Commonsense Reasoning and Knowledge Acquisition to Guide Deep Learning on Robots

Tiago Mota
Electrical and Computer Engineering
University of Auckland
Auckland 1023, NZ
Email: tmot987@aucklanduni.ac.nz

Mohan Sridharan
School of Computer Science
University of Birmingham
Birmingham B15 2TT, UK
Email: m.sridharan@bham.ac.uk

***Abstract*—Algorithms based on deep network models are being used for many pattern recognition and decision-making tasks in robotics and AI. Training these models requires a large labeled dataset and considerable computational resources, which are not readily available in many domains. Also, it is difficult to understand the internal representations and reasoning mechanisms of these models. The architecture described in this paper attempts to address these limitations by drawing inspiration from research in cognitive systems. It uses non-monotonic logical reasoning with incomplete commonsense domain knowledge, and inductive learning of previously unknown constraints on the domain's states, to guide the construction of deep network models based on a small number of relevant training examples. As a motivating example, we consider a robot reasoning about the stability and partial occlusion of configurations of objects in simulated images. Experimental results indicate that in comparison with an architecture based just on deep networks, our architecture improves reliability, and reduces the sample complexity and time complexity of training deep networks.**

I. Introduction

Consider an assistive robot[1] tasked with clearing away a variety of toys that children have arranged in different configurations in different rooms. Such a task poses a challenging scene understanding problem. It is difficult to provide many labeled training examