images for Lenet and Lenet(Att): (a) both detected the occlusion of the red cube by the green mug, but only the latter correctly estimated the tower's instability; and (b) both predicted the instability of the tower, but only Lenet(Att) detected the obstruction of the green cube by the yellow can. . . . 75
4.9 Effect of changing the number of backgrounds on the accuracy of the Lenet and Lenet(Att) networks for 100 and 5000 training images. Without the attention mechanism and commonsense reasoning, increasing the number of backgrounds reduces the classification accuracy. 76
4.10 Accuracy with and without the attention mechanism and commonsense reasoning of: (a) Lenet; (b) Alexnet. The number of background images was fixed at 100. Any desired accuracy is achieved with a smaller training set when commonsense reasoning and attention mechanism are used. 77
4.11 Normal axioms evolution over time. 79
4.12 Default axioms evolution over time. 80
4.13 Illustrative image used for planning experiments with and without the learned axioms. 81

5.1 Motivating scenario of a manipulator arranging objects in desired configurations on a tabletop. 84
5.2 Camera's view extracted from Baxter's grippers: (a) left; (b) right. 84

5.3 Architecture combines strengths of deep learning, non-monotonic logical reasoning with incomplete knowledge, and inductive learning. New components to the right of the dotted line support desired explainability. 85
5.4 Highlighted relation between blue and red cubes, used in the explanation constructed in Execution Example 1. 94
5.5 Highlighted tennis ball to be considered in the explanation provided in Execution Example 3. 96
5.6 Example of scene created in simulator. 97
5.7 Highlighted relation between white and red cubes, used in the explanation constructed in Execution Example 4. 98
5.8 Highlighted duck to be considered in the explanation provided in Execution Example 6. 99
5.9 Percentage distribution of optimal, sub-optimal and incorrect plans created for **real scenes** when (a) no axioms are learned and (b) new axioms are learned. 101
5.10 Percentage distribution of optimal, sub-optimal and incorrect plans created for **simulated scenes** when (a) no axioms are learned and (b) new axioms are learned. 101

List of Tables

3.1 Map of commonly used spatial prepositions – extracted by Belz et al. (2015) from VOC'08 1K – in primitives. VOC'08 1K contains descriptions of 1,000 images of Pascal VOC 2008 Shared Task Competition collected by Rashtchian et al. (2010), using Mechanical Turk. 49

3.2 Comparison of (a) MSR grounding trained with just human feedback; (b) MSR grounding trained with 200 pairs labeled by the QSR grounding and seven pairs labeled with human feedback; and (c) the combination of MSR grounding, trained as in (b), and QSR-based grounding with the choice made by the control node. 51

3.3 Comparison of (a) QSR-based grounding; (b) MSR-based grounding from $\approx$ 2000 pairs labeled with QSR-based grounding (no human feedback); and (c) using the control node to combine MSR-based grounding, as trained in (b), and QSR-based grounding. 52

3.4 QSR-based grounding compared with MSR-based groundings obtained using different amounts of human feedback. 53

4.1 Precision and recall for previously unknown axioms (normal, default) using decision tree induction. 78

4.2 Normal axioms legend. 79

4.3 Default axioms legend. 81

5.1 (**Program Analyzer**) Construct answer to input question . . . 88

5.2 Values of planning measures with the learned axioms expressed as a fraction of the values of these measures without the learned axioms in each paired trial. 100
5.3 Precision and recall of retrieving relevant literals for constructing answers to explanatory questions, with and without the learned axioms used for reasoning. These results refer to **Real scenes** from the robot's gripper. 103
5.4 Precision and recall of retrieving relevant literals for constructing answers to questions, with and without reasoning with learned axioms. These results refer to **Simulated scenes**. 103

List of abbreviations

ASP Answer Set Prolog.

CNN Convolutional Neural Network.

DNN Deep Neural Networks.

FOL First-Order Logic.

GIS Geographic Information Systems.

IPE Intuitive Physical Engine.

KRR Knowledge Representation and Reasoning.

MSR Metric Spatial Representation.

PDDL Planning Domain Definition Language.

QSR Qualitative Spatial Representation.

RCC Region Connection Calculus.

RGB-D Red Green Blue - Depth.

RNN Recurrent Neural Network.

ROI Regions of Interest.

Chapter 1

Introduction

Agents [1] deployed to assist humans in complex environments need to reason with and incrementally update their understanding of their surroundings. The data acquired from sensors (e.g., images from a camera) are a good source of information, but they are often unreliable in complex domains. Reasoning with such uncertain and incomplete information may lead to incorrect or sub-optimal decisions. Learning from interactions with humans and the domain can partially offset these difficulties, enabling robots to use current observations to update their existing knowledge. Images of scenes obtained from the robot's cameras are a rich source of information about the domain, and understanding the content of these images and scenes is a key focus of this thesis. State-of-the-art methods for scene understanding are based on deep network architectures and related algorithms, which have achieved high accuracy in pattern recognition tasks and in many applications in robotics and AI. However, these networks require a large number of labeled training samples and substantial computational resources, which are not readily available in many practical domains. Also, it is difficult to use prior knowledge about the domain, understand the working of the networks, or to reuse a trained network in related domains. In contrast, there are algorithms that support efficient learning from a small number of samples acquired interactively and incrementally. Also, research in non-monotonic logical reasoning allows agents to efficiently reason with incomplete commonsense domain knowledge, elegantly revising previously held conclusions.

[1] We use the terms "Agent" and "Robot" interchangeably

This thesis explores the complementary strengths of these methods in the context of assistive robots performing visual scene understanding tasks.

Visual scene understanding tasks range from segmenting and recognizing individual objects in an image of a scene to exploring the relations and interactions between two or more objects in images of a scene. The latter might include the relative spatial locations of objects, and physical interactions between the objects. These relations and interactions provide information that is essential to answer basic questions about the scene such as: is a particular arrangement of objects stable? Which action can be successfully executed in any given scenario? What will happen if a particular action is performed? This thesis considers the estimation of spatial relations between objects (in images of scenes) as a key capability for robots performing scene understanding tasks. Spatial relations are often described using prepositional words such as $right$, *below*, $front$, and in that refer to particular objects in terms of other objects or events. To estimate spatial relations between objects, the robot will need a vocabulary of such prepositional words and a grounding of these words, e.g., a mapping of these words to 3D regions or distances from reference points or objects. Existing groundings usually rely on hand-encoded, fixed rules or on an offline, separate training phase to learn these groundings. However, the grounding has to be revised over time in dynamic domains to account for factors such as sensing errors and changes in viewpoint. Since humans may not have the time or expertise to provide comprehensive feedback, it would be useful for the robot to acquire and revise this grounding over time. One contribution of this thesis is an approach that initially estimates the spatial relations between objects using a fixed hand-coded qualitative grounding, and then incrementally learns a histogram-based quantitative grounding of these relations.

Spatial relations and other interactions between objects identified in an image of a scene represent valuable information for the comprehension of the domain. Despite that, deep networks designed for scene understanding tasks do not explicitly consider this information. Instead, they try to capture the desired aspect of a scene directly from its pixels, in a bottom-up manner. The generalization ability of such networks is largely dependent on the availability of a sufficiently large set of training data, which are not readily available in many practical domains, and these approaches are computationally expensive. Also, it is challenging to learn incrementally and interactively, to reason with incomplete information, or to interpret the internal representation and mechanisms they learn. Research in cognitive systems could be used to address many of these limitations. For instance, there is considerable research in non-monotonic logical reasoning with commonsense knowledge. These approaches allow an agent to elegantly revise previously held conclusions over time. There are also algorithms that support efficient learning from data obtained incrementally and interactively. These developments have not been fully explored for scene understanding so far. This research explores the complementary strengths of relational representation, non-monotonic reasoning with incomplete commonsense knowledge, data-driven learning, and incremental induction of knowledge to mitigate these limitations.

Besides improving the understanding of their surroundings, a deep understanding of scenes may aid agents in providing comprehensive explanations for their decisions and beliefs, which is a key requirement for effective collaboration between robots and humans. Despite the effort employed in recent years on explainable reasoning and learning, providing transparency in decision-making systems continues to be an open problem. This is especially challenging in the case of systems that include modern data-driven algorithms, e.g., those based on deep network models. In contrast, studies in cognitive systems indicate that relational representations and reasoning with commonsense domain knowledge help provide explainable descriptions for agent's decisions, beliefs and experiences. We explore the expressiveness of the distributed relational representation of knowledge to provide these explanations. Since our architecture learns such knowledge from the same data used for constructing deep networks, the relational descriptions provided as explanations may also be considered as descriptions of the observed behavior of these networks, providing insights about their internal operation.

1.1 Objectives and Contributions of Thesis

In summary, state of the art methods for visual scene understanding are based on deep (neural) networks and related algorithms, which require many labeled training examples and substantial computational resources. These requirements are difficult to satisfy in many practical domains. Also, the decisions of these networks are rather difficult to understand. To address these challenges, this thesis seeks to combine the complementary strengths of a relational representation, symbolic reasoning based on commonsense knowledge, and learning from experience. More specifically, the objective of this thesis is to address the following fundamental scientific question: *how best to combine commonsense symbolic reasoning and data-driven deep learning such that they inform and guide each other to achieve reliable and efficient visual scene understanding in robotics?* Towards answering this question and achieving the corresponding objective, this thesis describes an architecture for robots that makes the following contributions:

1. The development of an incrementally and interactively learned model based on 1D and 2D histograms that grounds (i.e., assigns meaning in the physical world to) prepositional words representing the metric spatial relations between scene objects. An approach is also provided to combine this learned grounding with a fixed grounding of qualitative spatial relations between scene objects.

2. The development of an approach that combines the complementary strengths of deep learning and non-monotonic logical reasoning with incomplete commonsense domain knowledge, resulting in reliable and computationally efficient visual scene understanding. Reasoning is also used to guide the incremental learning of previously unknown state constraints that are used for subsequent reasoning.

3. The development of an approach that enables a robot to provide on-demand explanations of its decisions, beliefs, and the outcomes of hypothetical events, in the form of descriptions comprising the relations between relevant objects, actions, and attributes of the domain. This

approach is also able to automatically identify these objects, actions and attributes.

The architecture's capabilities are evaluated in the context of planning tasks and scene understanding tasks in simulated scenes or on a physical robot manipulating tabletop objects. Specifically, the robot (a) computes and executes actions to arrange objects in desired configurations; and (b) estimates occlusion of objects and stability of object configurations in images of scenes. Experimental results indicate the ability to (i) significantly improve the accuracy and computational efficiency of estimation and planning in comparison with an architecture that only uses deep networks; (ii) incrementally reduce uncertainty about the scene by learning the grounding of spatial relations and previously unknown state constraints; and (iii) reliably and efficiently construct relational descriptions of decisions and beliefs by automatically identifying and reasoning with the relevant knowledge despite noisy sensing and actuation.

1.2 Thesis outline

The thesis is divided into six chapters. The current chapter provides an overview of research problem of interest, and summarizes the research objective and contributions of this thesis.

Chapter 2 reviews the related work on scene understanding, knowledge representation and reasoning, and explanation generation, highlighting the limitations and gaps in the existing research that this thesis seeks to address.

Chapter 3 describes the first contribution of this thesis, a basic architecture comprising non-monotonic logical inference and an approach that enables the robot to incrementally learn the grounding of prepositional words representing the spatial relations between pairs of objects (e.g., front, above, close) in simulated and real world scenes. This grounding combines a manually-encoded qualitative spatial representation of the relations between objects, and an incrementally learned metric spatial representation based on histograms. The grounding is used to estimate spatial relations and used this knowledge for subsequent reasoning.

Chapter 4 presents the second contribution, which combines the complementary strengths of non-monotonic logical reasoning and statistical learning for visual scene understanding tasks. Specifically, the explicit knowledge interactively learned and represented as declarative rules guides the efficient construction of convolutional neural networks for scenes understanding. Also, previously unknown constraints are automatically learned using decision tree induction and merged with the existing knowledge.

Chapter 5 describes the third contribution of this thesis, which enables the robot to provide on-demand explanations of its decisions and beliefs. These explanations are provided in the form of relational descriptions between relevant domain objects, actions and attributes that are identified automatically. The chapter also highlights the importance of the knowledge acquired over time in an agent's planning.

The connections between the elements described in Chapters 3 to 5 are illustrated in Figure 1.1. The relational representation and non-monotonic logical reasoning component plays the central role in the proposed architecture. Although this module is not represented inside any particular chapter box in the figure, it is described in each chapter. The relations between modules are described in details in the relevant chapters.

Chapter 6 draws the conclusion, presents the lessons learned from this research, and proposes some possible directions for future works.

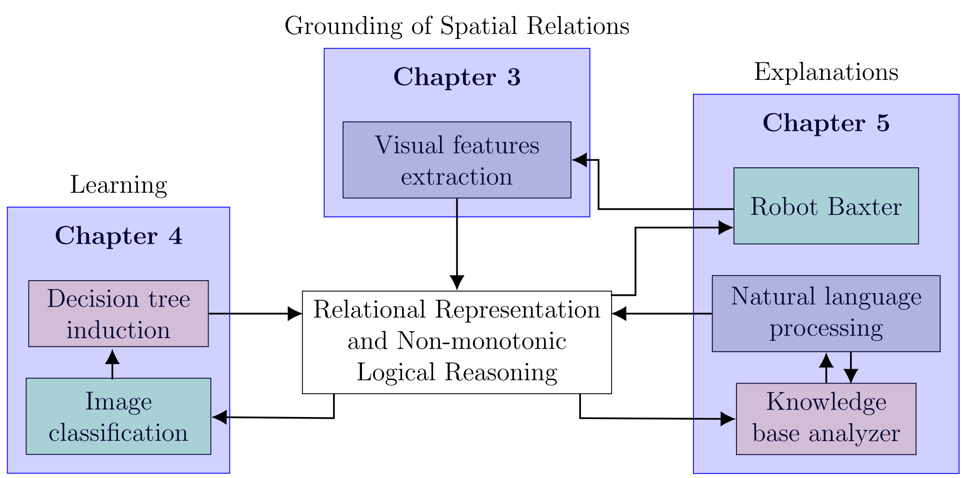


Figure 1.1: Outline of the thesis

Chapter 2

Literature Review

In this research, we attempted to address some of the limitations of modern data-driven algorithms by drawing inspiration from cognitive systems. We explored visual scene understanding tasks (Section 2.1) as a particular case to test the proposed architecture. In this context, the study of spatial relations between objects has played a relevant role for additional understanding of a scene (Section 2.2). Likewise, works exploring the physical mechanics governing interaction between objects in a scene has provided further insights into an image (Sections 2.1 and 2.3). These approaches present a number of limitations, such as the inability to deal with incomplete information. We explore the use of knowledge representation and reasoning (KRR) techniques (Section 2.4) to mitigate these limitations. Finally, we discuss how the relational representation of knowledge aids in providing explanations machines can provide to increase trust (Section 2.5), and summarize the challenges identified in the literature (Section 2.6).

2.1 Visual Scene Understanding

Computer vision focuses on building computational models to analyze and understand images and videos. A video contains a sequence of images, which are composed of pixels. A subset of pixels grouped together may represent one component of a scene, e.g., objects, animals, or backgrounds. The segmentation of an image in its components and the detection of an object of interest have been widely studied. Some approaches use features descriptors, such as scale-invariant feature transform (SIFT) (Lowe 1999), speeded up robust features (SURF) (Bay et al. 2008) and histogram objects gradient (HOG), to detect objects in a scene (Dalal and Triggs 2005; Felzenszwalb et al. 2010; Jégou and Zisserman 2014; Shi, Avrithis, and Jégou 2015; J. Li and Yimin Zhang 2013). Such descriptors consider the similarities between blocks of pixels, such as color and texture, to determine if they belong to the same object or not. Other works add the depth from 3D point clouds to segment RGB-D images (Potapova et al. 2014; Kootstra, Bergström, and Kragic 2010). More recent approaches apply deep learning to detect and localize objects in a scene (Sermanet et al. 2013; Redmon et al. 2016; Y. Chen et al. 2020).

The segmented image can be analyzed in terms of its components. The identification or classification of these components plays an important role in computer vision. Even when individual classes cannot be identified, the ability to perform accurate broad classification can be useful. For instance, the classification of an object as toxic indicates that it must be kept far from children even if the specific type of toxic material has not been recognized. Approaches to accomplish these tasks are also currently based on Convolutional Neural Network (CNN) (Wang et al. 2016; V. K. Singh et al. 2020), some of which have reached results close to the human performance.

A CNN is a type of deep network designed to process images. CNNs have many parameters based on size, number of layers, and activation functions, but the basic building blocks are convolutional, pooling, and fully connected layers. The convolutional and pooling layers are used in the initial or intermediate stages of the network, whereas the fully connected layer is typically one of the final layers. In a convolutional layer, a filter (or kernel) is convolved with the original input or the output of the previous layer. One or more convolutional layers are usually followed by one pooling layer. Common pooling strategies such as max-pooling and average-pooling are used to reduce the dimensions of the input data, limit the number of parameters, and control overfitting. The fully connected layers are equivalent to feedforward neural networks in which all neurons between adjacent layers are connected; they often provide the target outputs. In the context of images, convolutional layers extract useful attributes to model the mapping from inputs to outputs, e.g., the initial layers may extract lines and arcs, whereas the subsequent layers may compose complex shapes such as squares and circles.

Deep learning, in particular CNN, has been successfully applied to computer vision problems in recent years. It is enabling machines to perform some vision tasks even better than humans. For example, He et al. (2015) developed a CNN that surpassed human performance on the Imagenet 2012 dataset classification task.

Encouraged by the remarkable success CNN has achieved in computer vision, recent works have applied it in scene evaluation involving high-level physical reasoning. As an example, some studies have employed deep neural networks to predict the stability of a tower of blocks (W. Li, Fritz, and Leonardis 2016; Lerer, Gross, and Fergus 2016). Due to the lack of labeled data to train CNN in such tasks, they often rely on Intuitive Physical Engine (IPE) to create simulated images along with correspondent ground truth. As another example, Wu et al. (2015) trained a deep convolutional network to predict the movement of an object sliding down an inclined surface and possibly colliding with another object. Other works employ CNN to predict the movement of objects caused by the application of external forces (Mottaghi et al. 2016; Fragkiadaki et al. 2015), or the trajectory of an object after bouncing against a surface (Purushwalkam et al. 2019).

Despite the recent success of CNN and other deep network architectures in computer vision tasks, these approaches have some important limitations: (a) they usually require a large amount of training data; (b) they struggle to generalize for samples different from the training set; (c) they are computationally expensive; (d) it is rather difficult to use them for incremental and interactive learning; and (e) their internal representations and the mechanisms they learn are difficult to interpret.

In order to mitigate some of the above listed limitations, researchers have used prior (domain) knowledge during training (Sünderhauf et al. 2018). For instance, a Recurrent Neural Network (RNN) architecture augmented by arithmetic and logical operations has been used to answer questions using information from semi-structured tables (Neelakantan, Le, and Sutskever 2015). This work used textual information instead of the more informative visual data, and did not support reasoning with commonsense knowledge. Another example is the use of prior knowledge to encode state constraints in the CNN loss function; this reduces the effort in labeling training images, but it requires the constraints to be encoded manually as loss functions for each task (Stewart and Ermon 2017). Similarly, the model proposed by Yang, Ishay, and Lee (2020) uses the outcome of deep networks as input for non-monotonic reasoning, and back-propagates the inferences made by the reasoner to update the parameters of these networks. This approach also reduces the amount of labeled training samples, but requires the prior knowledge to be manually encoded. The structure of deep networks has also been used to constrain learning, e.g., by using relational frameworks for visual question answering (VQA) that consider pairs of objects and related questions to learn the relations between objects (Santoro et al. 2017; Cadene et al. 2019). This approach, however, only makes limited use of the available knowledge, and does not revise the constraints over time. Lutter, Ritter, and Peters (2019) impose Lagrangian mechanics as prior knowledge to a deep network. This reduced the training effort and improved generalization for controlling a robot's trajectories, but they explored sensor data, whereas we are interested in the more informative images as input.

The understanding of a scene can be improved by exploring the information related to the interaction among its components. The spatial relation between objects as well as their properties may reveal which actions are afforded in different scenarios. For instance, imagining an apple surrounded by a number of other fruits. In this situation, the relationship between apple and other fruits indicates that it could not be grabbed without the previous execution of another action. Researchers have applied spatial relations in different tasks, such as scenes description (Belz et al. 2015), objects localization (Wong, Kaelbling, and Lozano-Perez 2013), and recognition of actions in videos (Ziaeetabar et al. 2017). In contrast, approaches applying CNNs or simulators have provided physical scenes explanation, such as the stability of a tower of blocks, without explicit use of spatial relations. The former presents a number of limitations as described above. Simulators, known as Intuitive Physical Engine (IPE), also have important limitations (detailed in Section 2.3). We expect the employment of spatial relations between objects can aid in mitigating these limitations.

2.2 Spatial Relations

Our first step towards a deeper scene understanding is grounding spatial relations between objects, which are often described by prepositions. The concept of words such as *below*, *on*, *in*, and *behind*, are largely used by humans in different contexts: describing the location of objects; playing with blocks; or planning the actions required for grasping a mug occluded by a pile of bowls. Studies in cognitive neuroscience reflect the importance of spatial relations for humans. For instance, Laeng, Chabris, and S. M. Kosslyn (2004) mapped the left-hemisphere parietal lobe of the brain as responsible for producing representations describing such relations. According to the recognition-by-component model for object recognition, the human brain may describe objects by their parts and the spatial relation among them (Cave and S. M. Kosslyn 1993; Laeng, Shah, and S. Kosslyn 1999). In the AI front, the international workshop on Spatial Language Understanding (SpLU-2018/2019) has focused on how to express spatial relationships between objects in natural language (Dobnik, Ghanimifard, and Kelleher 2018; Ulinski, Coyne, and Hirschberg 2019). However, it is difficult to equip agents with a comprehensive knowledge of these prepositions.

Approaches found in the related literature for grounding and interpreting the spatial relations between objects are broadly based on the use of manually encoded rules, or the use of training or learning algorithms. When rules are manually encoded, the construction of a spatial vocabulary is often based on *Qualitative Spatial Representation (QSR)* (Moratz, Nebel, and Freksa 2003; Ye and Hua 2013; Zampogiannis et al. 2015; Elliott and Vries 2015). These approaches may not provide accurate estimates of the spatial relations, as they often approximate objects as points or establish rigid boundaries between spatial relations. Moreover, the spatial relations are encoded in advance, whereas the interpretations of these relations are likely to change over time in robotics domains.

As described by J. Chen et al. (2015), qualitative formalisms for representing spatial relations traditionally consider topological or positional aspects. Topological approaches, such as Region Connection Calculus (RCC) (Anthony G Cohn et al. 1997; Renz 2002), are well suited for Geographic Information Systems (GIS), being also applied in robotic tasks (Landsiedel et al. 2017) such as navigation. In contrast, since positional (or projective) relations are more accurate (Clementini 2019), they can better describe the spatial configuration of objects in indoor environments. For this reason, we focus on positional representations of spatial relations in this thesis.

Approaches that seek to train or learn the spatial relations or their grounding do so based on *Metric Spatial Representation (MSR)*, i.e., a set of measures such as angles and distances between objects. Algorithms based on MSR have been used in different applications in recent years. For example, an approach based on MSR has been developed to predict the success of a robot's action in a previously unseen scenario (Fichtl et al. 2015), while another approach enabled an agent to learn relations between objects and generalize them to new objects (Mees et al. 2017). Other work has focused on developing a system capable of choosing appropriate prepositions to describe an image (Belz et al. 2015). Gatsoulis et al. (2016) designed a software to extract QSR relations from videos. In the context of human-robot interaction, a system has been developed for executing a set of actions on objects and answering queries about spatial positions (Guadarrama et al. 2013), QSR and MSR have been compared for scene understanding (Thippur et al. 2015), and MSR and a kd-tree have been used to dynamically infer spatial relations between objects (Ziaeetabar et al. 2017). However, most of these approaches learn the representation of spatial relations offline or in a separate training phase. In contrast, we propose an approach that initially applies a hand-designed generic grounding, and incrementally and interactively learns a histogram-based specialized grounding from new experiences and feedback.

Different types of neural (or deep) network architectures have also been used in recent years to infer objects and their spatial relationships from images (Jund et al. 2018) and natural language expressions for tasks such as manipulation (Paul et al. 2018), navigation (Pronobis and Rao 2017), and human-robot interaction (Shridhar and Hsu 2017). These approaches require a large number of training examples of images and natural language expressions, learn the grounding offline, and are computationally expensive. The work proposed in this thesis seek to combine the complementary strengths of different groundings of the spatial relations (QSR and MSR) and to incrementally learn these groundings.

One way of addressing the unavailability of benchmark dataset with labeled spatial relations between objects is to use a Physical Engine (Bullet physics library [1]) to generate simulated scenes with labeled objects and relations. We next look at approaches that use such simulated data.

2.3 Intuitive Physical Engines

During early childhood, humans start to construct an internal model of intuitive physics (Baillargeon 2002). Such a model could be seen as a simplified version of Newtonian mechanics, which allows us to quickly draw conclusions regarding the dynamics of objects. For instance, humans playing football do not need to solve parabolic trajectory equations. Instead, they would use a simplified physical model to estimate the movements of a ball which has been kicked.

Inspired by the human approach, researchers have applied game simulators to provide machines with further comprehension of a scene, e.g., the stability of a tower of blocks. Also called Intuitive Physical Engine (IPE), they present an incomplete, oversimplified, probabilistic and approximate version of the interactions governing the physical world. Although the simplifications allow their use in real time applications, simulators are still computationally expensive, especially for complex domains.

As an example, Battaglia, Hamrick, and Tenenbaum (2013) applied an IPE to predict if a tower of blocks would fall, and in which direction. Their results indicate a high correlation between the proposed model and human answers, suggesting that it was able to capture human physical intuition. Bates et al. (2015) used probabilistic simulation to predict liquid dynamics, and Sanborn, Mansinghka, and Griffiths (2013) studied the dynamics of

[1] http://bulletphysics.org

colliding objects. Other works use IPE only for generating training data for deep learning approaches, e.g., W. Li, Fritz, and Leonardis (2016), Lerer, Gross, and Fergus (2016), and Wu et al. (2015).

The use of IPE in robots operating in complex dynamic domains presents some limitations: each cycle of simulation is computationally expensive; the ability to reason with incomplete knowledge is limited; and it is difficulty to learn from previous experiences.

Current studies in understanding high-level physical interactions from scenes often use CNN or IPE. A number of studies indicate the advantages and drawbacks of both approaches. For example, R. Zhang et al. (2016) compared the two methods in the stability prediction of a tower of blocks. They concluded that although both achieve high accuracy, only the IPE can generalize to unseen scenarios. In contrast, E. Davis and Marcus (2016) listed a number of limitations of simulators in the context of AI tasks involving high-level physical reasoning, such as the requirement of reasonably high-quality domain theory and the availability of complete information. Other works mention additional IPE limitations, e.g., that the role of the agent's previous experiences is unclear (Ullman et al. 2014), and the effects of the strong priors encoded in the simulators (Lerer, Gross, and Fergus 2016).

Some of the limitations faced by CNN and IPE to deal with high-level physics understanding of a scene may be mitigated by applying a suitable Knowledge Representation and Reasoning (KRR) paradigm. To do so, these paradigms should be able to represent existing knowledge, as well as incrementally update such knowledge by exploring the information freely available in the domain. These requirements improve reasoning and simplify further learning. Some relevant KRR techniques are discussed below.

2.4 Knowledge Representation and Reasoning

Knowledge Representation and Reasoning (KRR) is devoted to representing information about the world and reasoning with these representations for different tasks. There is a large history of research in AI that has led to the development of different paradigms for KRR. For instance, classical first-order logic has been extensively used in different applications, but it does not support key commonsense reasoning capabilities. There is also considerable research in logic-based knowledge induction and its use in learning from experience. Furthermore, statistical learning algorithms and deep architectures have also been used to represent and reason with knowledge.

2.4.1 First-Order Logic

First-Order Logic (FOL) is a symbolic representation that combines logical and nonlogical constructs using the same rules as Boolean algebra.

The logical constructs include *punctuation*, *connectives* ("$\neg$" = classical negation; " $\wedge$ " = logical disjunction "and"; " $\vee$ " = logical conjunction "or"; "$\exists$" = "there exist ..."; "$\forall$" = "For all ..."; " = " is the logical equality; " $\implies$ " = implication; " $\iff$ " = biconditional), and *variables*. The nonlogical symbols include *functions* and *predicates*, which are described in terms of their arguments. Logical symbols have fixed meaning in the language, whereas the meaning of nonlogical symbols depends on the application.

A *Term* in FOL is recursively defined as any *variable* or any *function* $f(t_1, ..., tn)$ in which each t_i argument is also a *term*; *formulas* are defined as *predicates* of *terms*, and its combination using *connectives*. The knowledge in FOL is represented by *sentences*, which correspond to *formulas* without free *variables*. Each such *sentence* is either *true* or *false*, which makes it difficult to reason with incomplete knowledge. Languages have been developed based on FOL for reasoning about dynamic domains, e.g., Planning Domain Definition Language (PDDL) (McDermott et al. 1998), which has been used in robotic applications (Cashmore et al. 2015; Kootbally et al. 2015; Krueger et al. 2019).

Classical FOL is unable to reduce the amount of inferred consequences as a result of the addition of new information; it is thus not suitable for reasoning with commonsense knowledge.

2.4.2 Non-monotonic Logical Reasoning

There is a rich history of research in nonmonotonic logical reasoning; early work include a logical system called circumscription (McCarthy 1980), and an extension of classical logic that allows defeasible rules (Raymond Reiter 1980). Here, we focus on Answer Set Prolog (ASP), a declarative language used in the proposed architecture. ASP can represent recursive definitions, defaults, causal relations, special forms of self-reference, and language constructs that occur frequently in non-mathematical domains and which are difficult to express in classical logic formalisms. ASP is based on the stable model (answer set) semantics of logic programs (Gelfond and Kahl 2014).

Many theories of action and change have been developed, producing a number of action languages, and used in a variety of applications (W. Chen, Swift, and Warren 1995; Tu et al. 2011). Action languages are often used in robotics and AI to describe the effects of actions in the states of a system over time. They are formal models of parts of natural language used for specifying state transition systems. In this thesis we use the action language $\mathcal{AL}$ to describe the domain and to translate this domain representation to an ASP program for non-monotonic logical inference.

Action Language $\mathcal{AL}$

The action language $\mathcal{AL}$ used in this thesis has sorted signature comprising *statics*, *fluents* and *actions*. Statics are domain attributes whose values do not change over time, whereas fluents can be changed. *Inertial* fluents can be directly modified by actions and obey the laws of inertia, whereas *defined* fluents do not. A domain literal is a domain attribute p or its negation $\neg p$. $\mathcal{AL}$ allows three types of statements:

$$\begin{array}{rlll} l & \textbf{if} & p_0, \ldots, p_m & \text{State Constraints} \\ a\ \textbf{causes}\ \ l_{in} & \textbf{if} & p_0, \ldots, p_m & \text{Causal Laws} \\ \textbf{impossible}\ \ a_0, \ldots, a_k & \textbf{if} & p_0, \ldots, p_m & \text{Executability Conditions} \end{array}$$

where a is an action, l is a literal, l_{in} is an inertial literal, and $p_0, \ldots, p_m$ are domain literals.

We show throughout this thesis some examples of system description in $\mathcal{AL}$ and the correspondent translation to ASP program. Please see Gelfond and Kahl (2014) for more details and examples of these kinds of translation.

Answer Set Prolog

In ASP-based representation of knowledge, the status *unknown* is added, besides the traditional *true* and *false*, meaning that the agent does not have to believe anything that it is not forced to believe. Also, unlike classical first-order logic, ASP supports non-monotonic logical reasoning, i.e., adding a statement can reduce the set of inferred consequences, aiding in the recovery from errors due to the incomplete knowledge. ASP can also draw conclusions due to lack of evidence to the contrary, using concepts such as *default negation* (negation by failure) and *epistemic disjunction*. For instance, unlike "`¬a`", which implies that "*a is believed to be false*", "`not a`" only implies that "*a is not believed to be true*"; and "`a or ¬a`" is not tautological.

The answer sets obtained by solving an ASP program represent the beliefs of an agent associated with the program. It is possible to reduce inference, planning and diagnostics to compute answer sets. Modern ASP solvers support efficient reasoning in large knowledge bases, and architectures based on ASP have been used in different applications (Erdem and Patoglu 2012) by an international community, and also gaining industry attention (Erdem, Gelfond, and Leone 2016; Falkner et al. 2018; Gençay, Schüller, and Erdem 2019).

Non-monotonic logical reasoning enable agents to elegantly recover from errors caused by dealing with incomplete knowledge. This is a key requirement for agents deployed in complex environments. However, reasoning with incomplete information may lead to sub-optimal or incorrect decisions. This problem can be partially offset by using the information freely available in the domain for incrementally inducting previously unknown knowledge.

2.4.3 Logic-based Knowledge Induction

One of the objectives of this research is to support learning and revision of knowledge from experience. To this end, we employ logic-based knowledge induction, which deals with the learning and revision of programs created from incomplete specifications.

The induction of rules is well established in AI research. For instance, Gil (1994) focused on augmenting the knowledge base encoded in first-order logic representation by considering the unexpected observations as the potential outcomes of the actions executed. Such approaches have the known limitations of FOL, e.g., inability to support non-monotonic logical reasoning. Other approaches based on non-monotonic logic support the revision of theory of action and change (Balduccini 2007; Certicky 2012) but they are computationally expensive and do not support interactive learning. One recent promising approach is that of Sridharan and Meadows (2017), in which any unexpected and unexplained observation triggers interactive learning to discover new axioms and revise existing ones. In addition, Law, Russo, and Broda (2018) have explored the generality aspect of inducted ASP-based representations.

Interactive task learning is a general framework for acquiring domain knowledge using labeled examples or reinforcement signals obtained from domain observations, demonstrations, or human instructions (Chai et al. 2018; John E. Laird et al. 2017; Kirk and John E Laird 2019). It can be viewed as building on early work on searching through the space of hypotheses and observations (Simon and Lea 1974), but such methods have rarely been explored for scene understanding. In this research, we are interested in learning in the context of deep neural architectures applied to visual scene understanding tasks. We present below some approaches that explore knowledge representation and reasoning combined with deep architecture models.

2.4.4 Knowledge Representation and Reasoning with Deep learning

Deep Neural Networks (DNN) have also been used to represent and reason with knowledge extracted directly from data. Sukhbaatar, Weston, Fergus, et al. (2015) proposed the *Memory Networks*, which apply Recurrent Neural Network (RNN) to deal with the question answering (QA) task based on previously informed description of facts. Similarly, Graves, Wayne, and Danihelka (2014) have employed Long Short-Term Memory (LSTM) — an improvement on RNNs to deal with the "vanishing and exploding gradient" problem — to learn simple algorithms applied to a data stream. Both approaches address the problem of learning from a data sequence, but they still suffer from the known limitations of DNN, e.g., high sample and computational complexity.

The difficulty in learning even basic arithmetic and logic operations makes it challenging to use deep networks for knowledge representation and reasoning. For instance, Joulin and Mikolov (2015) demonstrate that RNNs fail in adding two binary numbers. To overcome this challenge, Neelakantan, Le, and Sutskever (2015) created the *Neural Programmer*: an RNN augmented by a set of arithmetic and logic operations able to answer questions using table information, and a program inducted from training examples. This work is close in spirit to our proposed research since the knowledge extracted from experience induces a program for future reasoning. However, their program is implicitly represented by RNN parameters, whilst we propose to use a logic-based formalism. Also, they deal with textual information, in which RNNs have been successfully applied, whereas we are concerned with more complex visual information.

Studies in neural-symbolic learning and reasoning have highlighted the advantages and limitations of combining statistical learning with logical reasoning (Besold et al. 2017; A. d. Garcez et al. 2019). According to these studies, a purely symbolic approach would lack the parallelism and kinds of adaptive learning usually seen in machine learning techniques (Valiant 2008), whereas machine learning algorithms lack the expressiveness required for interpretation and validation of their internal representations (Smith and S. Kosslyn 2006). These methods typically rely on FOL (Guillame-Bert, Broda, and A. d. Garcez 2010; Evans and Grefenstette 2018), which suffers from the already described limitations, or simplify the neural architectures corresponding to specific symbolic representations (A. S. d. Garcez, Lamb, and Gabbay 2007; Penning et al. 2011). They also rarely support the automatic detection and correction of errors in the learned knowledge.

The widespread use of reasoning agents to assist humans depends on their ability to explain decisions. The implicit representation of knowledge learned by deep architectures is a significant challenge for such comprehension. Modern DNNs base their decisions on millions of parameters, which makes it difficult to understand why a particular choice was made. For this reason, studies have been conducted to improve the explainability of deep networks. For instance, Hendricks et al. (2016) developed an approach to classify different species of birds and associate the image description with class definition. It searches for parts of the image that significantly influence its classification performance, e.g., it determines the more prominent characteristics of each specie. This approach may shed some light on the network choices in the specific task of classifying birds, but it is difficult to be applied to other domains or tasks. We seek to design a model that provides explanatory descriptions for the agent's decision and beliefs, which can be transferred across multiple domains and tasks. Such descriptions are in the form of relations between objects, actions, and domain attributes, and are obtained from a relational description of knowledge.

2.5 Explainable Reasoning and Learning

The interaction between humans and robots is highly dependent on trust. Agents are more likely to be trusted if they can explain their reasoning systems. Besides improving trust in robots, explanations are valuable in aiding debugging for the system designer (Southwick 1991).

Early work on explanation generation was based on research in cognition, psychology, and linguistics (Friedman 1974; Grice 1975). For instance, Friedman (1974) presented a theory of explanation in terms of generality, objectivity, and connectivity, and Grice (1975) characterized cooperative response as being valid, informative, relevant, and unambiguous. Fundamental computational models were also developed for generation of explanations (Kleer, Mackworth, and R. Reiter 1992; Raymond Reiter 1987). Subsequent studies of human subjects have indicated the importance of coherence, simplicity, and generality (Read and Marcus-Newhall 1993), soundness and completeness (Kulesza et al. 2013), and other attributes of good explanations (Grice 1975). These findings have inspired computational models for explanations (Cawsey 1993; Moore and Paris 1993).

With the increasing use of AI and machine learning methods in different domains, there is much interest in academia, industry, and the military in understanding the decisions made within these methods as a means to establish accountability. Research indicates that this understanding aids in making the use of machines more acceptable to humans (Bonnie 1996; Lewandowsky, Mundy, and Tan 2000), and in enabling humans to trust the operation of agent-based reasoning or learning systems (Lewis 1998; Sycara et al. 1998).

Recent work in *explainable AI* and *explainable planning* can be broadly categorized into two groups (T. Miller 2019). Methods in one group modify or map learned models or reasoning systems to make their decisions more interpretable. Examples include explaining the predictions of any learned classifier by learning equivalent models that are interpretable (Koh and Liang 2017; Ribeiro, S. Singh, and Guestrin 2016), or biasing a planning system towards making decisions that are easier for humans to understand (Yu Zhang et al. 2017). Methods in the other group provide descriptive explanations of the decisions of reasoning systems in an attempt to make the decisions more transparent. Examples include methods that explain goal/plan changes (Dannenhauer et al. 2018), alternative plans (Borgo, Cashmore, and Magazzeni 2018), or causal and temporal relations (Seegebarth et al. 2012). Much of this research is agnostic to how an explanation is structured and presented (Borgo, Cashmore, and Magazzeni 2018; Chakraborti et al. 2017), or assumes complete domain knowledge (Chakraborti et al. 2017). Since the state of the art techniques for many pattern recognition and decision making problems in robotics, computer vision, and AI are based on deep (neural) networks and related algorithms, there has also been a lot of work recently on interpreting the representation and reasoning mechanisms of these networks and algorithms (Hendricks et al. 2016; Samek, Wiegand, and Müller 2017; Liu et al. 2019). However, understanding the representation and reasoning behavior of deep networks continues to be an open problem.

In this thesis, we consider robots as integrated systems that represent, reason with, and learn from incomplete domain knowledge and noisy observations. We are interested in the ability to answer explanatory questions about the decisions, and the underlying beliefs and choices. We are also interested in answering questions about hypothetical or counterfactual situations (T. Miller 2019); humans often ask about events that did not occur or options that were not pursued to infer causal relations (David and Tom

2000). Such answers are based on the distributed relational representation of knowledge, and are presented in the form of relations between objects, actions and domain attributes. Recent surveys state that these important capabilities are not supported by existing systems (Anjomshoae et al. 2019; T. Miller 2019).

2.6 Challenges in the literature

Based on the literature review presented in this chapter, the specific gaps we attempt to address are:

1. The spatial relations between objects play a key role in scene understanding tasks, but existing approaches for grounding spatial relations either rely on human input or require a separate offline training phase. These groundings thus can not be updated based on new experiences.

2. Existing approaches for scene understanding that use domain knowledge predominantly require such knowledge to be provided a priori. The requirement limits the ability to learn from new experience. These approaches are computationally expensive, require many training examples, and do not support the key desired capability for incremental and interactive learning.

3. The complex internal representation and mechanisms deep architectures learn make it difficult to understand their decisions; this consequently limits the trust humans place in such networks.

In the next chapters we try to address the gaps identified above. Specifically, each chapter will tackle one of the gaps presented above as follows:

- **Chapter 3** presents an incrementally and interactively learned model that grounds prepositional words representing the metric spatial relations between scene objects. This learned grounding is combined with a manually encoded grounding to address the gap 1.

- **Chapter 4** combines the complementary strengths of non-monotonic logical reasoning with incomplete commonsense domain knowledge, data-driven models, inductive learning to bridge the gap 2.
- **Chapter 5** explores the relational representation of knowledge as a key for enabling agents to provide on-demand explanations of its decisions, beliefs, and the outcomes of hypothetical events for mitigating the gap 3.

Chapter 3

Learning the Ground of Spatial Relations between Objects

As discussed in Section 2.2, the understanding of scenes requires the correct interpretation of the interactions among objects, including their spatial relations. As a result, robots have to reason with incomplete knowledge of domain objects and their relations in order to assist humans in complex domains. This chapter focuses on reasoning with spatial relations between objects, and on incrementally acquiring the *grounding* (i.e., meaning in the physical world) of words that describe these relations.

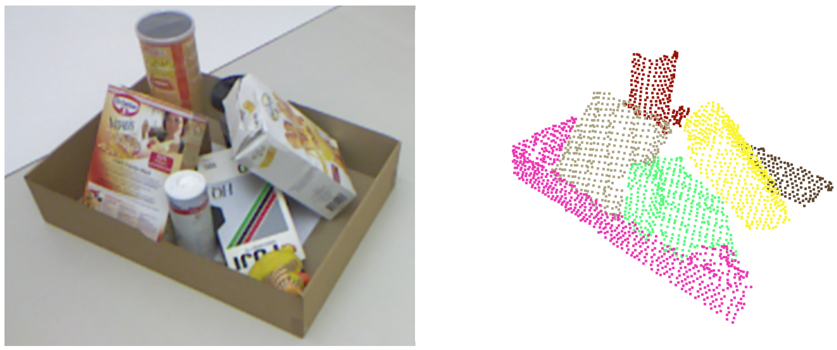
3.1 Introduction

The components of the architecture described in this chapter seek to provide robots with the grounding of spatial relations between scene objects, and have the following characteristics:

- Spatial relations between objects initially based on a generic grounding of prepositions in the 3D regions around objects are included in ASP, representing the incomplete domain knowledge.

- Both non-monotonic logical inference with the existing knowledge and human input (when available) are used to infer spatial relations between point clouds in new scenes, incrementally learning a specialized, histogram-based grounding of prepositions.

- Human input (when available) is also used to incrementally compute the relative accuracy of spatial relations inferred by the generic and specialized groundings, using the more reliable grounding for subsequent scenes.



(a) (b)

Figure 3.1: (a) Illustrative image of scene with objects; and (b) segmented version with 3D point clouds of objects in different colors.

The input comprises 3D point clouds of objects in a scene, e.g., Figure 3.1b, and a generic grounding of prepositions for seven position-based and three distance-based relations. Learning corresponds to the incremental acquisition and revision of histograms as specialized grounding of these relations. We do not explicitly represent the uncertainty in processing visual input; any conclusion drawn with high probability is elevated to a logic statement with complete certainty. The designed architecture enables robots to (a) infer spatial relations using a generic, manually-encoded grounding; (b) incrementally acquire a specialized grounding of spatial relations from a small number of examples; and (c) determine the relative confidence in each grounding and use the more reliable grounding for subsequent inference. These capabilities are evaluated on a benchmark dataset of tabletop objects and simulated scenes of furniture. This chapter is based on a conference paper published by Mota and Sridharan (2018).

3.2 Architecture description

The architecture focuses on seven position-based prepositions (*in, above, below, front, behind, right, left*) and three distance-based prepositions (*touching, not-touching, far*). These prepositions are used to encode spatial relations between specific scene objects as logic statements in Answer Set Prolog (ASP). The Qualitative Spatial Representation (QSR) module provides an initial, manually-encoded, generic grounding of spatial relations, which is used to extract spatial relations between pairs of 3D point clouds of each input scene. Human feedback, when available, is also used to label the spatial relations between any pair of point clouds in a scene. Both the QSR-based output and human feedback are transmitted by the control node to the Metric Spatial Representation (MSR) module, which incrementally acquires and revises the MSR-based grounding of prepositions in the form of histograms. The control node computes the relative trust in the QSR and MSR groundings based on the human feedback, which is assumed to be accurate. The more reliable grounding is used to extract logic statements representing spatial relations between scene objects in subsequent images; these are added to the ASP program. The key components of the architecture are depicted in Figure 3.2 and described below.

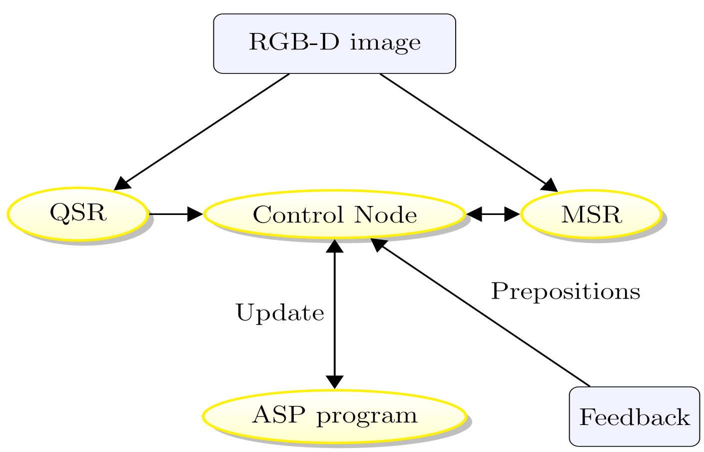


Figure 3.2: Proposed architecture.

The architecture includes other modules, e.g., the 3D point cloud of a scene is sub-sampled and the Euclidean cluster extraction segmentation algorithm (Rusu 2010)[1] is used to segment the point cloud into objects.

3.2.1 Domain Representation in ASP

As explained in Section 2.4.2, the domain is initially described in the action language $\mathcal{AL}$, and then translated to an ASP program for non-monotonic logical inference.

The domain description in $\mathcal{AL}$ has a *sorted signature* Σ and axioms. Σ includes *sorts* such as *object*, *location*, *color*, *shape*, and *step* (for temporal reasoning); *statics*, i.e., domain attributes that do not change over time; and *fluents*, i.e., domain attributes whose values can be changed. In our case, the spatial relations are fluents such as:

$$obj\_relation(relation,\ object,\ object). \tag{3.1}$$

which are described in terms of their arguments' sorts. The argument $relation$ represents a spatial relation, such as in, *above*, $touching$, $left$. We choose the second object of each such relation as the reference object. The domain axioms encode some rules to infer relations based on the spatial relations whose grounding is acquired:

$$\begin{aligned} obj\_relation(above, A, B) \ \textbf{if} \ & obj\_relation(below, B, A). \\ obj\_relation(under, A, B) \ \textbf{if} \ & obj\_relation(touch, A, B), \\ & obj\_relation(below, A, B). \end{aligned} \tag{3.2}$$

where the two axioms describe state constraints, and the second axiom says that any object A that is *below* object B and touching it is considered to be *under* it.

To reason with incomplete domain knowledge we construct the ASP program (Π), in which the statements of 3.2 are translated to:

[1] Available at *www.pointclouds.org* for download.

$$
\begin{aligned}
holds(obj\_relation(above, A, B), I) \leftarrow & holds(obj\_relation(below, B, A), I). \\
holds(obj\_relation(under, A, B), I) \leftarrow & holds(obj\_relation(touch, A, B), I), \\
& holds(obj\_relation(below, A, B), I).
\end{aligned}
\tag{3.3}
$$

where the predicate $holds(fluent, step)$ implies that a particular fluent holds true at a particular timestep. When action effects are to be modeled, the signature and axioms include *actions* with their preconditions and effects; a *history* of observations and executed actions is also considered. The ground literals in an *answer set* obtained by solving Π represent beliefs of an agent associated with Π. All reasoning (e.g., planning and inference) can be reduced to computing answer sets of Π (Gelfond and Kahl 2014). We use the SPARC system (Balai, Gelfond, and Yuanlin Zhang 2013) to compute answer set(s) of ASP programs.

3.2.2 Qualitative Spatial Representation

We implement a QSR model similar to that proposed by Zampogiannis et al. (2015). For any given 3D point cloud representing the reference object, a bounding box containing it (i.e., convex cuboid around the object) is created—see Figure 3.3a. The space around this object is divided into six non-overlapping pyramids representing the relations *left, right, front, behind, above* and *below*—see Figure 3.3b. In our implementation, the spatial relation of an object with respect to a reference object is determined by the non-overlapping pyramid around the reference that has most of the point cloud of the object. Also, any object with most of its point cloud located inside the bounding box of the reference object is said to be *in* the reference object. This definition of *in* can lead to errors, especially in domains with non-convex objects, e.g., a book that is actually *under* a large table may be classified (incorrectly) as being *in* the table because the bounding box of the table envelopes most of the point cloud of the book.

For ease of representation, our approach differs from Zampogiannis et al. (2015) in the definition of the distance-related prepositions: *touching, not-touching* and *far*. For a pair of point cloud clusters, the 10% closest distances

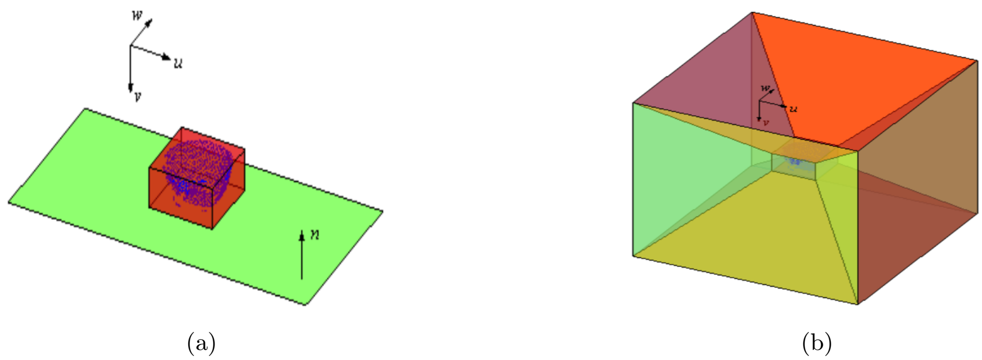


Figure 3.3: (a) Bounding box for point cloud of a particular object; and (b) Pyramids delimiting space around the bounding box.

between pairs of points drawn from the point clouds are computed, and the following criteria determine if the two objects are touching, not touching, or distinctly separated (i.e., far) from each other:

$$\begin{aligned} &touching \;\Rightarrow\; distance(10\%) \;\leq\; 0.01 \\ &not\text{-}touching \;\Rightarrow\; 0.01 < distance(10\%) \;< 1.0 \\ &far \;\Rightarrow\; distance(10\%) \;\geq\; 1.0 \end{aligned} \tag{3.4}$$

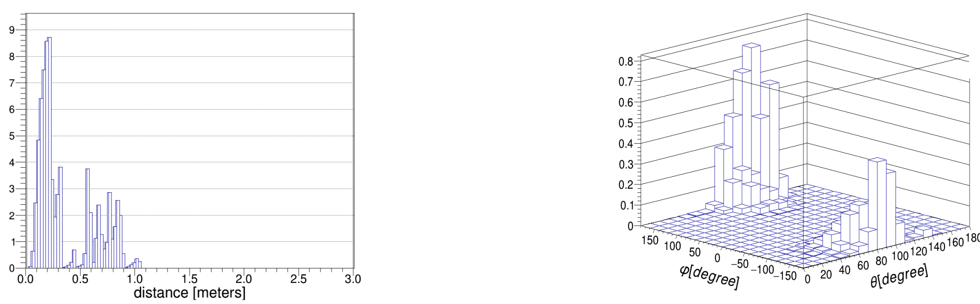
where distances are measured in meters, i.e., two objects are touching if the 10% closest distances are less than or equal to $0.01m$. Although the generic, manually-encoded grounding based on the QSR model does not change over time, it is initially used by the robot to identify spatial relations between objects. This is based on the reasonable assumption that the robot has an initial idea of its camera's pose with respect to the scene. Next, we describe a specialized grounding of spatial relations that can be acquired over time.

3.2.3 Metric Spatial Representation

An MSR-based grounding of the spatial relations is also used to identify spatial relations between objects. Unlike the QSR-based grounding, the MSR model supports incremental updates from observations and human feedback.

Assume that the MSR module receives a pair of point cloud clusters corresponding to two objects, and the prepositions of the spatial relations between the objects, e.g., from QSR or humans. The MSR module grounds each preposition using histograms, also referred to as "visual words", which are created by considering the point cloud data in a spherical coordinate system—each point is represented by its distance to a reference point and two angles (i) $\theta \in [0°, 180°]$; and (ii) $\varphi \in [-180°, 180°]$. On a robot, the coordinate frame for grounding is defined with respect to the robot's coordinate frame, its camera, and/or reference objects; information in one coordinate frame can be transformed to other coordinate frames. Also, sensor input processing introduces noise, but the non-monotonic logical reasoning and incremental learning modules of our architecture enable elegant recovery from errors due to noise.

Each of the seven position-based prepositions (*in, left, right, front, behind, above, below*) are grounded as 2D histograms of angles θ and φ, whereas each of the three distance-based prepositions (*touching, not-touching, far*) are grounded using 1D histograms of the 10% closest distances between points in pairs of objects. Figures 3.4a and 3.4b show a distance and position histogram respectively. All histograms are normalized to ensure that large objects with many points do not have any undue influence on the grounding of relations.



(a) (b)

Figure 3.4: (a) Example of 1D histogram grounding "not-touching"; (b) Example of 2D position histogram grounding "left".

Any learned MSR-based grounding(s) are used on new scenes. For any given pair of point cloud clusters in a new scene, the corresponding 2D and 1D histograms (i.e., visual words) are constructed. The existing visual words that are most similar to the extracted visual words are used to assign the distance-based and position-based spatial relations between the corresponding scene objects, e.g., "$object_1$ is above $object_2$ and touching it". These inferred spatial relations are automatically translated to statements added to the ASP program, e.g., $obj\_relation(above,\ obj_1,\ obj_2)$ and $obj\_relation(touch,\ obj1,\ obj2)$.

The similarity between visual words is computed using the *intersection* measure for 1D (distance) histograms. For the 2D (position) histograms, we use the χ^2 measure, e.g., for any two histograms H and G:

$$D_{\chi^2}(H,G) = \sum_i \frac{|h_i - g_i|^2}{2(h_i + g_i)} \tag{3.5}$$

where h_i and g_i are bins in H and G respectively; smaller values denote greater similarity. We use this measure for 2D histograms because the boundaries between the position-based relations are more difficult to define than those between distance-based relations. For implementation reasons, the inverse of $D_{\chi^2}(H,G)$ was used in the experiments.

Once the spatial relations between a pair of point cloud clusters have been determined in a new scene, this information updates the learned visual words using a standard normalized histogram merging approach, which means that the MSR-based grounding is continuously updated. For merging, the weight of the current scene for updating the existing visual word depends on whether the label is provided by humans or QSR-based grounding, according to the following equation:

$$H_{k+1} \leftarrow H_k + \frac{\beta}{\sqrt{k}}(H_{Current} - H_k) \begin{cases} \beta = \frac{\sqrt{k}}{2} \ for\ human\ input \\ \beta = 1\ for\ QSR\ grounding \end{cases} \tag{3.6}$$

where k corresponds to the number of examples that have updated a specific histogram so far, H_k is the existing histogram after k updates, and $H_{Current}$ is the histogram constructed from the current image. In the cases where the label is provided by humans, β cancels out the decremental effect of k, resulting in the updated histogram (H_{k+1}) being the result of the average of existing and current histograms:

$$H_{k+1} \leftarrow \frac{(H_k + H_{Current})}{2} \tag{3.7}$$

Equation 3.7 reflects the importance of human feedback in any stage, and enables our system to recover from errors and adapt to different environments, as discussed in the results section. If human feedback is not available, the influence of a new input on the existing visual word depends on the number of scenes that has updated it so far. This effect is caused by the term $\sqrt{k}$ in the denominator of equation 3.6.

3.2.4 Combined Model and Other Relations

In addition to ASP-based inference using QSR and MSR groundings, spatial relations between point cloud clusters can also be determined by human feedback. While the QSR-based grounding remains unchanged and the MSR-based grounding changes as new scenes are processed, human input is assumed to be accurate, i.e., each human participant providing feedback is expected to be able to interpret spatial relations correctly. Since the QSR-based and MSR-based groundings may disagree on the relation between some pairs of objects, the control node initially assigns high (low) confidence to the QSR-based (MSR-based) grounding. The relative confidence in each grounding is then updated based on the number of times the output from the grounding matches human input. The more reliable grounding is used for subsequent scenes. Incorrect human annotation can thus affect the confidence in a grounding only if the number of such annotations is comparable to the number of correct annotations.

Object shapes and sizes may also influence spatial relations depending on the viewpoint. However, since the MSR-based grounding is based on histograms of relative distances and angles, it can be used to infer spatial relations over a range of viewpoints. Also, the architecture has two mechanisms to limit and recover from errors. If the QSR-based grounding is applicable, e.g., viewpoint has not changed substantially, the system can use it to obtain an initial estimate of spatial relations and incrementally acquire the MSR-based grounding. If the QSR-based grounding is not applicable, it is still possible to acquire an MSR-based grounding from human input and use it for subsequent inference. Furthermore, the MSR-based grounding is obtained from a small number of images and is transferable.

There are some important caveats related to the proposed approach. First, the QSR-based grounding is assumed to be reasonably accurate initially; if this assumption does not hold and no human input is available, an inaccurate MSR-based grounding may be acquired, resulting in incorrect estimates of spatial relations. A case of incorrectly learned visual words, and how this can be fixed by using human feedback, is presented in the results section. Second, human feedback improves the specialized grounding (MSR) and overall accuracy, but it is not essential for estimating spatial relations. Third, the encoded prepositions (with learned groundings) are translated to logic statements (i.e., observation literals) in an ASP program. These observations and the commonsense knowledge encoded in the ASP program limit possible relations between scene objects and help infer composite relations (e.g., *on, close to, next to*, etc). As showed in Table 3.1, many of these prepositions can be expressed as a combination of the seven position-based and three distance-based primitive relations considered in our architecture. For instance, the spatial relation *on* may be defined by the axiom:

$$
\begin{aligned}
holds(obj\_relation(on, O_1, O_2), I) \ \leftarrow \ & holds(obj\_relation(above, O_1, O_2), I), \\
& holds(obj\_relation(touch, O_1, O_2), I).
\end{aligned} \tag{3.8}
$$

which states that if object O_1 is above O_2 and touching it, then O_1 is on O_2. Finally, we currently assume that each pair of objects is related through one position-based and one distance-based spatial relation, but not all the prepositions are (or need to be) mutually exclusive.

3.3 Experimental Setup

Recall that the architecture presented in this chapter aims to overcome the first gap identified in Section 2.6, which is the difficulty in updating the existing groundings from interaction. To demonstrate the incremental and interactive learning ability of the proposed architecture, the experiments were designed to test two hypotheses: **H1**, in which the proposed approach enables more effective use of human feedback; and **H2**, in which the combination of the manually-encoded QSR grounding and the automatically-learned

	on	on top of	upon	atop	over	near	close to	inside	within	beside	by	next to	under	underneath	beneath
in								√	√						
left						√	√			√	√	√			
right						√	√			√	√	√			
above	√	√	√	√	√	√	√								
below						√	√						√	√	√
front						√	√			√	√	√			
behind						√	√			√	√	√			
touching	√	√	√	√						√	√	√	√	√	
not-touching						√	√								√
far															

Table 3.1: Map of commonly used spatial prepositions – extracted by Belz et al. (2015) from VOC'08 1K – in primitives. VOC'08 1K contains descriptions of 1,000 images of Pascal VOC 2008 Shared Task Competition collected by Rashtchian et al. (2010), using Mechanical Turk.

MSR grounding performs better than each grounding used individually. The experiments seek to test the role of interactive learning in the efficient use of human feedback and the accuracy in grounding spatial relations between scene objects.

For experimental evaluation, we used the Table Object Scene Database (TOSD)[2] and simulated scenes. TOSD contains 111 scenes for training and 131 scenes for testing. Many of these scenes include complex object configurations (Figure 3.1a), while some scenes have only two objects (Figure 3.6a). Since TOSD includes segmentation labels but not spatial relation labels, we manually labeled 200 pairs of objects from the testing set. In addition, simulation scenes were generated with a real-time physics engine (Bullet physics library) by manually encoding the grounding of spatial relations. Different subsets of 21 household objects from the Yale-CMU-Berkeley dataset (Calli et al. 2015), along with a table and a shelf, were used to create 1400 simulated scenes (200 for each preposition). An additional 25 labeled scenes for each preposition (175 total) were used for training.

[2]https://repo.acin.tuwien.ac.at/tmp/permanent/TOSD.zip

The performance measure was the accuracy of the labels assigned to spatial relations. We also qualitatively evaluated the ability to identify and correct errors. Below, *all claims are statistically significant at the* 95% *significance level*.

3.4 Results and discussion

The first set of experiments was designed as follows, with the results summarized in Table 3.2:

1. Pairs of objects extracted from the TOSD training set were randomly divided into 10 subsets.

2. Seven pairs of objects from each subset were used to train the MSR-based grounding with human feedback. Each pair represents one of the position-based spatial relations (*in, left, right, front, behind, above, below*).

3. Seven pairs of objects from each subset labeled with human feedback, and 200 pairs with relations labeled using the QSR-based grounding, were used to train the MSR-based grounding.

4. The control node chose between QSR-based grounding and the MSR-based grounding trained using the QSR-based grounding and human feedback.

The three schemes (#2, #3, #4 above) were evaluated on 200 object pairs in test scenes of varying complexity. Table 3.2 indicates that the MSR-based grounding acquired using the QSR-based grounding (Scheme #3) makes better use of human feedback than that acquired not using the QSR-based grounding (Scheme #2), which supports <u>**H1**</u>. Note that the same amount of human feedback is provided with Schemes #2 and #3. The difference is that the latter scheme bootstraps off the generic knowledge encoded in the QSR-based grounding. These results indicate that using prior knowledge, an appropriate representation for knowledge, experience, and human feedback, improves performance. Also, the control node-based combination of the two

Training sets	Accuracy of labels over test set of 200 object pairs		
	MSR (feedback)	MSR (QSR + feedback)	Combined model
Sets 1	65%	77%	84%
Sets 2	82%	80%	94%
Sets 3	68%	80%	85%
Sets 4	66%	83%	87%
Sets 5	65%	74%	82%
Sets 6	68%	77%	86%
Sets 7	64%	87%	90%
Sets 8	64%	84%	91%
Sets 9	62%	82%	87%
Sets 10	52%	72%	81%
Mean	65%	79%	87%
Std Dev	7.2%	4.6%	8.3%

Table 3.2: Comparison of (a) MSR grounding trained with just human feedback; (b) MSR grounding trained with 200 pairs labeled by the QSR grounding and seven pairs labeled with human feedback; and (c) the combination of MSR grounding, trained as in (b), and QSR-based grounding with the choice made by the control node.

groundings provides better accuracy than just using the MSR-based grounding.

The second set of experiments was designed as follows, with the results summarized in Table 3.3:

1. Pairs of objects extracted from the training set of the TOSD were randomly divided into five subsets.

2. An MSR-based grounding was acquired using QSR-based labels for four out of the five subsets ($\approx$ 2000 pairs) in each run.

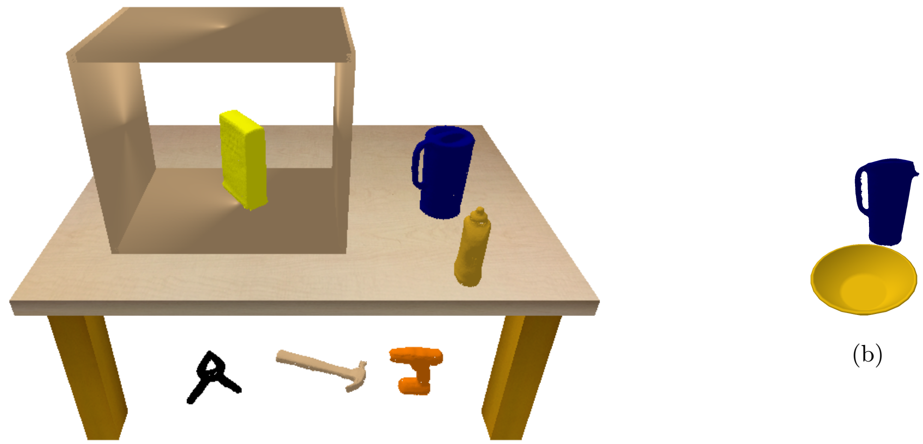
Training sets	Accuracy of labels over test set of 200 object pairs		
	QSR only	MSR trained by QSR	Combined model
Sets 1+2+3+4	70%	62%	96%
Sets 1+2+3+5	70%	62%	96%
Sets 1+2+4+5	70%	60%	95%
Sets 1+3+4+5	70%	60%	96%
Sets 2+3+4+5	70%	60%	96%
Mean	70%	61%	96%
Std Dev	0	1.1%	0.5%

Table 3.3: Comparison of (a) QSR-based grounding; (b) MSR-based grounding from $\approx$ 2000 pairs labeled with QSR-based grounding (no human feedback); and (c) using the control node to combine MSR-based grounding, as trained in (b), and QSR-based grounding.

3. The use of the control node to choose between the MSR-based grounding (trained as in Scheme #2 above) and the QSR-based grounding was also considered.

The two different schemes (#2, #3) were evaluated on a set of 200 object pairs in scenes of varying complexity. Ground truth, once again, was obtained manually. Table 3.3 indicates that the control-node based combination of the groundings estimates spatial relations more accurately than using either grounding individually, which supports hypothesis **H2**.

Next, the MSR-based groundings were acquired from different amounts of human feedback (with no QSR)—one, 15, and 25 training sets, each set with seven object pairs from simulated scenes. These groundings were tested on 1400 object pairs from simulated (test) scenes of varying complexity. Examples of simulated scenes are showed in Figure 3.5. Table 3.4 shows that spatial relations are estimated accurately even when a small number of labeled samples are used to acquire the grounding.



(a) (b)

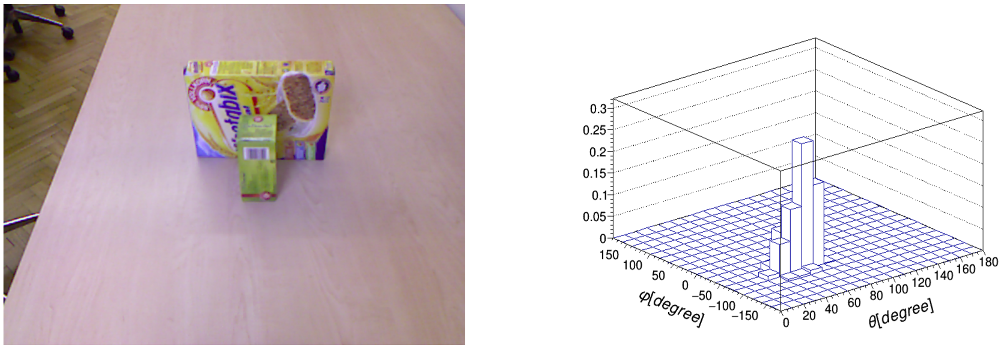
Figure 3.5: Examples of simulated scenes used in experiments.

Model	QSR	MSR after 1 training set	MSR after 15 training sets	MSR after 25 training sets
Accuracy of labels over test set of 1400 object pairs	61.9%	96.1%	98.5%	98.6%

Table 3.4: QSR-based grounding compared with MSR-based groundings obtained using different amounts of human feedback.

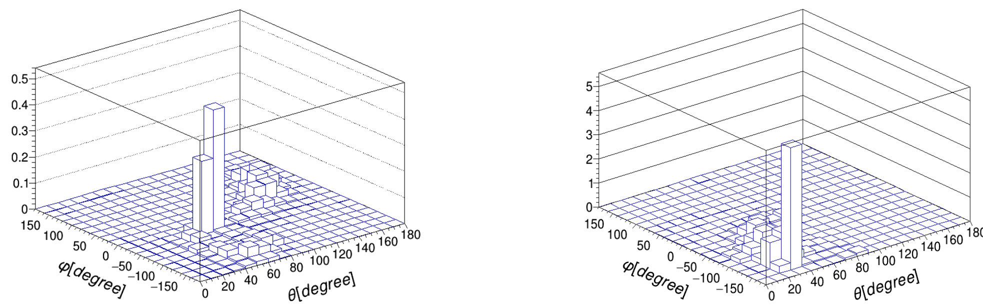
The next experiments were similar to those for Table 3.2, but with a larger number of simulated scenes. The MSR-based grounding acquired using just human input had accuracy of 95.9%, whereas the grounding obtained using human input and the QSR-based grounding had accuracy of 97.2%. These results are similar to those with the TOSD showed in Table 3.2.

Further analysis indicates that most errors from the control node-based combination of the groundings correspond to truly ambiguous spatial relations, e.g., a scene in which object A can be considered to be to the "left" or "behind" object B. Multiple labels are acceptable in such cases, and we just need to let the inference system allow multiple answers. In other cases, e.g., when each grounding is used individually, errors are due to the grounding being (or becoming) inaccurate—even in these cases, results do not depend on the order in which the training and test data are provided.



(a) (b)

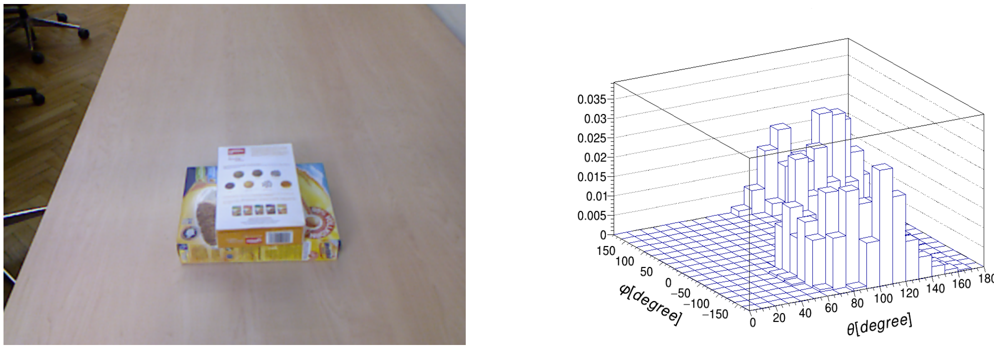
Figure 3.6: (a) Image from TOSD dataset; (b) Histogram generated from the image using the smaller box as the reference object.



(a) (b)

Figure 3.7: Histograms representing learned MSR groundings for: (a) *above*; and (b) *behind*.

We also evaluated the ability to identify and correct errors. For the TOSD image in Figure 3.6a, the MSR-based grounding incorrectly stated that the larger box was *above* the smaller one. We compared the learned visual words for this label and correct label ("behind")—Figures 3.7a and 3.7b respectively—with the histogram extracted from the object pair in the image—Figure 3.6b. The inverse of the χ^2 measure between the learned and observed visual words was 0.325 for *above* and 0.319 for *behind*. Even the QSR-based grounding detected 349 points in the *above* region and 23 in the *behind* region. The error was thus due to the incorrect input provided by the QSR-based grounding to the MSR-based grounding. We then visually compared the 2D histograms between the two objects (Figure 3.6b) with the MSR-based grounding for *above* and *behind* (Figure 3.7). The extracted his-



(a) (b)

Figure 3.8: (a) Image with one object *above* another; and (b) revised 2D histogram for *above*.

togram was more similar to the grounding for *above*—under standard viewpoints and orientations, $\theta > 90°$ for *above*, but many points corresponded to $\theta \approx 60°$ in this case. To correct this error, we processed an image that actually contained an instance of the *above* relation, Figure 3.8a. The θ values in the revised histogram for *above* were mostly $\in [90°, 120°]$—Figure 3.8b. The MSR-based grounding then provided the correct spatial relation between the objects in Figure 3.6a—the inverse of the χ^2 similarity scores were 0.319 for *behind* and 0.088 for *above* after the correction.

3.5 Conclusion

The correct interpretation of spatial relations between objects is a basic skill that robots should master to assist humans in complex domains. The architecture described in this chapter uses Answer Set Prolog (ASP) to represent and reason with incomplete domain knowledge, which includes spatial relations. The relations provided by a generic qualitative grounding (QSR) and human input (if available) are used to incrementally acquire a more specialized quantitative grounding of spatial relations (MSR). Also, a relative measure of confidence in the two groundings is computed to enable the use of the more reliable grounding for inferring spatial relations in the subsequent scenes. Experimental evaluation demonstrates the ability to reliably estimate spatial relations in a benchmark dataset of complex tabletop images and simulated scenes of furniture, even with a small number of labeled training samples. In the next chapter, we include modules for scene understanding that use these spatial relations and explore the interplay between reasoning and learning on the context of simulated scenes understanding.

Chapter 4

Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning

In Chapter 3, we presented an approach to incrementally and interactively learn the grounding of spatial relations between scene objects. Here, these groundings are used as input to an architecture that embeds non-monotonic logical reasoning with incomplete commonsense domain knowledge, and incremental inductive learning of constraints governing domain states, to guide the learning of deep network architectures. This architecture is tested in the tasks of estimating the *partial occlusion* of objects and the *stability* of object configurations, based on limited training examples, in the context of an assistive robot clearing away toys that children have spread in different rooms, as showed in Figure 4.1.

4.1 Introduction

In this chapter we describe how the architecture does the following:

- Attempts to perform the estimation tasks based on non-monotonic logical reasoning with incomplete commonsense domain knowledge and the extracted geometric relationships between scene objects.

Figure 4.1: A simulated scene with toys. The robot has to reason about occlusion and stability of structures to reduce clutter.

- Automatically identifies relevant regions of the images not processed by non-monotonic logical reasoning; these regions guide the training of deep networks and are processed by the learned networks during testing.
- Uses the labeled examples, i.e., images with occlusion labels for objects and stability labels for object structures, to train decision trees for incremental learning of previously unknown constraints governing domain states.

This chapter is based on the conference paper published in Mota and Sridharan (2019a).

4.2 Proposed Architecture

The proposed architecture takes RGB-D images of scenes with different object configurations as input. The key components are showed in Figure 4.2. During training, the inputs include the occlusion labels of objects and the stability labels of object configurations in the images. The components presented in Chapter 3 are used to ground the spatial relations between objects. An object is considered to be occluded if the view of any minimal fraction of

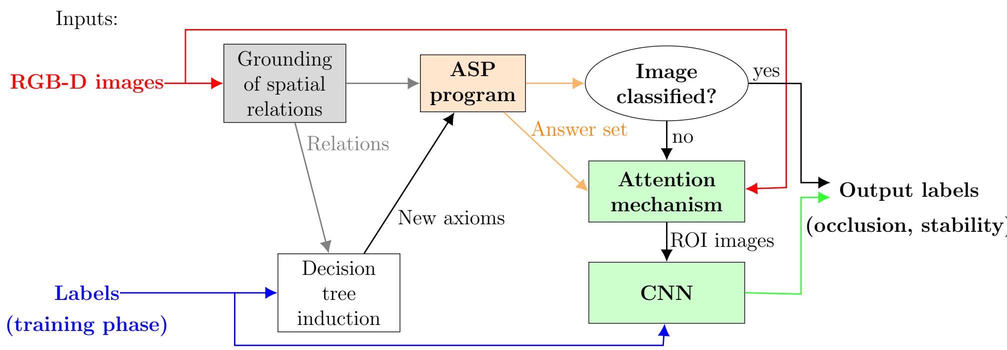


Figure 4.2: Architecture combines the complementary strengths of non-monotonic logical reasoning, deep learning, and decision tree induction, in order to perform the scene understanding tasks reliably and efficiently.

its frontal face is hidden by another object, and a structure is unstable if any object in the structure is unstable. A decision tree induction algorithm maps object attributes and spatial relations to the target classes. Branches in the tree that have sufficient support among the training examples are used to construct axioms representing state constraints. The learned constraints are encoded in an ASP program along with the commonsense domain knowledge and the computed spatial relations. If ASP-based reasoning provides the desired labels, no further analysis of this image is performed. Otherwise, an attention mechanism uses domain knowledge to identify the image's Regions of Interest (ROI), with each ROI containing one or more objects. A CNN is trained to map these ROIs to the desired labels. During testing, any input RGB-D image is assigned the desired class labels either by ASP-based reasoning or by processing the image ROIs using the learned CNN (i.e., decision trees are not used). We describe these components using the following illustrative domain.

Example 1. [Robot Assistant (RA)] A simulated robot analyzes images of scenes containing toys in different configurations. The goal is to: (i) estimate occlusion of objects and stability of object structures; and (ii) rearrange object structures so as to minimize clutter. Domain knowledge includes object attributes such as $size$ (small, medium, large), $surface$ (flat, irregular) and $shape$ (cube, cylinder, duck), and the $relation$ between objects (above, below, front, behind, right, left, close). The robot can move objects to achieve the desired goals. Knowledge also includes axioms governing domain dynamics, but some axioms may be unknown, e.g.:

- Placing an object on top of an object with an irregular surface causes instability;
- Removing all objects in front of an object causes this object to be not occluded.

Inducted decision trees can provide the unknown axioms to be automatically encoded in the ASP program.

4.2.1 Knowledge Representation with ASP

As in Chapter 3, the domain is initially described in action language $\mathcal{AL}$, and then translated to an ASP program for non-monotonic logical inference. The domain description in $\mathcal{AL}$ comprises a system description $\mathcal{D}$ and a history $\mathcal{H}$. $\mathcal{D}$ comprises a sorted signature Σ and axioms. In addition to the domain described in Chapter 3, Σ also includes the sorts $robot$, $size$, and $surface$. Statics include some object attributes such as $obj\_size(object, size)$ and $obj\_surface(obj, surface)$. Again, the fluents $obj\_relation(relation, object, object)$ model relations between objects in terms of their arguments' sorts, e.g., $obj\_relation(above, A, B)$ implies object A is *above* object B—the last argument in these relations is the reference object. Fluents also describe other aspects of the domain, e.g.:

$$in\_hand(robot, object), \quad stable(object) \tag{4.1}$$

which describe if a robot has an object in hand, and whether a particular object is stable. Actions in the RA domain include $pickup(robot, object)$ and $putdown(robot, object)$. A *state* of the domain is then a collection of ground literals, i.e., statics, fluents, actions and relations with values assigned to their arguments.

The axioms of $\mathcal{D}$ are defined in terms of the signature and govern domain dynamics. These axioms include a distributed representation of the constraints related to domain actions, i.e., causal laws and executability conditions that define the preconditions and effects of actions, and state constraints. The axioms of the RA domain include statements such as:

$$pickup(robot, object) \ \textbf{causes} \ in\_hand(robot, object) \tag{4.2a}$$

$$obj\_relation(below, B, A) \ \textbf{if} \ obj\_relation(above, A, B) \tag{4.2b}$$

$$obj\_relation(behind, B, A) \ \textbf{if} \ obj\_relation(infront, A, B) \tag{4.2c}$$

$$\textbf{impossible} \ pickup(robot, object) \ \textbf{if} \ in\_hand(robot, object) \tag{4.2d}$$

where Statement 4.2(a) is a causal law which states that if the robot executes the *pickup* action on an object, it ends up holding the object. Statements 4.2(b-c) describe state constraints regarding some spatial relations between two objects. Statement 4.2(d) describes an executability condition which indicates that a robot cannot pick up an object that it is already holding.

A history $\mathcal{H}$ of a dynamic domain typically includes records of observations of fluents received at a particular time step, i.e., $obs(fluent, boolean, step)$ and actions actually executed by the robot at a particular time step, i.e., $hpd(action, step)$.

To reason with the incomplete domain knowledge, we construct the CR-Prolog/ASP program $\Pi(\mathcal{D}, \mathcal{H})$ from the system description $\mathcal{D}$ in $\mathcal{AL}$ and the history $\mathcal{H}$—please see our code repository (Mota and Sridharan 2019b). The program Π includes the signature and axioms of $\mathcal{D}$, inertia axioms, reality checks, closed world assumptions for defined fluents and actions, and observations, actions, and defaults from $\mathcal{H}$. Planning, diagnostics and inference tasks can then be reduced to computing *answer sets* of Π, which represent the beliefs of the robot associated with Π (Gelfond and Kahl 2014). We use SPARC (Balai, Gelfond, and Yuanlin Zhang 2013) to compute answer set(s) of ASP programs.

For instance, Statements 4.2(a-d) of $\mathcal{AL}$ are translated to:

$$holds(in\_hand(robot, object), I + 1) \leftarrow \tag{4.3a}$$
$$occurs(pickup(robot, object), I).$$
$$holds(obj\_relation(above, A, B), I) \leftarrow \tag{4.3b}$$
$$holds(obj\_relation(below, B, A), I).$$
$$holds(obj\_relation(infront, A, B), I) \leftarrow \tag{4.3c}$$
$$holds(obj\_relation(behind, B, A), I).$$
$$\neg occurs(pickup(robot, object), I) \leftarrow \tag{4.3d}$$
$$holds(in\_hand(robot, object), I).$$

where the predicate $holds(fluent, step)$ implies that a particular fluent holds true, and the predicate $occurs(action, step)$ implies that a particular action is to be executed at a particular timestep.

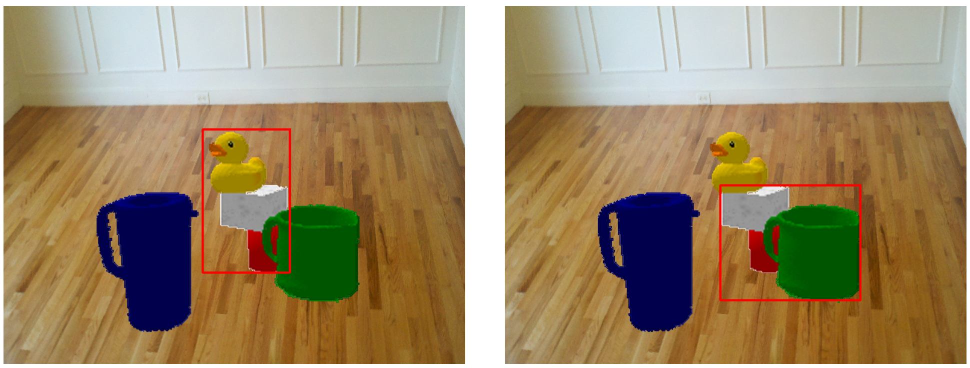
The spatial relations extracted from RGB-D images are converted to statements in ASP program. The program also includes axioms that encode default knowledge, e.g., statements such as "larger objects on smaller objects are typically unstable".

$$\begin{aligned} \neg holds(stable(A), I) \leftarrow\ & holds(obj\_relation(above, A, B), I), \\ & size(A, large),\ size(B, small), \\ & not\ holds(stable(A), I) \end{aligned} \tag{4.4}$$

Since the robot only believes that which it is forced to believe, the inability to compute an answer set indicates an unresolved inconsistency due to incomplete knowledge or an error in the encoding that needs to be probed further. In the context of the scene understanding tasks, the robot would either be unable to make a decision regarding occlusion and stability, or provide an incorrect estimate (when ground truth is available). This situation is addressed in our architecture using the attention mechanism and deep networks as described below.

4.2.2 Attention Mechanism

The attention mechanism module is used when ASP-based reasoning cannot assign labels to objects in the input image, or the assigned label is incorrect (during the training phase). In each such image, this module identifies and directs attention to regions of interest (ROIs) that contain information relevant to the task at hand. To do so, it first identifies each axiom in the ASP program whose head corresponds to a relation/fluent of interest. The relations in the body of each selected axiom are used to identify ROIs that are considered for further processing; the remaining image regions are unlikely to provide relevant information and are not analyzed further.



(a) (b)

Figure 4.3: Examples of ROIs highlighted by the attention mechanism for: (a) balance of structure; (b) occlusion of objects.

For instance, consider the task of estimating the stability of object configurations in Figure 4.3a. The head of Statement 4.5(a), which implies that a particular object is stable, holds true in any state in which all the relations in the body of the axiom are satisfied. Statement 4.6 defines some conditions under which an object is considered to be unstable. Both these statements will be considered by the attention mechanism for this task. As the body of these two axioms contains the spatial relation *above*, when analyzing stability in Figure 4.3a, the attention mechanism will consider the stack comprising the duck, the red can and the white cube (indicated by the red rectangle), since they satisfy this relation – the mug and the pitcher can be disregarded.

In a similar manner, the Statement 4.5(b) will only be explored further when the task is to examine the occlusion of objects. As this equation contains the relation *behind* in its body, when analyzing occlusion in Figure

4.3b the attention mechanism will only consider pairs of objects for which such relation holds true. The red rectangle in the figure indicates that the region containing the mug, the red can and the white cube are examined, whereas the other two objects (duck and pitcher) and the rest of the picture are disregarded.

The identified ROIs are then used for constructing CNN as described below.

4.2.3 Convolutional Neural Networks

The ROIs identified by the attention mechanism serve as input to a deep network. Recall that ROIs are only extracted from images that could not be classified using ASP-based reasoning, and that pixels of any such ROI are considered to provide information relevant to the task at hand. We explore two variants of a CNN. The training dataset comprises ROIs and the target labels to be assigned to objects (and structures) in the ROIs. The CNN learns the mapping between the image pixels and target labels, and then assigns these labels to ROIs in previously unseen test images that ASP-based reasoning is unable to process.

As discussed in Chapter 2, a CNN is usually composed of a number of layers, each layer representing a different level of abstraction. For instance, while estimating the stability of object configurations, the CNN's layers may represent attributes such as whether: (i) a tower of blocks is aligned; (ii) a round object is under another object; or (iii) a tower has a small base.

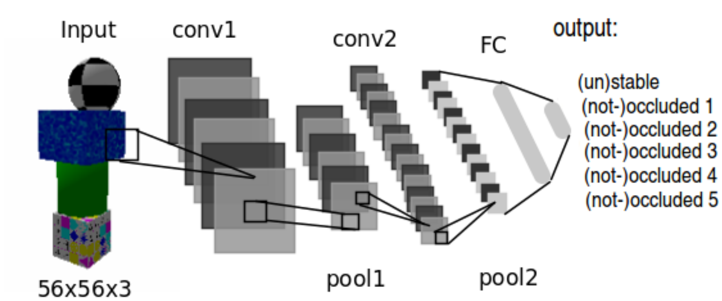


Figure 4.4: Lenet architecture.

In this chapter, we report the results obtained for two well-known CNN architectures: (i) Lenet (LeCun et al. 1998), initially proposed for recognizing hand-written digits; and (ii) Alexnet (Krizhevsky, Sutskever, and Hinton 2012), which has been widely used since it provided the best results on the Imagenet benchmark dataset.

The Lenet has two convolutional layers, each one followed by a max-pooling layer and an activation layer. Two fully connected layers are placed at the end. Unlike the 28×28 gray-scale input images and the ten-class softmax output layer used in the original implementation, we consider 56×56 RGB images as input and an output vector representing the occlusion and stability of each object in the image. Figure 4.4 is a pictorial representation of this network. As described later in Section 4.3, we consider ROIs with up to five objects, and the network outputs estimate the occlusion of each object and the stability of the structure in the ROI.

The Alexnet architecture, on the other hand, contains five convolutional layers, each followed by max-pooling and activation layers, along with three fully connected layers at the end. In our experiments, 227×227 RGB images were used as input and the output classes determined the target variables estimating occlusion and stability. The number of outputs is the same as with the Lenet architecture.

Due to the multi-class labeling problem, the sigmoid activation function was used in both networks. We used the Adam optimizer (Kingma and Ba 2014) in TensorFlow (Abadi et al. 2016), with a learning rate of 0.0001 for the Alexnet network and 0.0002 for the Lenet network, and the weights were initialized randomly. The number of training iterations varied depending on the network and the number of training examples. For example, Lenet, using 100 and $5,000$ image samples, was trained for $10,000$ and $40,000$ iterations respectively, whereas the Alexnet, with 100 and $5,000$ samples, was trained for $8,000$ and $20,000$ iterations respectively. The learning rate and number of iterations were chosen experimentally using validation sets. The number of epochs was chosen as the stopping criteria, instead of the training error, in order to allow the comparison between networks learned with and without the attention mechanism. The code for training the deep networks is in our open source repository (Mota and Sridharan 2019b).

The CNN is only trained on regions of images for which ASP-based reasoning provides an incorrect outcome or is unable to provide an outcome.

We consider any such trained CNN to represent previously unknown knowledge not encoded in the ASP program, i.e., the observed incorrect outcome or lack of any outcome is due to reasoning with incomplete or incorrect knowledge. To revise the knowledge encoded in the ASP program, and to better understand the behavior of the trained deep networks, we used decision tree induction for incrementally learning the previously unknown state constraints from the same images and labels used to train the deep networks.

4.2.4 Decision Tree Induction and Axiom Merging

The proposed architecture builds on the relational representation to incrementally learn previously unknown knowledge in the form of axioms that represent state constraints, and to merge them with the existing knowledge. Specifically, separate decision trees are constructed in the RA domain for estimating stability and occlusion, using the spatial relations between pairs of objects (and the attributes of these objects) identified in the ROIs used for training the CNNs. The labels assigned to the leaf nodes are stable/unstable or occluded/not occluded. Some examples of decision trees are shown in Figures 4.5 and 4.6.

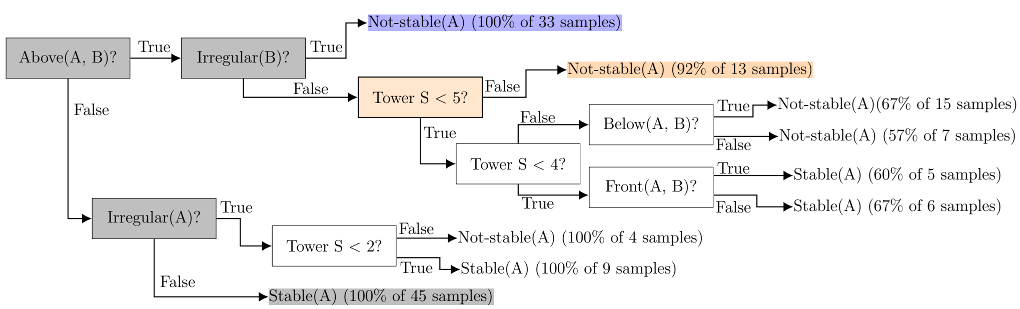


Figure 4.5: Example of a decision tree constructed for stability estimation using some labeled examples. Highlighted branches are used to construct previously unknown axioms.

We use an existing algorithm that constructs decision trees by computing the potential change in entropy (i.e., information gain) caused by a split based on each attribute. One half of the available examples are used for training. Once a tree is constructed, any branch of the tree in which the leaf

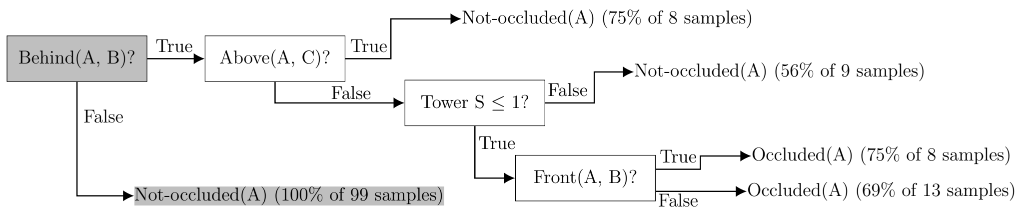


Figure 4.6: Example of a decision tree constructed for occlusion estimation using some labeled examples. Highlighted branch is used to construct previously unknown axiom.

represents a precision higher than 95%, i.e., most examples corresponding to a particular class, is used to construct candidate axioms that are validated using the other half of the labeled examples. The validation process: (i) removes axioms without a minimum level of support from the training examples; and (ii) compares the discovered axioms to only retain the most general version of each axiom. Since the number of labeled examples is small, we reduce the effect of noise through an ensemble learning approach, i.e., we repeat the learning and validation steps a number of times (e.g., 100) and only the axioms voted more than a minimum number of times (e.g., 40%) are encoded in the ASP program for subsequent reasoning.

Consider the branches highlighted in gray in Figures 4.5 and 4.6, which can be translated into the following axioms:

$$stable(A) \leftarrow \ \neg obj\_relation(above, A, B),\ \neg obj\_surface(A, irregular) \tag{4.5a}$$

$$\neg occluded(A) \leftarrow \ \neg obj\_relation(behind, A, B) \tag{4.5b}$$

where Statement 4.5(a) implies that any object without an irregular surface that is not above another object is stable, whereas Statement 4.5(b) says that an object is not occluded if it is not located behind another object. As another example, the branch highlighted in gray and blue in Figure 4.5 translates to:

$$\neg stable(A) \ \leftarrow \ obj\_relation(above, A, B),\ obj\_surface(B, irregular) \tag{4.6}$$

which states that an object is unstable if it is located above another object with an irregular surface.

Our architecture is also able to discover default knowledge that holds in all but a few exceptional circumstances. To find these axioms while still allowing for some exceptions, we reduced the threshold for selecting a branch of a tree to construct candidate axioms (e.g., from 95% to 70%). As an example of default knowledge, the branch highlighted in gray and orange in Figure 4.5 could translate to:

$$\neg stable(A) \;\leftarrow\; tower\_height(A, N),\; N > 4,\; not\ stable(A). \tag{4.7}$$

which says that an object placed on a tower taller than 4 objects is typically unstable. The term "typically" is expressed in the axiom by the default negation of the negated head (*not* $stable(A)$). This is included in the body of any default axiom learned (i.e., any axiom learned when the threshold for selecting a branch is reduced).

Algorithm 1 describes the algorithm for learning axioms and merging them with the existing knowledge (Lines 1-14). If labeled training samples (i.e., image ROIs with stability and occlusion labels of objects) are available, they are used to induce new state constraints (Lines 2-11). A training set is created by randomly selecting 50% of the labeled examples, with the remaining examples making up the validation set (Line 4). The training set is used to construct the decision tree(s) based on each attribute that has not yet been used (Line 5); the attribute that is likely to provide the highest reduction in entropy is selected. Next, the branches of the tree (from root to leaves) that satisfy certain minimum requirements are selected to construct candidate axioms (Line 6). These minimum requirements include thresholds on purity of samples at any given leaf, and on the support from the labeled examples, e.g., $\geq$ 95% samples at the leaf belong to a particular (correct) label, a branch under consideration has support from $\geq$ 5% of the training samples, etc. The selected branches of the learned decision trees represent previously unknown candidate constraints, and the thresholds are set to construct such candidate axioms cautiously. Small changes in the value of these thresholds do not cause any significant change in the branches of

Algorithm 1: **Learning and merging axioms**

Input : Relational domain attributes from image ROIs; occlusion and stability labels of objects in ROIs; thresholds th_1 (95%, purity threshold), th_2 (5%, support threshold), th_3 (40%, tree support threshold), th_4 (10%, axiom strength threshold); ensemble_count (100).

Output: Learned axioms.

```
while true do
    if labeled_samples then
        for j = 1 : ensemble_count do
            // Split training samples for learning and validation
            training_set, validation_set =
              random_split(labeled_samples)
            // Decision tree induction
            learned_tree = tree_induction(training_set)
            // Create candidate axioms
            candidate_axioms = select(learned_tree, th1, th2)
            // Validate axioms
            validated_axioms = validate(candidate_axioms,
              validation_set, th2)
        end
        // Choose validated axioms with sufficient support
        axioms = select(validated_axioms, th3)
        // Add validated axioms and merge similar axioms
        add_merge(axioms)
    end
    // Update strength of axioms
    update_strength(axioms)
    // Remove axioms with low strength
    remove(axioms, th4)
end
```

the tree selected to construct axioms. The values of these thresholds can be revised to achieve different desired behavior. For instance, to identify default constraints that hold in all but a few exceptional circumstances (see Section 4.2.1), we lowered the threshold for selecting a branch of a tree to construct candidate axioms (from 95% to 70%). As we will discuss later, lowering the thresholds results in the discovery of additional axioms, but also introduces noise.

Once the candidate axioms are constructed, each one is validated using the other (so far unseen) half of the labeled examples (Line 7). The validation process removes axioms without a minimum level of support (e.g., 5%) from the labeled examples. Since the number of labeled examples available for training is often small, we reduce the effect of noise through an ensemble learning approach (loop in lines 3-8), i.e., we repeat the learning and validation steps a number of times (e.g., 100) and only the axioms identified in more than a minimum number of iterations (e.g., 40%, Line 9) are retained. Adding all retained axioms can lead to the ASP program including different versions of the same axiom over time. For instance, two axioms may have identical heads with one axiom's body containing all the literals of the other, or two ground axioms may include sorts that are subsorts of a more general sort. To address this issue, similar axioms are grouped together. Each possible combination of axioms from different groups (one from each group at a time) is then encoded in an ASP program along with the axioms that do not belong to any such group. The resulting program is used to classify ten labeled scenes chosen randomly. Axioms in the program that results in the highest accuracy are retained whereas the other axioms in each group are discarded (Line 10).

The axiom learning approach described so far is based on a small number of labeled examples in a dynamic domain. The learned axioms may be incorrect (e.g., incorrect negation in the head, or incorrect literals in the body), incomplete (e.g., one or more missing literals in the body), or overspecified (e.g., one or more irrelevant literals in the body). Reasoning with these axioms can lead to sub-optimal or incorrect behavior. To address this issue, we incorporated a heuristic approach inspired by the human forgetting mechanism. This approach associates a strength to each axiom. An axiom's strength is revised over time based on a decay factor using an exponential relation: $axiom\_relevance = e^{-\alpha . n}$, where α represents the decay factor (initially 1), and n is the number of time steps since the axiom was learned. In each time step, irrespective of whether any new axioms are learned, the strength of all learned axioms are updated (line 12). If an axiom is reinforced, i.e., learned again or used, its strength is elevated to the maximum value (i.e., 1) again, and its decay factor is divided by $\sqrt[n]{2}$, a value chosen experimentally such that it varies between 2 (for $n = 1$) and 1 (for $n \to \infty$). Any axiom whose strength value falls below a threshold (e.g., 0.1) is removed from further consideration (line 13).

4.3 Experimental setup

Since the proposed architecture attempts to mitigate the gap 2 identified in Section 2.6, which is the large amount of training data required by deep networks and their difficulty in learning incrementally, the experiments were designed to test the following hypotheses:

H1 Reasoning with commonsense domain knowledge and the attention mechanism improves the accuracy of deep networks.

H2 Reasoning with commonsense domain knowledge and the attention mechanism reduces sample complexity and time complexity of training deep networks.

H3 The architecture is able to incrementally learn previously unknown axioms, and use these axioms to improve the accuracy of decision making.

H4 Our approach for revising the strength of axioms and merging the learned with the existing knowledge is able to identify and remove incorrect axioms.

To simulate experiments in a dynamic domain for which a large number of training samples are not available, we used a real-time physics engine (bullet physics library) to generate 6000 labeled images for estimating occlusion and stability of objects. The dataset containing these images and correspondent point cloud files are available in an open source repository (Mota and Sridharan 2020a). Each image had ROIs with up to five objects with different colors, textures and shapes. The objects included cylinders, spheres, cubes, a duck, and five household objects from the Yale-CMU-Berkeley dataset (apple, pitcher, mustard bottle, mug, and cracker box) (Calli et al. 2015). We considered three different arrangements of these objects:

- **Towers**: images containing two to five objects stacked on top of each other;
- **Spread**: images with five objects placed on the flat surface (i.e., the ground); and
- **Intersection**: images with two to four objects stacked on each other, with the rest (one to three) spread on the flat surface.

The vertical alignment of stacked objects is randomized, creating either a stable or an unstable arrangement. The horizontal distance between spread objects is also randomized, which can create scenes with complex, partial or no occlusion. Lighting, orientation, camera distance, camera orientation, and background, were also randomized. Also, for the experimental trials summarized below, the ASP program was initially missing three state constraints each, which were related to stability estimation and occlusion estimation.

A second dataset was derived from the dataset described above to simulate the effect of the attention mechanism. Recall that this module extracts ROIs from images in the original dataset that could not be classified using ASP-based reasoning. This is carried out by identifying relevant axioms and relations in the ASP program. Only pixels in these ROIs were considered for analysis. CNNs trained using these two datasets were compared as a function of the amount of training data and the complexity of the networks. Occlusion is estimated for each object (i.e., five outputs) and stability is estimated for the structure (i.e., one output). An additional 600 labeled simulated scenes were also created and used for evaluation, e.g., for the approach to update the strength of the learned axioms (more details later).

The main performance measure was the accuracy of the labels assigned to objects and structures in images. Below, all claims are statistically significant at the 95% significance level. As the baseline for comparison, we trained the Lenet and Alexnet architectures without the commonsense reasoning and attention mechanism modules, i.e., directly on the RGB-D input images, and evaluated them on the test dataset.

4.4 Results and discussion

The first set of experiments evaluated the architecture ability to label objects and structures in images with respect to stability and occlusion. It was designed as follows, with results summarized in Figure 4.7:

1. Training datasets of different sizes (100, 200, 1000, and 5000 images) were used to train the Lenet and Alexnet networks. The remaining images were used to test the learned models. The baseline CNNs – with results plotted as "Lenet" and "Alexnet" in Figure 4.7 – do not use the attention mechanism and commonsense reasoning.

2. The datasets after the application of the attention mechanism were derived from the original datasets in Step 1. The selection of images as well as the pixels from each image was based on the target task and relations of interest.

3. The datasets created in Step 2 were used to train and test the Lenet and Alexnet networks, with the results plotted as “Lenet(Att)” and “Alexnet(Att)” in Figure 4.7.

Figure 4.7 indicates that integrating commonsense reasoning with deep learning improves the accuracy of the deep networks for the estimation of stability and occlusion. Training and testing the deep networks with only relevant ROIs of images that cannot be processed by commonsense reasoning simplifies the learning process, making it easier to learn accurate mapping between inputs and outputs and resulting in higher accuracy than the baselines for any given number of training images. The improvement is more pronounced when the training set is smaller, but there is improvement at all training dataset sizes considered in our experiments. These results support hypothesis **H1**.

Figure 4.8 shows two examples of the improvement provided by the attention mechanism. In Figure 4.8a, both *Lenet* and *Lenet(Att)* were able to recognize the occlusion of the *red cube* caused by the *green mug*, but only the latter, which uses the attention mechanism and commonsense reasoning, was able to estimate the instability of the tower. In Figure 4.8b, both networks

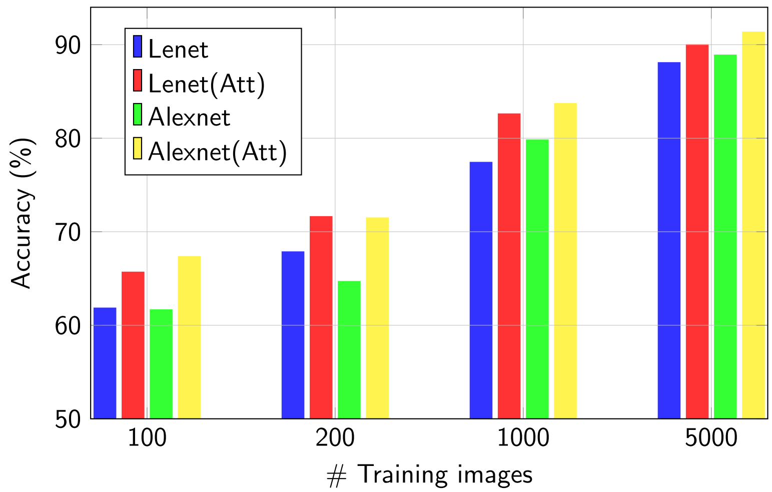
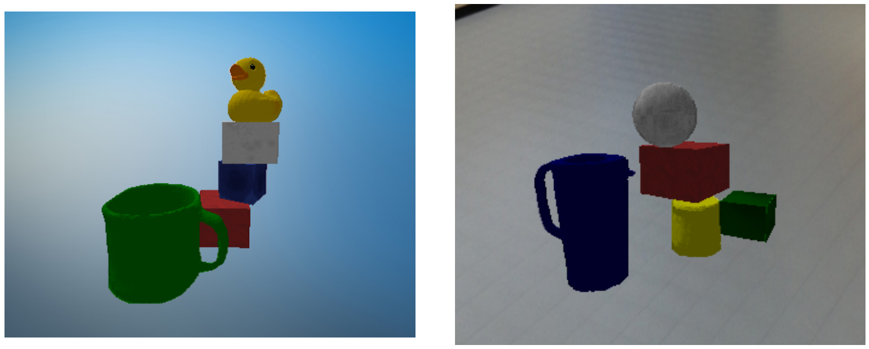


Figure 4.7: Accuracy of Lenet and Alexnet with and without commonsense reasoning and the attention mechanism. The number of background images was fixed at 100. Our architecture improves accuracy in comparison with the baselines.

correctly predicted the instability of the tower. However, only *Lenet(Att)* was able to identify the occlusion of the *green cube* by the *yellow can*. The classification errors are most probably because a similar example had not been observed during training This is a common limitation of deep architectures. The attention mechanism eliminates the analysis of unnecessary parts of images and focuses only on the relevant parts, resulting in a more targeted network that provides better classification accuracy. For these experiments, the CNNs were trained with 1000 images.

The number of different backgrounds (selected randomly) was fixed at 100 for the experimental results in Figure 4.7. The effect of the background on the observed performance varies depending on the number of training examples. For instance, we had (on average) one image that used each background image when the training data had 100 training samples, and we had 50 images per background for the training dataset with 5000 training examples. However, in real scenarios, it is unlikely that we will get a uniform distribution of backgrounds; other factors such as lighting, viewpoint, and orientation will be different in different images. To analyze the effect of different backgrounds, we explored the use of the Lenet architecture with



(a) (b)

Figure 4.8: Examples of test images for Lenet and Lenet(Att): (a) both detected the occlusion of the red cube by the green mug, but only the latter correctly estimated the tower's instability; and (b) both predicted the instability of the tower, but only Lenet(Att) detected the obstruction of the green cube by the yellow can.

different number of training examples (100 and 5000) and different number of backgrounds (30, 50, and 100). As shown in Figure 4.9, varying the background does have an impact on accuracy, which degraded from $\approx 65\%$ for one background per 10 images to $\approx 62\%$ when we have one background per image (i.e., 100 backgrounds for 100 images). The degradation is smaller, i.e., $\approx 1\%$, for 5000 training examples with number of backgrounds varying from $10 - 100$; however, for 1000 backgrounds (one background per five training images) the accuracy was reduced in $\approx 2\%$. These results indicate that a network trained with a larger number of images is less sensitive to variations in background, and that the inclusion of different backgrounds has a negative effect on performance for the baseline (e.g., Lenet) architecture. On the other hand, Figure 4.9 shows that the Lenet(Att) presents similar accuracy regardless of variations in the background, which indicates that the inclusion of commonsense reasoning and attention mechanism minimizes the effect of background on classification performance.

The second set of experiments was designed as follows to evaluate the effect of the proposed architecture on the computational effort for training deep networks, with results summarized in Figure 4.10:

1. The Lenet and Alexnet networks were trained with training datasets

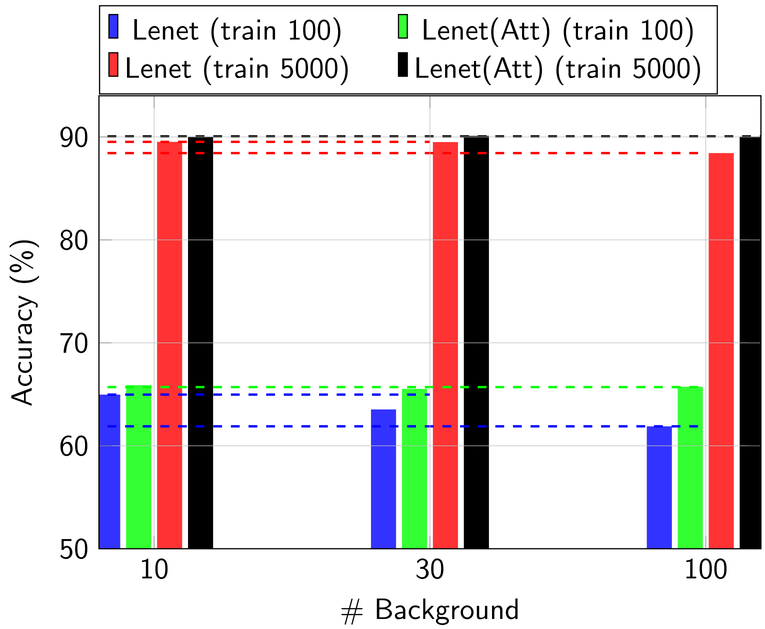


Figure 4.9: Effect of changing the number of backgrounds on the accuracy of the Lenet and Lenet(Att) networks for 100 and 5000 training images. Without the attention mechanism and commonsense reasoning, increasing the number of backgrounds reduces the classification accuracy.

containing between $100 - 1000$ images, in step-sizes of 100. Separate datasets were created for testing. As in the previous experiment, the baseline CNNs did not use the attention mechanism and commonsense reasoning;

2. Datasets were derived from the original datasets from Step 1 after applying the attention mechanism. The selection of images as well as the pixels from each image was based on the target task, and the axioms and spatial relations of interest; and

3. The datasets created in Step 2 were used to train and test deep networks, wiht the results plotted as "Lenet(Att)" and "Alexnet(Att)" in Figure 4.10.

Figure 4.10 shows that using the attention mechanism and reasoning with commonsense knowledge helps achieve any desired level of accuracy

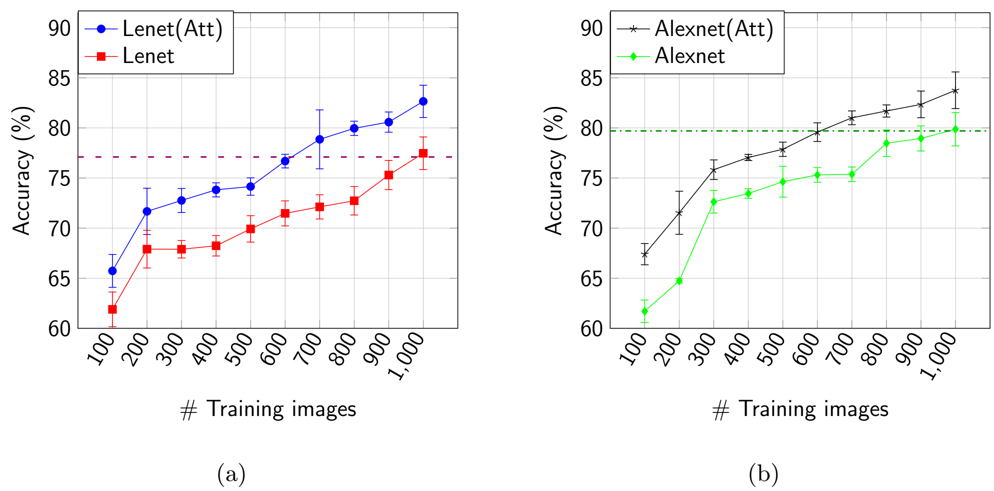


Figure 4.10: Accuracy with and without the attention mechanism and commonsense reasoning of: (a) Lenet; (b) Alexnet. The number of background images was fixed at 100. Any desired accuracy is achieved with a smaller training set when commonsense reasoning and attention mechanism are used.

with much fewer training examples. The purple dashed line in Figure 4.10(a) indicates that the baseline Lenet needs ≈ 1000 images to reach an accuracy of 77%, whereas our architecture reduces this number to ≈ 600. The experiments indicate a similar difference between the Alexnet and Alexnet(Att) for 79.5% accuracy—see the dark green dash-dotted line in Figure 4.10(b). These results indicate that the use of commonsense knowledge helps train deep networks with fewer examples, reducing both the computational requirements and storage requirements, an outcome which supports hypothesis **<u>H2</u>**.

The third set of experiments was designed as follows to evaluate the ability of learning previously unknown axioms, with results summarized in Table 4.1:

1. Ten sets of 50 labeled images were created, as described in Section 4.3;

2. The axiom learning algorithm was trained with each set three times, using thresholds of 95% and 70% at the leaf nodes of the decision trees. These are the values assigned to the threshold th_1 described in Algorithm 1 in Section 4.2.4;

3. The precision and recall for the unknown axioms, e.g., Statements 4.5(a), 4.5(b), and 4.6, but excluding defaults (i.e., with the threshold $th_1 = 95\%$), are summarized as "unknown (normal)" in Table 4.1;

4. The precision and recall for the unknown default statements, e.g., Statement 4.4, (i.e., with the threshold $th_1 = 70\%$) are summarized as "unknown (default)" in Table 4.1;

Axiom type	**Precision**	**Recall**
Unknown (normal)	98%	100%
Unknown (default)	78%	62%

Table 4.1: Precision and recall for previously unknown axioms (normal, default) using decision tree induction.

Table 4.1 demonstrates the ability to learn previously unknown axioms. Errors are predominantly variants of the target axioms that are not in the most generic form, i.e., they have irrelevant literals but are not actually wrong. The lower precision and recall with defaults are understandable because it is challenging to distinguish between defaults and their exceptions. These results support hypothesis **H3**.

The fourth set of experiments were designed as follows to evaluate hypothesis **H4** on revising and merging axioms, with results plotted in Figures 4.11 and 4.12, for normal and default axioms respectively:

1. Ten sets of 60 labeled scenes were created, as described in Section 4.3. Each set was used in one run of ensemble learning (see Algorithm 1).

2. In each cycle, 50 images were used for decision tree induction and axioms extraction (Section 4.2.4). The choice of the best version of similar axioms was based on the other 10 images.

3. The forgetting parameters (decay factor and axiom strengths) were updated, and the axioms with strength bellow 10% eliminated (orange dashed line in the figures).

4. Steps 2 and 3 were repeated for 10 learning cycles.

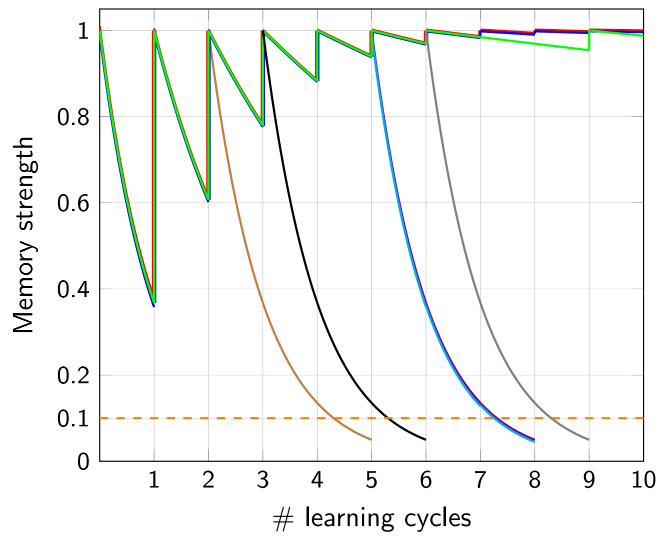


Figure 4.11: Normal axioms evolution over time.

Table 4.2: Normal axioms legend.

Color	Axiom
	holds(partially_stable(A, B), I) :- -holds(relation(above, A, B), I), -has_surface(A, irregular).
	-holds(partially_occluded(A, B), I) :- -holds(relation(behind, A, B), I).
	-holds(stable(A), I) :- holds(irregular_below(A), I).
	-holds(partially_occluded(A, B), I) :- holds(relation(above, A, B), I).
	-holds(partially_stable(A, B), I) :- holds(relation(above, A, B), I), holds(tower_height(A, N), I), N>4.
	-holds(stable(A), I) :- holds(small_base(A), I), holds(tower_height(A, N), I), N>4.
	holds(partially_stable(A, B), I) :- -holds(relation(above, A, B), I), holds(relation(behind, A, C), I).
	holds(stable(A), I) :- holds(tower_height(A, N), I), N<=1.

Figure 4.11 shows how the eight axioms in Table 4.2 behaved over 10 cycles. The top three axioms, shown in green, red and blue, are learned or reinforced in almost every cycle. Although the axioms corresponding to the green-colored plot were not re-learned in cycles 7 and 8, it was able to

maintain a high level of strength. This is because it was reinforced in all previous cycles, resulting in a small decay factor. In contrast, the other five axioms from Table 4.2 were not re-learned or frequently used, so that they were quickly removed.

Figure 4.12 shows the equivalent plot for 10 default axioms included in Table 4.3. Note that the axioms included on lines 4 and 5 in Table 4.3 (represented by the brown and lime green-colored plots in Figure 4.12) are different versions of the same axiom; the axiom on line 4 contains an extra (unnecessary) literal *(holds(obj_relation(front, A, D), I))*. The algorithm then identified the similarity, keeping the more general version of the axiom (line 5 in Table 4.3) and removing the other version (line 4 in Table 4.3) in cycle 5; this is where the brown-colored plot stops in Figure 4.12. These results support hypothesis **H4**.

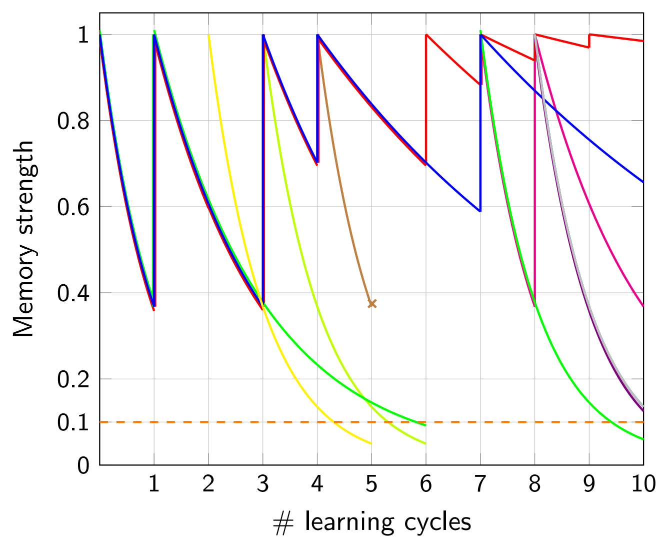


Figure 4.12: Default axioms evolution over time.

Finally, we ran experiments in which the robot computed minimal plans to pick up and clear particular objects. We observed that the number of plans computed when the learned axioms are included in the ASP program are much smaller than when the axioms are not included—this makes sense because the learned axioms are constraints that eliminate possible paths in

Table 4.3: Default axioms legend.

Color	Axioms
	-holds(stable(A), I) :- holds(small_base(A), I).
	-holds(partially_stable(A, B), I) :- holds(relation(above, A, B), I).
	-holds(stable(A), I) :- holds(tower_height(A, N), I), N>4.
	holds(partially_occluded(A, B), I) :- holds(relation(behind, A, B), I), -holds(relation(above, A, C), I), -holds(relation(front, A, D), I).
	holds(partially_occluded(A, B), I) :- holds(relation(behind, A, B), I), -holds(relation(above, A, C), I).
	-holds(partially_occluded(A, B), I) :- holds(relation(above, A, B), I).
	holds(partially_stable(A, B), I) :- -holds(relation(above, A, B), I).
	holds(partially_stable(A, B), I) :- holds(relation(behind, A, B), I).
	holds(partially_stable(A, B), I) :- holds(relation(front, A, B), I).
	holds(stable(A), I) :- -holds(small_base(A), I).

the transition diagram. For instance, the goal in one set of experiments was to clear the large red box partially hidden behind the white box and the duck in Figure 4.13. With all the axioms the robot found eight plans (all of which were correct); however, with some axioms missing, the robot found as many as 90 plans, many of which were incorrect. A plan was considered to be correct if executing it (in simulation) resulted in the corresponding goal being achieved. All these results support hypothesis **H3**.

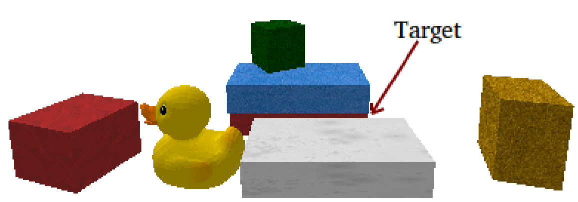


Figure 4.13: Illustrative image used for planning experiments with and without the learned axioms.

4.5 Conclusion

Deep network architectures and algorithms represent state-of-the-art technique for many tasks in robotics and AI. However, they require large training datasets and considerable computational resources, and the complex internal representations they learn make it difficult to understand their operation. The architecture described in this chapter draws inspiration from research in cognitive systems to address these limitations. It integrates the principles of non-monotonic logical reasoning with commonsense knowledge, and decision tree induction of knowledge, with deep learning. The underlying intuition is that commonsense knowledge is available in almost every application domain—in fact, some such knowledge is often necessary to optimize the parameters of deep networks. Our architecture, on the other hand, explores a sophisticated approach for fully exploiting this knowledge. Reasoning with domain knowledge simplifies learning—the robot only needs to learn about aspects of the domain not already encoded by the existing knowledge. A more accurate mapping is thus learned between the desired inputs and outputs using a smaller set of labeled examples. We have experimentally validated our intuition in the context of estimating the occlusion of objects and the stability of object structures in simulated images. Our architecture improves accuracy, and reduces storage and computation requirements, especially when large labeled training datasets are not readily available.

In the next chapter, we explore the use of this architecture on a robot manipulating tabletop objects and providing explanations for its decisions and beliefs.

Chapter 5

Reasoning with Incomplete Commonsense Knowledge for Transparent Decision Making on Robots

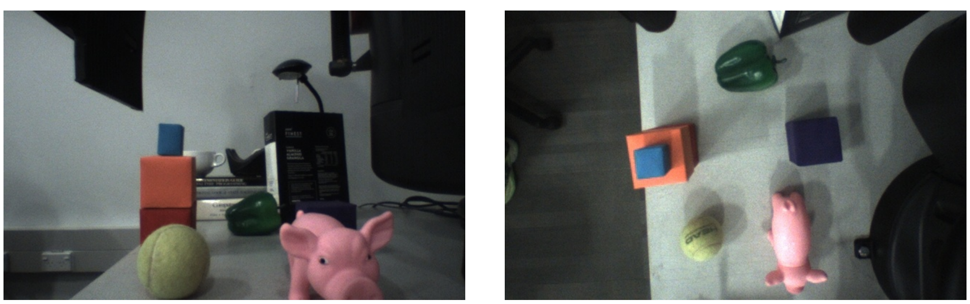
In this chapter, we seek to expand the architecture, described in Chapter 4, that tightly couples the complementary strengths of knowledge-based and data-driven algorithms in order to provide on-demand explanations of agent's decisions, beliefs, and the outcomes of hypothetical events.

5.1 Introduction

We describe changes to the architecture presented in Chapter 4 and focus primarily on the new methodology for generating explanations. We explore how the implementation of this methodology builds on the interplay between the representational choices, reasoning mechanisms, and learning algorithms in our architecture.

As a motivating example, we consider a scene understanding and manipulation task, as shown in Figure 5.1. The robot has to extract information from images captured by the on-board cameras, e.g., Figures 5.2a and b, arrange objects in different spatial configurations, and answer explanatory questions about its plans, the decisions and choices related to these plans, and the beliefs that informed these decisions. In this context, we describe an approach to automatically extract relevant information and construct answers to such questions.

Figure 5.1: Motivating scenario of a manipulator arranging objects in desired configurations on a tabletop.



(a) (b)

Figure 5.2: Camera's view extracted from Baxter's grippers: (a) left; (b) right.

This chapter is based on the conference paper accepted for publication at the 17th European Conference on Multi-agent Systems (EUMAS 2020).

5.2 Architecture

The overall architecture for representation of, explainable reasoning with, and learning from incomplete domain knowledge and noisy observations is shown in Figure 5.3. The components to the left of the dotted vertical line represent the architecture presented in Chapter 4, which used non-monotonic logical reasoning to guide deep learning for visual classification tasks. The components to the right of the dotted line are introduced in this chapter for answering explanatory questions from a human about decisions, beliefs, and hypothetical situations. These answers are constructed by automatically identifying, extracting, and analyzing relevant information. The new components (for explanatory descriptions) are described in the following domain, in which the robot Baxter executes plans as required by humans, and then provides the required explanations.

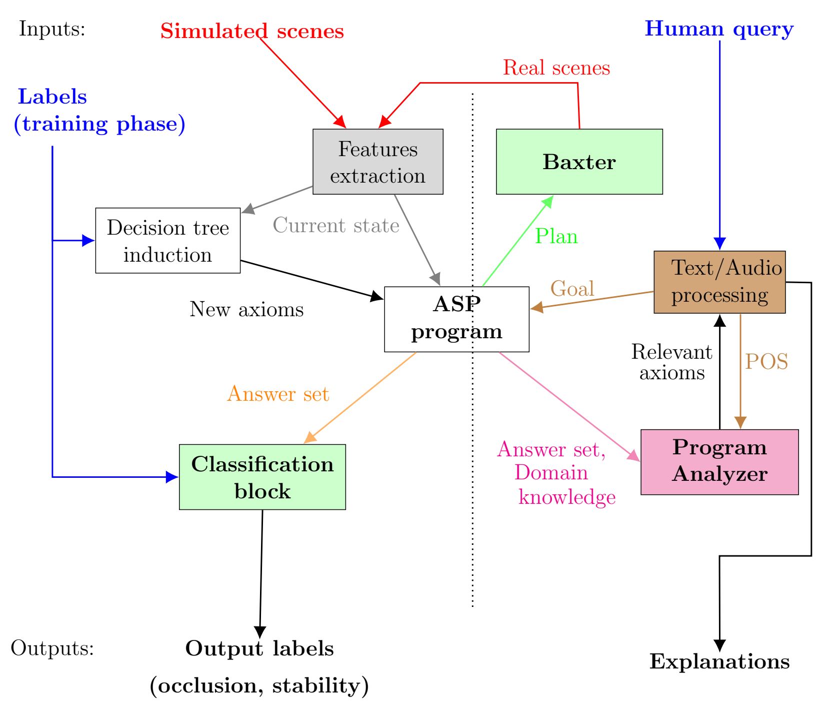


Figure 5.3: Architecture combines strengths of deep learning, non-monotonic logical reasoning with incomplete knowledge, and inductive learning. New components to the right of the dotted line support desired explainability.

Example 2. [Robot Assistant (RA)] A Baxter robot (see Fig. 5.1) analyzes scenes containing toys in different configurations. The goal is to: (i) estimate the occlusion of scene objects and the stability of structures, and rearrange the objects according to human requirements; and (ii) explain its actions and beliefs as required by humans. The domain knowledge includes object attributes such as *size* (small, medium, large), $surface$ (flat, irregular) and *shape* (cube, apple, duck), and the *relation* between objects (above, below, front, behind, right, left, close). The robot can move objects to achieve the desired goals. Knowledge also includes axioms governing domain dynamics, but some axioms may be unknown, e.g.:

- Placing an object on top of an object with an irregular surface causes instability.
- Removing all objects in front of an object causes this object to be not occluded.

The unknown axioms can be incrementally learned using decision tree induction, encoded in the robot's knowledge base, and then used for explanations and subsequent reasoning and learning. This domain differs from Example 1, described in Chapter 4, due to the use of real images extracted from the robot's cameras, instead of simulated scenes, and the inclusion of the agent's ability to explain its decisions and beliefs.

Since the components to the left of the dashed line in Figure 5.3 were presented in Chapters 3 and 4, we only describe below the new components (to the right of the line) that extend the baseline architecture to provide explanatory descriptions of decisions and beliefs. These new components enable the robot to process human verbal input and (a) extract goals that are achieved by planning and plan execution; or (b) answer explanatory questions about decisions, beliefs, or actions (executed or hypothetical). For that, the architecture follows the steps bellow:

- The human speech is converted into text using the speech transcription tool (A. Zhang 2017).

- Relevant information is extracted from the transcribed text using a part-of-speech tagger, the lemma list (Someya 1998) and their synonyms and antonyms from WordNet (G. A. Miller 1995). This information is used to determine the nature of the human request (e.g., achieve goal, or construct explanation).

- If the execution of a task is required, the relevant part-of-speech is mapped to facts from the knowledge base for setting up goals for planning. The ASP program computes a plan, which is transmitted to Baxter for execution, and the computed answer set is stored for future explanations.

- If an explanatory question is asked, the relevant information is extracted from the ASP program and answer sets (algorithm in Table 5.1). This information is used with suitable templates to provide human understandable responses that are converted to speech (Bhat 2018).

The new components are described in more details below.

5.2.1 Interaction interface and control loop

To answer explanatory questions, the robot first needs to interpret the questions correctly. A human's verbal input (of such a question) is processed using existing software. Specifically, verbal input is transcribed using speech recognition software (A. Zhang 2017), labeled using a part-of-speech tagger, normalized with the lemma list (Someya 1998), and the synonyms and antonyms for the words retrieved from WordNet (G. A. Miller 1995). The processed text helps identify the type of request, which may be the execution of a task or an explanation. In the former case, the related goal is passed to the ASP program for planning. In the latter case, the "Program Analyzer" component (described below) automatically infers and extracts the relevant literals to compose a suitable answer. These literals are placed in appropriate locations in associated generic templates for sentences, resulting in human-understandable (textual) explanations. Finally, the response is converted to synthetic speech (Bhat 2018).

Table 5.1: (**Program Analyzer**) Construct answer to input question

Input : Literal of input question; $\Pi(\mathcal{D}, \mathcal{H})$; answer templates.
Output: Answer and answer Literals.

```
// Compute answer set
AS = AnswerSet(Π)
if question = plan description then
    // Retrieve all actions from answer set
    answer_literals = Retrieve(AS, actions)
else if question = "why action X at step I?" then
    // Extract actions after step I
    next_actions = Retrieve(AS, actions for step > I)
    // Extract axioms influencing these actions
    relevant_axioms = Retrieve(Π, head = ¬ next_actions)
    // Extract relevant literals from Answer Set
    relevant_literals = Retrieve(AS, Body(relevant_axioms)
      ∈ I∧ ∉ I + 1)
    // Output literals
    answer_literals = pair(relevant_literals, next_actions)
else if question = "why not action X at step I?" then
    // Extract axioms relevant to action
    relevant_axioms = Retrieve(Π, head = ¬ occurs(X))
    // Extract relevant literals from Answer Set
    answer_literals = Retrieve(AS, Body(relevant_axioms) ∈ I∧ ∉ I + 1)
else if question = "why belief Y at step I?" then
    // Extract axioms influencing this belief
    relevant_axioms = Retrieve(Π, head = Y)
    // Extract body of axioms
    answer_literals = Recursive_Examine(AS, Body(relevant_axioms))
Construct_Answer(answer_literals, answer_templates)
```

5.2.2 Program Analyzer

The key component of the explanation generation system has to extract the relevant knowledge to address the human request or question. This is accomplished by automatically mapping human verbal input to related axioms, facts, and answer sets, which are used to construct explanatory descriptions.

The algorithm in Table 5.1 describes the approach for automatically identifying the axioms and facts related to the query from the ASP program and the corresponding answer sets. In the algorithm and description below, the questions are assumed to have been posed at time step I unless stated otherwise. The specific steps to be followed depends on the kind of query posed:

1. Answers to **plan description** queries are obtained directly from the corresponding answer sets (lines 1 - 3, Table 5.1). For any given goal, ASP-based reasoning provides a sequence of actions of the form *occurs(action1, step1), ..., occurs(actionN, stepN)*. These actions are used to construct the answer describing the plan.

2. To answer questions of the form **why action X at step I?** about action choices made at time step I, the following steps are used (lines 4 - 10, Table 5.1):

   - For each action that occurred after time step I, the robot examines relevant executability condition(s) that would prevent its execution at step I (lines 5 - 6). For instance, in Figure 5.2a, consider the execution of actions *occurs(pickup(robot, blue_block), 0)* and *occurs(pickup(robot, orange_block), 2)* at time steps 0 and 2, respectively. An executability condition related to the second pickup action would be:

   $$\neg occurs(pickup(robot, A), I) \leftarrow \\ holds(obj\_relation(below, A, B), I)$$

     which says that a *robot* can not *pickup* an object A located below an object B. Grounding the axiom for the specific blocks in the scene, we obtained the literal *obj_relation(below, orange_block, blue_block)*.

   - If the condition obtained above is present in the answer set at the time step of interest (0 in this example), and it is not present (or its negation is present) in the next step, it is considered to be the reason for executing the action (X) under consideration (line 8 in Table 5.1).

   - The desired answer then contains the future action and the condition that was modified with the execution of the action at the time step under consideration (line 9). In the illustrative example considered here, the question could be "Why did you pick up the blue block?", with the answer being "I had to pick up the orange block, and it was located below the blue block".

3. For questions about **hypothetical actions** that have not been selected for execution, the algorithm in Table 5.1 (lines 11 - 16):

   - Searches for executability conditions in the ASP program that have the hypothetical action in the head, i.e., prevent the action from being selected for execution.
   - For each executability condition, examines whether the literals in the body are satisfied in the corresponding answer set. If yes, uses them to construct the answer.

   For instance, in the scenario in Figure 5.2a, suppose that action *putdown(robot, blue_block, table)* occurred at step 1. For the question "Why did you not put down the blue cube on the tennis ball?", a related executability condition is:

   $$\neg occurs(putdown(robot,\ A,\ B), I)\ \leftarrow\ has\_surface(B,\ irregular)$$

   which says that an object cannot be placed on another object with an irregular surface. The answer set indicates that the tennis ball has an irregular surface. The response constructed is "Because the tennis ball has an irregular surface".

4. To explain **beliefs** the algorithm (lines 17 -25) searches for support axioms, in which the belief is the head and the corresponding body is satisfied in the current state. For the literals in the body, the search is repeated until no more axioms are found (lines 20 - 23), and the relevant literals are facts, statics, or actions applied in previous steps. For instance, to explain the belief that the object $ob1$ to be unstable in step I, the robot finds the support axioms:

   $$\neg holds(stable(ob1), I)\ \leftarrow holds(small\_base(ob1), I)$$

and the current belief may include that object $ob1$ is believed to have a small base. Searching for why the object $ob1$ is believed to have a small base, the follow axiom is identified:

$$holds(small\_base(A), I) \leftarrow holds(relation(below,\ B,\ A),\ I),\\ has\_size(B,\ small),\ has\_size(A,\ big)$$

Asking "why do you believe object $ob1$ is unstable?", would provide the answer "Because object $ob2$ is below object $ob1$ and $ob2$ is small".

5.2.3 Baxter platform

The component interfacing with the physical robot (Baxter) obtains sensor inputs and executes the actions from a computed plan to accomplish a required task. Unlike the Chapter 4, which only looked at scene understanding tasks (occlusion, stability), the robot now plans and executes actions to achieve desired goals. Observations with high probability, i.e., images such as Figures 5.2a and 5.2b from the cameras on the robot's grippers, are elevated to facts in the ASP program. This may result in incorrect information being added to the program, but the non-monotonic logical reasoning capability allows for elegant recovery from such errors.

5.3 Experimental setup and results

In this section, we describe the experimental setup, execution traces and the qualitative results of experimental evaluation. We focus primarily on the ability to construct appropriate answers for explanatory questions about images captured by Baxter's cameras in scenarios corresponding to the RA domain in Example 2.

5.3.1 Experimental setup

Considering that the approach proposed in this chapter attempts to address the gap 3 identified in Chapter 2, the experiments were design to test the following hypotheses:

H1 : Reasoning with incrementally learned and merged state constraints improves the quality of plans generated.

H2 : Exploiting the links between representation, reasoning, and learning provides accurate explanations of decisions and beliefs.

In our experiments, we considered four types of explanations: describing the plan; justifying the execution of an action at a given time step; justifying not choosing a hypothetical action; and explaining particular beliefs. The quality of the explanations was measured in terms of precision and recall of the literals included in the provided answer when compared with the expected response.

Experimental trials considered images from the robot's cameras and simulated scenes. Real world images contained five to seven objects characterized by different colors, textures, shapes, and sizes. The objects included cubes, a pig, a capsicum, a tennis ball, an apple, an orange, and a pot. These objects were either stacked on each other or spread on the table, as shown in Figure 5.2a. A total of 40 configurations were created, each with five different goals for planning, and four questions for each plan, resulting in a total of 200 plans and 800 questions. We also used a real-time physics engine to create 40 simulated images, each with seven to nine objects (three to five stacked and the remaining on a flat surface). Objects included cylinders, spheres, cubes, a duck, and five household objects from the Yale-CMU-Berkeley dataset (apple, pitcher, mustard bottle, mug, cracker box). Five different goals were considered for planning, with four questions for each plan, resulting in the same number of plans and questions as with real-world data.

To simulate different types of situations, we randomized the initial configuration of objects in the scene and the content of the requests from the human participant. Axioms are learned from simulated images in a training phase as described in Chapter 4. To explore the effect of the learned knowledge on constructing explanations, we ran experiments with and without some learned axioms in the knowledge base, measuring the corresponding number of optimal, sub-optimal, and incorrect plans, and the planning time. An *optimal* plan achieves the desired goal in the minimum number of steps; a *sub-optimal* plan requires more than the minimum number of steps and/or includes an exception to default knowledge (e.g., stacking an object on a smaller object); and a *wrong* plan leads to undesirable states or results in the goal not being achieved; this occurs, for example, when a related axiom is not applied because it is not known to the robot.

5.3.2 Execution Trace

Recall that the kind of literals extracted to construct explanations depends on the question posed. For instance, the description of a plan includes all the actions in the plan; the justification for an action choice examines future actions in the corresponding plan being executed, whereas questions about actions that were not executed are answered by examining beliefs that prevented these actions from being considered. Below, we examine some execution traces of constructing explanations in the context of the RA domain. The first three execution examples consider a real-world scene captured from the robot's gripper—see Figure 5.2a. The other three examples refer to an image created by the physics simulator and are shown in Figure 5.6.

Execution Example 1. [Plans, actions, and beliefs]
Consider a robot that faces the object configuration shown in Figure 5.2a. It has to move the red block on top of the orange block. The robot answers the following questions after executing a plan for this goal:

- **Human:** "Please describe your plan."

  Baxter: " I picked up the blue block. I put the blue block on the table. I picked up the orange block. I put the orange block down on the table. I picked up the red block. I put the red block on the orange block."

- The human may ask the robot to justify a particular action.

  Human: "Why did you pick up the blue block?"

  Baxter: "Because I had to pick up the red block, and the red block was below the blue block."

The constructed answer can also be used to highlight the relevant image regions that influenced this answer, as shown in Figure 5.4, aiding in better comprehension of the explanation and the corresponding decision.

- The human now may ask about particular actions that were not considered.

  Human: "Why did you not put down the orange block on the blue block?"

  Baxter: "Because the blue block is small."

- The human may also ask about particular beliefs.

  Human: "Why did you believe that the red block was below the blue block?"

  Baxter: "Because I observed the red block below the blue block in step zero."

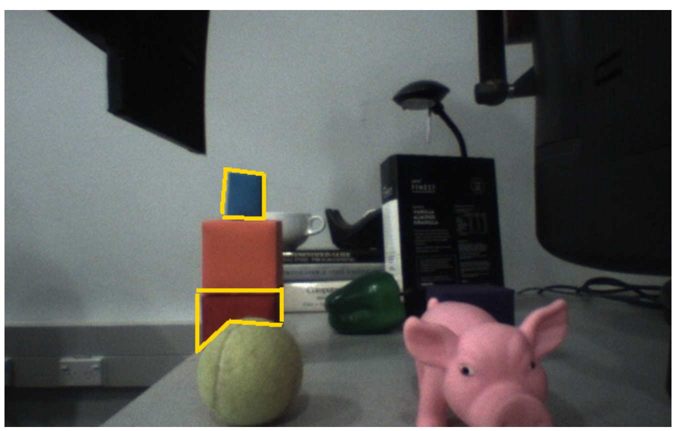


Figure 5.4: Highlighted relation between blue and red cubes, used in the explanation constructed in Execution Example 1.

Note that the answer to the question about the hypothetical action the robot did not execute considers the default knowledge learned from simulated scenes, which states that objects placed on a small base are typically unstable. The non-monotonic logical reasoning capability enables the robot to reason about exceptions to this default and recover from errors.

Execution Example 2. [Reasoning and explanation]
Continuing with our example and the scenario in Figure 5.2a, the subsequent interactions are as follows:

- **Human:** "Put the tennis ball on the blue cube."

  The goal *holds(relation(on, ball, blue_block), I)* is then encoded in the ASP program for planning. Since not trying to accomplish a goal is not an option, and the default rule about not stacking objects on small bases prevents placing objects on the blue block, the apparent inconsistency is solved by invoking an exception to the default and planning to place the ball on the top of the blue block. Then, for the following question after plan execution:

- **Human:** "Please describe the plan you executed."

  Robot: "I picked up the ball. I put down the ball on the blue block."

- The human may now explore the belief of the agent that requires it to consider exceptions to the default knowledge:

  Human: "Why do you believe the ball is on the blue block?"

  Robot: "Because I observed the ball on the blue block in step one."

Combining reasoning with constructing explanations thus allows the robot to adapt to unforeseen exceptions elegantly.

Execution Example 3. [Learning and explanation]
In some situations, the robot may be unable to accomplish the human command because it conflicts with its internal representation of the world. Even in such cases, the robot is able to answer explanatory questions. For instance, consider the scenario in Figure 5.2a, with the following interaction:

- **Human:** "Please put the pig on the ball."

  The robot does not execute this action because an incrementally learned axiom states that any object placed on another with an irregular surface will be unstable.

- However, our architecture allows an agent to provide explanations for the not execution of an action, so that:

  Human: "Why can't you put the pig on the top of the ball?"

  Robot: "Because the ball has an irregular surface."

  Once again, the answer can be used to highlight ROIs in Figure 5.5 relevant to constructing this answer.

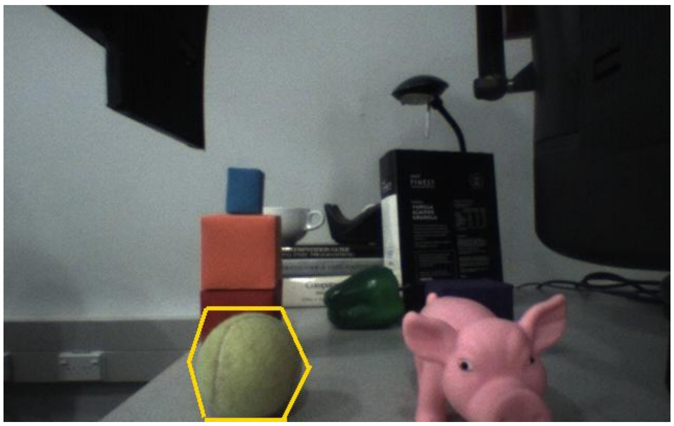


Figure 5.5: Highlighted tennis ball to be considered in the explanation provided in Execution Example 3.

This example illustrates that integrating learning with reasoning and explanation generation enables the robot to explain the newly acquired knowledge and the reason for not acting according to the human command.

For the next examples, the interactions considered the simulated scene depicted in Figure 5.6.

Execution Example 4. [Plans, actions, and beliefs]
Now the robot starts with objects as shown in Figure 5.6. It has to move the pitcher on top of the red block. The robot answers the following questions after executing a plan for this goal:

- **Human:** "Please describe your plan."

  Baxter: " I picked up the green can. I put the green can on the table. I picked up the white block. I put the white block down on the table. I picked up the pitcher. I put the pitcher on the red block."

Images/chapter5/image_00512_crop.png

Figure 5.6: Example of scene created in simulator.

- The human may ask the robot to justify a particular action.

  Human: "Why did you pick up the white block?"

  Baxter: "Because I had to put down the pitcher on the red block, and the red block was below the white block."

  As for the real-world scene, the constructed answer can also be used to highlight the relevant image regions that influenced this answer, as shown in Figure 5.7, aiding in better comprehension of the explanation and the corresponding decision.

- The human now may ask about particular actions that were not considered.

  Human: "Why did you not put down the pitcher on the green can?"

  Baxter: "Because the green can is small."

- The human may also ask about particular beliefs.

  Human: "Why did you believe that the red block was below the white block?"

Baxter: "Because I observed the red block below the white block in step zero."

Figure 5.7: Highlighted relation between white and red cubes, used in the explanation constructed in Execution Example 4.

Once again, the answer to the question about the hypothetical action the robot did not execute considers the default knowledge learned from simulated scenes, which states that objects placed on a small base are typically unstable. The non-monotonic logical reasoning capability enables the robot to reason about exceptions to this default and recover from errors.

Execution Example 5. [Reasoning and explanation]
Continuing with the example using the scenario in Figure 5.6, the subsequent interactions are:

- **Human:** "Put the mug on the green can."

  The goal *holds(relation(on, mug, green_ can), I)* is then encoded in the ASP program for planning. Since not trying to accomplish a goal is not an option, and the default rule about not stacking objects on small bases prevents placing objects on the green can, the apparent inconsistency is solved by invoking an exception to the default and planning to place the mug on the top of the green can. Then, for the following question after plan execution:

- **Human:** "Please describe the plan you executed."

  Robot: "I picked up the mug. I put down the mug on the green can."

- The human may now explore the belief of the agent that requires it to consider exceptions to the default knowledge:

Human: "Why do you believe the mug is on the green can?"

Robot: "Because I observed the mug on the green can in step one."

The construction of explanations combined with non-monotonic reasoning thus allows the robot to adapt to unforeseen exceptions elegantly.

Execution Example 6. [Learning and explanation]
In some situations, the robot may be unable to accomplish the human command because it conflicts with its internal representation of the world. Even in such cases, the robot is able to answer explanatory questions. For instance, consider the scenario in Figure 5.6, with the following interaction:

- **Human:** "Please put the pitcher on the duck."

  The robot does not execute this action because an incrementally learned axiom states that any object placed on another with an irregular surface will be unstable.

- However, our architecture allows an agent to provide explanations for the not execution of an action, so that:

  Human: "Why can't you put the pitcher on the top of the duck?"

  Robot: "Because the duck has an irregular surface."

  Once again, the architecture is able to use the answer to highlight ROIs in Figure 5.8 relevant to constructing this answer.

Figure 5.8: Highlighted duck to be considered in the explanation provided in Execution Example 6.

This is another example of how integrating learning with reasoning and explanation generation enables the robot to explain the newly acquired knowledge.

Table 5.2: Values of planning measures with the learned axioms expressed as a fraction of the values of these measures without the learned axioms in each paired trial.

Measures	**Ratio (with/without)**	
	Real scenes	**Simulated scenes**
Number of plans	0.36	0.33
Optimal	1	1
Sub-optimal	0.43	0.55
Incorrect	0	0
Time	0.66	0.89

5.3.3 Experimental results

The first set of experiments was designed as follows with results summarized in Table 5.2:

1. Forty initial object configurations were arranged (similar to that in Figure 5.1). The robot (Baxter) automatically extracted information (attributes, spatial relations) from images corresponding to top and frontal views (using the cameras on the left and right grippers) and encoded it in the ASP program as the initial state.

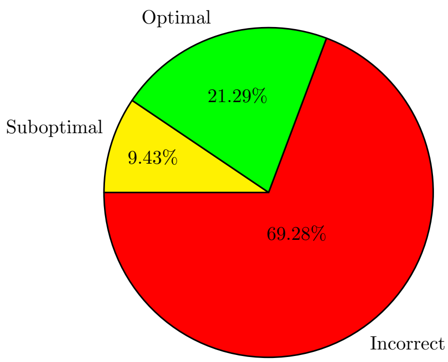
2. For each initial state, five goals were randomly chosen to be encoded in the ASP program. The robot reasoned with the existing knowledge to create plans for these 200 combinations (40 initial states, five goals).

3. The plans were evaluated in terms of the number of optimal, sub-optimal and incorrect plans, and planning time.

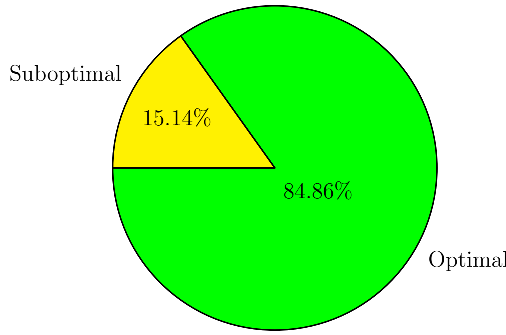
4. Experiments were repeated with and without the learned axioms, with the results for the former computed as a fraction of those for the latter.

5. Steps 1 to 4 were repeated for simulated scenes. Table 5.2 summarizes the average results.

Since the quality measure of each plan depends on the initial conditions and the goal, we conducted paired trials with and without the learned constraints included in reasoning. The initial conditions and goals/tasks were identical within each paired trial, and differed between different paired trials. We computed the value of each planning measure with the learned constraints as a fraction of the value without the learned constraints. The average of these ratios over all the trials is in Table 5.2.

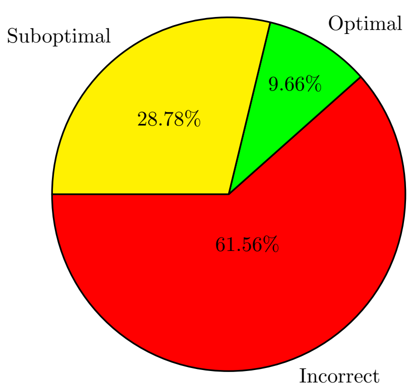


(a) Without learning.

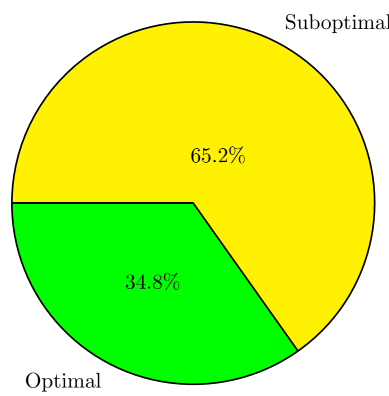


(b) With learning

Figure 5.9: Percentage distribution of optimal, sub-optimal and incorrect plans created for **real scenes** when (a) no axioms are learned and (b) new axioms are learned.



(a) Without learning.



(b) With learning

Figure 5.10: Percentage distribution of optimal, sub-optimal and incorrect plans created for **simulated scenes** when (a) no axioms are learned and (b) new axioms are learned.

Computing plans by reasoning with the learned axioms significantly reduces the total number of plans (to 36% for real scenes and 33% for simulated scenes), and the number of sub-optimal plans (to 43% and 55% for real and simulated scenes respectively), removing all incorrect plans and keeping all the optimal ones. When the learned axioms are used for reasoning, sub-optimal plans were created only when the assigned goal could only be achieved by creating an exception to a default, e.g., stacking an object on a small base. Also, the inclusion of extra axioms reduced the computation time (for planning) by 35% for real images and 11% using simulations; the ASP program without the learned axioms computed almost three times the number of plans computed with the learned axioms.

In addition, among the plans computed for different real-world scenes, $\approx 85\%$ were optimal and $\approx 15\%$ were suboptimal when the robot reasoned with the learned constraints; without the learned constraints, $\approx 21\%$, $\approx 9\%$ and $\approx 70\%$ of the computed plans were optimal, suboptimal, and incorrect respectively, as shown in Figure 5.9. The equivalent distribution for simulated scenes is shown in Figure 5.10. Furthermore, the inclusion of learned constraints significantly reduced the search space and thus the planning time in comparison with when the learned axioms are not included. These results indicate that the knowledge acquired over time improves the overall quality of plans, which supports hypothesis **H1**.

The second set of experiments was designed as follows with results summarized in Table 5.3:

1. For each of the 200 combinations (40 configurations, five goals) from the first set of experiments, considering (once again) knowledge bases with and without the learned axioms.

2. The robot first had to describe the plan and justify the choice of a randomly-chosen action in the plan. Then, one parameter of the chosen action was changed randomly to pose a question about why this new action could not be applied. Finally, one of the beliefs related to the previous two questions had to be justified.

3. The literals present in the answers were compared against the expected literals in the ideal response, with the average precision and recall scores shown in Table 5.3.

Table 5.3: Precision and recall of retrieving relevant literals for constructing answers to explanatory questions, with and without the learned axioms used for reasoning. These results refer to **Real scenes** from the robot's gripper.

	Precision		**Recall**	
Query Type	**Without**	**With**	**Without**	**With**
Plan description	91.77%	100%	91.77%	100%
Why X?	91.75%	94.75%	91.98%	94.75%
Why not X?	93.57%	95.16%	87.91%	98.88%
Belief	93.04%	99.35%	93.63%	100%

Table 5.4: Precision and recall of retrieving relevant literals for constructing answers to questions, with and without reasoning with learned axioms. These results refer to **Simulated scenes**.

	Precision		**Recall**	
Query Type	**Without**	**With**	**Without**	**With**
Plan description	90.04%	100%	90.04%	100%
Why X?	93.0%	93.0%	93.0%	93.0%
Why not X?	93.22%	100%	89.43%	98.04%
Belief	97.22%	99.19%	97.9%	100%

4. The same experiments were separately conducted for simulated scenes, with results summarized in Table 5.4.

Tables 5.3 and 5.4 show that when the learned axioms are used for reasoning, the precision and recall of relevant literals (for constructing the explanation) are higher than when the learned axioms are not included. This is especially true when the robot has to answer questions about the execution of actions that it has not actually executed. Note that the precision and recall rates are reasonable even when the learned axioms are not included; this is because not all the learned axioms are needed to accurately answer each explanatory question. When the learned axioms are included in the reasoning, errors are very rare and correspond to some additional literals being included in the answer (i.e., over-specification). In addition, when we specifically remove axioms related to the goal under consideration, precision and recall values are much lower. Furthermore, there is noise in both sensing and actuation, especially with robot experiments. For instance, recognition of spatial relations, learning of state constraints, and manipulation have approximate error rates of 15%, $5-10\%$, and 15% respectively. Experimental results thus indicate that coupling reasoning, learning, and explanation allows the robot to provide accurate relational descriptions in response to questions about decisions, beliefs, and hypothetical events, which supports hypothesis **H2**. Additional examples of images, questions, and answers are in our repository (Mota and Sridharan 2020b).

5.4 Conclusions

With the increasing use of sophisticated AI and ML algorithms for different tasks in robotics and AI, there is considerable interest in understanding the operation of these algorithms. In this chapter, we have focused on integrated robot systems that represent, reason with, and learn from incomplete domain knowledge and noisy data. For such a system, we have presented an approach for reliably and efficiently answering explanatory questions about the plans, decisions, and beliefs. We have implemented this approach in the architecture, described in Chapter 4, that combines the complementary strengths of non-monotonic logical reasoning with incomplete domain knowledge, deep learning, and inductive learning. In the context of a robot sensing and manipulating objects, we have demonstrated that the ability of our approach to exploit the interplay between knowledge-based reasoning and data-driven learning enables the approach to reliably answer explanatory questions. In addition, the architecture is able to make good use of the learned knowledge to further improve its performance.

Chapter 6

Conclusion and Future Work

6.1 Conclusion

In this thesis, we presented a system that integrates the grounding of spatial relations between objects, relational representation of knowledge, commonsense reasoning, incremental induction of knowledge, and data-driven learning for scene understanding tasks. We attempted to fill in the gaps of the state-of-the-art algorithms in computer vision by exploring non-monotonic logical reasoning (and representation) with incomplete commonsense knowledge. Our main goal was equipping agents with the ability to learn incrementally from a small number of samples.

As we discussed throughout this thesis, deep learning and associated algorithms have achieved outstanding results in AI and robotics in recent years. However, the need for a large amount of training data limits the application of deep architectures, and the difficulty in interpreting the internal representations they learn imposes a serious barrier to its adoption in many critical areas, such as medical diagnosis and autonomous vehicles. These limitations have motivated recent studies trying to reduce the training effort, e.g., by using preexisting knowledge to simplify learning, and to provide any explanation for deep architecture decisions. We explored the relational representation, non-monotonic reasoning with commonsense knowledge, and induction of previously unknown state constraints for incrementally learning the knowledge freely available in the domain, and efficiently training deep architectures. Explanations are then obtained from the relational description of knowledge and the answer set provided by the reasoner.

The combination of a manually encoded (QSR) and an incrementally updated (MSR) representation of spatial relations between objects, along with a symbolic relational representation of knowledge, produced reliable estimates of these relations. While the QSR component encodes the initial knowledge about spatial relations, the MSR allows the adaptation to specific characteristics of the domain, as well as the use of human feedback if available. The non-monotonic reasoning and the relational representation of knowledge in ASP enables the combination of basic prepositions in more elaborate relations, and allows elegant recovery from errors caused by reasoning with incomplete knowledge.

The learning model explores the complementary strengths of incremental induction of knowledge, non-monotonic logical reasoning, and deep learning for scene understanding tasks. Specifically, the induction of decision trees, using the groundings of spatial relations and image labels, discovers unknown axioms available in the domain. These axioms are encoded in ASP for reasoning with new scenes, and the resulting program is used for efficiently training Convolutional Neural Networks. These CNNs deal only with images that the reasoner is unable to classify. As a result, this model uses the knowledge available in the domain to reduce the effort for training deep architectures.

Explanatory descriptions of agent's decisions and beliefs are then based on the incrementally updated relational representation of knowledge and the answer set computed by the reasoner. The symbolic relational representation in ASP facilitates the retrieval of knowledge required to compose the responses. Since our architecture learns (i.e., using decision tree induction) such knowledge from the same data used for constructing deep networks, the relational descriptions provided as explanations may also be considered as descriptions of the observed behavior of these networks, providing insights about their internal operation. In addition, our architecture allows the interleaving between learning from simulated images and applying this knowledge in the real world. In the experiments, unknown state constraints were learned from scenes created by simulators, whereas the explanations also consider real world scenes extracted from the robot Baxter's onboard cameras.

In summary, the research presented in this thesis resulted in three main contributions:

- A grounding model for representation of spatial relations between domain objects able to learn incrementally and interactively from experiences.
- An approach that combines the complementary strengths of decision tree induction, non-monotonic logic reasoning, and deep learning for scene understanding tasks that reduces the complexity for training deep architectures by considering the knowledge available in the domain.
- A method that explores the relational description of incrementally updated knowledge to facilitate the retrieval of the information required for answering decision making related queries, improving the quality of explanations for agent's decisions, beliefs and experiences.

6.2 Future work

The work presented in this thesis opens up multiple directions for future research. First, the interplay between reasoning and learning could be further explored to understanding the behavior of deep network models. We already showed how the training of deep networks can be optimized, and the same samples used to train these networks can also be used to induct previously unknown axioms that help understand their behavior. The approach presented here could be further explored to incrementally understand the behavior of different deep network architectures in terms of axioms. Although we explored Convolutional Neural Networks (CNN) applied the specific task of physics understanding from visual scenes, the approach can be extended to many other applications involving deep architectures, e.g., Visual Question Answering (VQA), Recommendation Systems, boardgames, which may also employ different types of deep architectures, such as Recursive Neural Networks (RNN) and Feedforward Neural Networks (FNN).

The architecture presented in this thesis uses decision tree induction in order to discover state constraints of a domain that includes an agent manipulating tabletop objects. The learning capabilities of our architecture can be expanded to learn different types of domain knowledge, such as causal laws and executability conditions. This in turn could enhance the architecture's ability of reasoning and of better exploring the behavior of deep network models. Our architecture could also be explored in more complex domains, e.g, robots moving between different rooms in an office, tasked with finding and delivering objects to humans or other robots. Such a domain imposes some challenging problems, such as multi-agent planning and collective learning, in which simultaneous execution of actions are possible and the learned knowledge have to be shared among agents.

In addition, the architecture's ability to explain decisions, beliefs and experiences based on the declarative representation of knowledge would be expanded to accommodate further interactions with humans requiring the adjustment of these descriptions to different needs. The architecture described in this thesis assumes a perfect communication between humans and robots, in which agents would perfectly understand humans' questions and humans would be always satisfied with robots' responses. However, ambiguous questions and unsatisfactory answers are likely to occur in human-robot interactions. Future work could expand our architecture to construct clarification queries for ambiguous questions, and to restructure the description provided by the agent in order to express different levels of abstraction and detail as required by humans.

Bibliography

Abadi, M et al. (2016). "TensorFlow: large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467*. URL: `https://arxiv.org/abs/1603.04467`.

Anjomshoae, Sule et al. (2019). "Explainable agents and robots: Results from a systematic literature review". In: *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. Montreal, Canada.

Baillargeon, Renée (2002). "The acquisition of physical knowledge in infancy: A summary in eight lessons". In: *Blackwell handbook of childhood cognitive development*. Leslie Pieraut-Le Bonniec Spelke & Kestenbaum, pp. 47–83.

Balai, Evgenii, Michael Gelfond, and Yuanlin Zhang (Sept. 2013). "Towards Answer Set Programming with Sorts". In: *International Conference on Logic Programming and Nonmonotonic Reasoning*. Corunna, Spain.

Balduccini, Marcello (2007). "Learning Action Descriptions with A-Prolog: Action Language C." In: *AAAI Spring Symposium: Logical Formalizations of Commonsense Reasoning*, pp. 13–18.

Bates, Christopher et al. (2015). "Humans predict liquid dynamics using probabilistic simulation." In: *CogSci*.

Battaglia, Peter W, Jessica B Hamrick, and Joshua B Tenenbaum (2013). "Simulation as an engine of physical scene understanding". In: *Proceedings of the National Academy of Sciences* 110.45, pp. 18327–18332. ISSN: 0027-8424. DOI: `10.1073/pnas.1306572110`. eprint: `arXiv:1404.2263v1`. URL: `http://www.pnas.org/content/110/45/18327.full.pdf%20http://www.pnas.org/cgi/doi/10.1073/pnas.1306572110`.

Bay, Herbert et al. (2008). "Speeded-up robust features (SURF)". In: *Computer vision and image understanding* 110.3, pp. 346–359.

Belz, Anja et al. (Sept. 2015). "Describing Spatial Relationships between Objects in Images in English and French". In: *Proceedings of the 2015 Workshop on Vision and Language* 15, pp. 104–113.

Besold, Tarek R et al. (2017). "Neural-symbolic learning and reasoning: A survey and interpretation". In: *arXiv preprint arXiv:1711.03902*.

Bhat, Natesh M (2018). *pyttsx3 - Text-to-speech x-platform*. https://github.com/nateshmbhat/pyttsx3.

Bonnie, M (1996). "Muir and Neville Moray. Trust in automation. part ii. experimental studies of trust and human intervention in a process control simulation". In: *Ergonomics* 39.3, pp. 429–460.

Borgo, Rita, Michael Cashmore, and Daniele Magazzeni (2018). "Towards Providing Explanations for AI Planner Decisions". In: *IJCAI Workshop on Explainable Artificial Intelligence*, pp. 11–17.

Cadene, Remi et al. (2019). "Murel: Multimodal relational reasoning for visual question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1989–1998. URL: http://openaccess.thecvf.com/content_CVPR_2019/papers/Cadene_MUREL_Multimodal_Relational_Reasoning_for_Visual_Question_Answering_CVPR_2019_paper.pdf.

Calli, Berk et al. (2015). "Benchmarking in Manipulation Research". In: *IEEE Robotics and Automation Magazine* September, pp. 36–52. DOI: 10.1109/MRA.2015.2448951. arXiv: 1502.03143.

Cashmore, Michael et al. (2015). "ROSPlan: Planning in the Robot Operating System." In: *ICAPS*, pp. 333–341.

Cave, Carolyn Backer and Stephen M Kosslyn (1993). "The role of parts and spatial relations in object identification". In: *Perception* 22.2, pp. 229–248.

Cawsey, Alison (1993). "Planning interactive explanations". In: *International Journal of Man-Machine Studies* 38.2, pp. 169–199.

Certicky, M (2012). "Action learning with reactive answer set programming: Preliminary report". In: *ICAS 2012, The eighth international conference on autonomic and autonomous systems*, pp. 107–111.

Chai, J. Y. et al. (July 2018). “Language to Action: Towards Interactive Task Learning with Physical Agents”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. Stockholm, Sweden. URL: `https://www.ijcai.org/proceedings/2018/0001.pdf`.

Chakraborti, Tathagata et al. (2017). “Plan Explanations as Model Reconciliation: Moving Beyond Explanation as Soliloquy”. In: *International Joint Conference on Artificial Intelligence*, pp. 156–163.

Chen, Juan et al. (2015). “A survey of qualitative spatial representations”. In: *The Knowledge Engineering Review* 30.1, pp. 106–136.

Chen, Weidong, Terrance Swift, and David S Warren (1995). “Efficient top-down computation of queries under the well-founded semantics”. In: *The Journal of logic programming* 24.3, pp. 161–199.

Chen, Yilun et al. (2020). “DSGN: Deep Stereo Geometry Network for 3D Object Detection”. In: *arXiv preprint arXiv:2001.03398*. URL: `https://arxiv.org/pdf/2001.03398.pdf`.

Clementini, Eliseo (2019). “A Conceptual Framework for Modelling Spatial Relations”. In: *Information Technology And Control* 48.1, pp. 5–17.

Cohn, Anthony G et al. (1997). “Qualitative spatial representation and reasoning with the region connection calculus”. In: *GeoInformatica* 1.3, pp. 275–316.

Dalal, Navneet and Bill Triggs (2005). “Histograms of oriented gradients for human detection”. In: *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. Vol. 1. IEEE, pp. 886–893.

Dannenhauer, Dustin et al. (2018). “Explaining Rebel Behavior in Goal Reasoning Agents”. In: *ICAPS Workshop on Explainable Planning*, pp. 12–18.

David, Hume and Beauchamp Tom (2000). “An Enquiry concerning Human Understanding: A Critical Edition”. In:

Davis, Ernest and Gary Marcus (2016). “The scope and limits of simulation in automated reasoning”. In: *Artificial Intelligence* 233, pp. 60–72. DOI: `10.1016/j.artint.2015.12.003`. URL: `www.elsevier.com/locate/artint`.

Dobnik, Simon, Mehdi Ghanimifard, and John Kelleher (2018). "Exploring the functional and geometric bias of spatial relations using neural language models". In: *Proceedings of the Combined Workshop on Spatial Language Understanding (SpLU) and Grounded Communication for Robotics (RoboNLP)*, pp. 1–11.

Elliott, Desmond and Arjen P De Vries (2015). "Describing Images using Inferred Visual Dependency Representations". In: *Acl*, pp. 42–52.

Erdem, Esra, Michael Gelfond, and Nicola Leone (2016). "Applications of Answer Set Programming." In: *AI Magazine* 37.3.

Erdem, Esra and Volkan Patoglu (2012). "Applications of action languages in cognitive robotics". In: *Correct Reasoning*. Springer, pp. 229–246.

Evans, Richard and Edward Grefenstette (2018). "Learning explanatory rules from noisy data". In: *Journal of Artificial Intelligence Research* 61, pp. 1–64.

Falkner, Andreas et al. (2018). "Industrial applications of answer set programming". In: *KI-Künstliche Intelligenz* 32.2-3, pp. 165–176.

Felzenszwalb, Pedro F et al. (2010). "Object detection with discriminatively trained part-based models". In: *IEEE transactions on pattern analysis and machine intelligence* 32.9, pp. 1627–1645.

Fichtl, Severin et al. (2015). "Using Relational Histogram Features and Action Labelled Data to Learn Preconditions for Means-End Actions". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (Workshop on Sensorimotor Contingencies for Robotics)*. URL: `http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.716.641%7B%5C&%7Drep=rep1%7B%5C&%7Dtype=pdf`.

Fragkiadaki, Katerina et al. (2015). "Learning visual predictive models of physics for playing billiards". In: *arXiv preprint arXiv:1511.07404*.

Friedman, Michael (1974). "Explanation and scientific understanding". In: *Journal of Philosophy* 71.1, pp. 5–19.

Garcez, Artur d'Avila et al. (2019). "Neural-symbolic computing: An effective methodology for principled integration of machine learning and reasoning". In: *arXiv preprint arXiv:1905.06088*.

Garcez, Artur S d'Avila, Luis C Lamb, and Dov M Gabbay (2007). "Connectionist modal logic: Representing modalities in neural networks". In: *Theoretical Computer Science* 371.1-2, pp. 34–53.

Gatsoulis, Yiannis et al. (2016). "QSRlib: a software library for online acquisition of Qualitative Spatial Relations from Video". In:

Gelfond, Michael and Yulia Kahl (2014). *Knowledge Representation, Reasoning and the Design of Intelligent Agents*. Cambridge University Press.

Gençay, Eray, Peter Schüller, and Esra Erdem (2019). "Applications of nonmonotonic reasoning to automotive product configuration using answer set programming". In: *Journal of Intelligent Manufacturing* 30.3, pp. 1407–1422.

Gil, Yolanda (1994). "Learning by experimentation: Incremental refinement of incomplete planning domains". In: *Machine Learning Proceedings 1994*. Elsevier, pp. 87–95.

Graves, Alex, Greg Wayne, and Ivo Danihelka (2014). "Neural turing machines". In: *arXiv preprint arXiv:1410.5401*.

Grice, H Paul (1975). "Logic and Conversation". In: *Syntax and semantics*. Ed. by P. Cole and J. Morgan. New York: Academic Press, pp. 41–58.

Guadarrama, Sergio et al. (2013). "Grounding spatial relations for human-robot interaction". In: *IEEE International Conference on Intelligent Robots and Systems*, pp. 1640–1647. ISSN: 21530858. DOI: `10.1109/IROS.2013.6696569`.

Guillame-Bert, Mathieu, Krysia Broda, and Artur d'Avila Garcez (2010). "First-order logic learning in artificial neural networks". In: *The 2010 International Joint Conference on Neural Networks (IJCNN)*. IEEE, pp. 1–8.

He, Kaiming et al. (2015). "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification". In: *Proceedings of the IEEE International Conference on Computer Vision* 2015 International Conference on Computer Vision, ICCV 2015, pp. 1026–1034. ISSN: 15505499. DOI: `10.1109/ICCV.2015.123`. arXiv: `1502.01852`.

Hendricks, Lisa Anne et al. (2016). "Generating visual explanations". In: *European Conference on Computer Vision*. Springer, pp. 3–19.

Jégou, Hervé and Andrew Zisserman (2014). "Triangulation embedding and democratic aggregation for image search". In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE, pp. 3310–3317.

Joulin, Armand and Tomas Mikolov (2015). "Inferring algorithmic patterns with stack-augmented recurrent nets". In: *Advances in neural information processing systems*, pp. 190–198.

Jund, Philipp et al. (2018). "Optimization Beyond the Convolution: Generalizing Spatial Relations with End-to-End Metric Learning". In: *ICRA*.

Kingma, Diederik P and Jimmy Ba (2014). "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980*. URL: `https://arxiv.org/abs/1412.6980`.

Kirk, James R and John E Laird (2019). "Learning hierarchical symbolic representations to support interactive task learning and knowledge transfer". In: *Proceedings of the 28th International Joint Conference on Artificial Intelligence*. AAAI Press, pp. 6095–6102.

Kleer, J. de, A. K. Mackworth, and R. Reiter (1992). "Characterizing Diagnoses and Systems". In: *Artificial Intelligence* 56.2-3, pp. 197–222.

Koh, Pang Wei and Percy Liang (Aug. 2017). "Understanding Black-box Predictions via Influence Functions". In: *International Conference on Machine Learning (ICML)*. Sydney, Australia, pp. 1885–1894.

Kootbally, Zeid et al. (2015). "Towards robust assembly with knowledge representation for the planning domain definition language (PDDL)". In: *Robotics and Computer-Integrated Manufacturing* 33, pp. 42–55.

Kootstra, Gert, Niklas Bergström, and Danica Kragic (2010). "Fast and automatic detection and segmentation of unknown objects". In: *Humanoid Robots (Humanoids), 2010 10th IEEE-RAS International Conference on*. IEEE, pp. 442–447.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*, pp. 1097–1105.

Krueger, Volker et al. (2019). "Testing the vertical and cyber-physical integration of cognitive robots in manufacturing". In: *Robotics and Computer-Integrated Manufacturing* 57, pp. 213–229.

Kulesza, Todd et al. (2013). "Too much, too little, or just right? Ways explanations impact end users' mental models". In: *2013 IEEE Symposium on Visual Languages and Human Centric Computing*. IEEE, pp. 3–10.

Laeng, Bruno, Christopher F Chabris, and Stephen M Kosslyn (2004). "Asymmetries in Encoding Spatial Relations". In: *The asymmetrical brain*, p. 303.

Laeng, Bruno, Jinesh Shah, and Stephen Kosslyn (1999). "Identifying objects in conventional and contorted poses: Contributions of hemisphere-specific mechanisms". In: *Cognition* 70.1, pp. 53–85.

Laird, John E. et al. (2017). "Interactive Task Learning". In: *IEEE Intelligent Systems* 32.4, pp. 6–21. URL: https://ieeexplore.ieee.org/abstract/document/8012335.

Landsiedel, Christian et al. (2017). "A review of spatial reasoning and interaction for real-world robotics". In: *Advanced Robotics* 31.5, pp. 222–242.

Law, Mark, Alessandra Russo, and Krysia Broda (2018). "The complexity and generality of learning answer set programs". In: *Artificial Intelligence*.

LeCun, Yann et al. (1998). "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11, pp. 2278–2324.

Lerer, Adam, Sam Gross, and Rob Fergus (2016). "Learning Physical Intuition of Block Towers by Example". In: arXiv: 1603.01312. URL: https://arxiv.org/pdf/1603.01312.pdf%20http://arxiv.org/abs/1603.01312.

Lewandowsky, Stephan, Michael Mundy, and Gerard Tan (2000). "The dynamics of trust: Comparing humans to automation." In: *Journal of Experimental Psychology: Applied* 6.2, p. 104.

Lewis, Michael (1998). "Designing for human-agent interaction". In: *AI Magazine* 19.2, pp. 67–67.

Li, Jianguo and Yimin Zhang (2013). "Learning surf cascade for fast and accurate object detection". In: *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*. IEEE, pp. 3468–3475.

Li, Wenbin, Mario Fritz, and Aleš Leonardis (2016). "Visual Stability Prediction and Its Application to Manipulation". In: *Proceedings of the*

Intuitive Physics Workshop at NIPS 2016. arXiv: 1609.04861. URL: https://arxiv.org/pdf/1609.04861.pdf%20http://arxiv.org/abs/1609.04861.

Liu, Shusen et al. (2019). "Generative counterfactual introspection for explainable deep learning". In: *arXiv preprint arXiv:1907.03077*.

Lowe, David G (1999). "Object recognition from local scale-invariant features". In: *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*. Vol. 2. Ieee, pp. 1150–1157.

Lutter, Michael, Christian Ritter, and Jan Peters (2019). "Deep lagrangian networks: Using physics as model prior for deep learning". In: *International Conference on Learning Representations*.

McCarthy, John (1980). "Circumscription—a form of non-monotonic reasoning". In: *Artificial intelligence* 13.1-2, pp. 27–39.

McDermott, Drew et al. (1998). "PDDL-the planning domain definition language". In: *Technical Report, Available at: www.cs.yale.edu/homes/dvm*.

Mees, Oier et al. (2017). "Metric Learning for Generalizing Spatial Relations to New Objects". In: *arXiv preprint arXiv:1703.01946*. arXiv: 1703.01946. URL: https://arxiv.org/pdf/1703.01946.pdf%20http://arxiv.org/abs/1703.01946.

Miller, George A (1995). "WordNet: a lexical database for English". In: *Communications of the ACM* 38.11, pp. 39–41.

Miller, Tim (Feb. 2019). "Explanations in Artificial Intelligence: Insights from the Social Sciences". In: *Artificial Intelligence* 267, pp. 1–38.

Moore, Johanna D and Cécile L Paris (1993). "Planning text for advisory dialogues: Capturing intentional and rhetorical information". In: *Computational linguistics* 19.4, pp. 651–694.

Moratz, Reinhard, Bernhard Nebel, and Christian Freksa (2003). "Qualitative Spatial Reasoning about Relative Position". In: *Spatial Cognition III*, p. 1034. ISSN: 1045926X. DOI: 10.1016/j.jvlc.2006.11.001.

Mota, Tiago and Mohan Sridharan (2018). "Incrementally Grounding Expressions for Spatial Relations between Objects". In: *International Joint Conference on Artificial Intelligence*, pp. 1928–1934. URL: https://www.ijcai.org/proceedings/2018/0266.pdf.

Mota, Tiago and Mohan Sridharan (June 2019a). "Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots". In: *Robotics Science and Systems*. Freiburg, Germany.

— (2019b). *Software related to the thesis*. https://github.com/tmot987/Scenes-Understanding.

— (2020a). *Dateset related to the thesis*. https://drive.google.com/drive/folders/1t9KmxHx9STsKji7H2O4hpQf4a3CQrl2D?usp=sharing.

— (2020b). *Software related to the thesis*. https://github.com/tmot987/Scenes-Understanding/tree/master/Explanations.

Mottaghi, Roozbeh et al. (2016). ""What happens if..." Learning to Predict the Effect of Forces in Images". In: *European Conference on Computer Vision*. Springer, pp. 269–285.

Neelakantan, Arvind, Quoc V Le, and Ilya Sutskever (2015). "Neural programmer: Inducing latent programs with gradient descent". In: *arXiv preprint arXiv:1511.04834*.

Paul, Rohan et al. (2018). "Efficient grounding of abstract spatial concepts for natural language interaction with robot platforms". In: *The International Journal of Robotics Research* 37.10, pp. 1269–1299.

Penning, H Leo H de et al. (2011). "A neural-symbolic cognitive agent for online learning and reasoning". In: *Twenty-Second International Joint Conference on Artificial Intelligence*.

Potapova, Ekaterina et al. (2014). "Attention-driven object detection and segmentation of cluttered table scenes using 2.5 d symmetry". In: *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, pp. 4946–4952.

Pronobis, Andrzej and Rajesh Rao (July 2017). "Learning Deep Generative Spatial Models for Mobile Robots". In: *RSS Workshop on Spatial-Semantic Representations in Robotics*. Cambridge, USA.

Purushwalkam, Senthil et al. (2019). "Bounce and Learn: Modeling Scene Dynamics with Real-World Bounces". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=BJxssoA5KX.

Rashtchian, Cyrus et al. (2010). "Collecting Image Annotations Using Amazon's Mechanical Turk". In: pp. 139–147.

Read, Stephen J and Amy Marcus-Newhall (1993). "Explanatory coherence in social explanations: A parallel distributed processing account." In: *Journal of Personality and Social Psychology* 65.3, p. 429.

Redmon, Joseph et al. (2016). "You only look once: Unified, real-time object detection". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788.

Reiter, Raymond (1980). "A logic for default reasoning". In: *Artificial intelligence* 13.1-2, pp. 81–132.

— (1987). "A Theory of Diagnosis from First Principles". In: *Artificial Intelligence* 32, pp. 57–95.

Renz, Jochen (2002). *Qualitative spatial reasoning with topological information*. Springer-Verlag.

Ribeiro, Marco, Sameer Singh, and Carlos Guestrin (Aug. 2016). "Why Should I Trust You? Explaining the Predictions of Any Classifier". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. San Francisco, USA, pp. 1135–1144.

Rusu, Radu Bogdan (2010). "Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments". In: *KI - Kunstliche Intelligenz* 24.4, pp. 345–348. ISSN: 09331875. DOI: `10.1007/s13218-010-0059-6`. arXiv: `arXiv:1011.1669v3`. URL: `http://link.springer.com/10.1007/s13218-010-0059-6`.

Samek, Wojciech, Thomas Wiegand, and Klaus-Robert Müller (2017). "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models". In: *arXiv preprint arXiv:1708.08296*.

Sanborn, Adam N, Vikash K Mansinghka, and Thomas L Griffiths (2013). "Reconciling intuitive physics and Newtonian mechanics for colliding objects." In: *Psychological review* 120.2, p. 411.

Santoro, Adam et al. (2017). "A simple neural network module for relational reasoning". In: *Advances in neural information processing systems*, pp. 4967–4976. URL: `http://papers.nips.cc/paper/7082-a-simple-neural-network-module-for-relational-reasoning`.

Seegebarth, Bastian et al. (2012). “Making hybrid plans more clear to human users-a formal approach for generating sound explanations”. In: *Twenty-Second International Conference on Automated Planning and Scheduling*.

Sermanet, Pierre et al. (2013). “Overfeat: Integrated recognition, localization and detection using convolutional networks”. In: *arXiv preprint arXiv:1312.6229*.

Shi, Miaojing, Yannis Avrithis, and Hervé Jégou (2015). “Early burst detection for memory-efficient image retrieval”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 605–613.

Shridhar, Mohit and David Hsu (July 2017). “Grounding Spatio-Semantic Referring Expressions for Human-Robot Interaction”. In: *RSS Workshop on Spatial-Semantic Representations in Robotics*. Cambridge, USA.

Simon, Herbert A. and Glenn Lea (1974). “Problem Solving and Rule Induction: A Unified View”. In: *Knowledge and Cognition*. Lawrence Eribaum, Oxford, UK, 15–26.

Singh, Vivek Kumar et al. (2020). “Breast tumor segmentation and shape classification in mammograms using generative adversarial and convolutional neural network”. In: *Expert Systems with Applications* 139, p. 112855. URL: `https://www.sciencedirect.com/science/article/pii/S0957417419305573`.

Smith, Edward and Stephen Kosslyn (2006). *Cognitive psychology: Mind and brain*. Upper Saddle River, NJ: Prentice Hall Inc.

Someya, Yasumasa (1998). *English lemma list*.

Southwick, Richard W (1991). “Explaining reasoning: an overview of explanation in knowledge-based systems”. In: *The knowledge engineering review* 6.1, pp. 1–19.

Sridharan, Mohan and Ben Meadows (May 2017). “A Combined Architecture for Discovering Affordances, Causal Laws, and Executability Conditions”. In: *International Conference on Advances in Cognitive Systems (ACS)*. Troy, USA.

Stewart, Russell and Stefano Ermon (2017). “Label-free supervision of neural networks with physics and domain knowledge”. In: *Thirty-First AAAI*

Conference on Artificial Intelligence. URL: `http://phys.csail.mit.edu/papers/16.pdf`.

Sukhbaatar, Sainbayar, Jason Weston, Rob Fergus, et al. (2015). "End-to-end memory networks". In: *Advances in neural information processing systems*, pp. 2440–2448.

Sünderhauf, Niko et al. (2018). "The limits and potentials of deep learning for robotics". In: *The International Journal of Robotics Research* 37.4-5, pp. 405–420. URL: `https://journals.sagepub.com/doi/abs/10.1177/0278364918770733`.

Sycara, Katia P et al. (1998). "Calibrating trust to integrate intelligent agents into human teams". In: *Proceedings of the Thirty-First Hawaii International Conference on System Sciences*. Vol. 1. IEEE, pp. 263–268.

Thippur, Akshaya et al. (2015). "A Comparison of Qualitative and Metric Spatial Relation Models for Scene Understanding". In: *Aaai* Section 4, pp. 1632–1640. URL: `https://pdfs.semanticscholar.org/af7b/83dca60af4ab7d96a2fb7bb0d6f757493f5e.pdf`.

Tu, Phan Huy et al. (2011). "Approximation of action theories and its application to conformant planning". In: *Artificial Intelligence* 175.1, pp. 79–119.

Ulinski, Morgan, Bob Coyne, and Julia Hirschberg (2019). "SpatialNet: A Declarative Resource for Spatial Relations". In: *Proceedings of the Combined Workshop on Spatial Language Understanding (SpLU) and Grounded Communication for Robotics (RoboNLP)*, pp. 61–70.

Ullman, Tomer et al. (2014). "Learning physics from dynamical scenes". In: *Proceedings of the 36th Annual Conference of the Cognitive Science society*, pp. 1640–1645.

Valiant, Leslie G (2008). "Knowledge infusion: In pursuit of robustness in artificial intelligence". In: *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

Wang, Jiang et al. (2016). "Cnn-rnn: A unified framework for multi-label image classification". In: *Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on*. IEEE, pp. 2285–2294.

Wong, Lawson L S, Leslie Pack Kaelbling, and Tomas Lozano-Perez (2013). "Manipulation-based active search for occluded objects". In: *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 2814–2819. ISSN: 10504729. DOI: 10.1109/ICRA.2013.6630966.

Wu, Jiajun et al. (2015). "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning". In: *Advances in neural information processing systems*, pp. 127–135.

Yang, Zhun, Adam Ishay, and Joohyung Lee (2020). "NeurASP: Embracing Neural Networks into Answer Set Programming". In: *International Joint Conference on Artificial Intelligence*.

Ye, Jun and Kien A Hua (2013). "Exploiting depth camera for 3D spatial relationship interpretation". In: *Proceedings of the 4th ACM Multimedia Systems Conference on - MMSys '13*, pp. 151–161. DOI: 10.1145/2483977.2483998.

Zampogiannis, Konstantinos et al. (2015). "Learning the Spatial Semantics of Manipulation Actions through Preposition Grounding". In: *IEEE International Conference on Robotics and Automation*, pp. 1389–1396. ISSN: 10504729. DOI: 10.1109/ICRA.2015.7139371.

Zhang, Renqiao et al. (2016). "A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding". In: *arXiv preprint arXiv:1605.01138*.

Zhang, Yu et al. (2017). "Plan explicability and predictability for robot task planning". In: *International Conference on Robotics and Automation*, pp. 1313–1320.

Zhang, Anthony (2017). *Speech Recognition (version 3.8)*. https://github.com/Uberi/speech_recognition.

Ziaeetabar, Fatemeh et al. (2017). "Semantic Analysis of Manipulation Actions Using Spatial Relations". In: *International Conference on Robotics and Automation*, pp. 4612–4619.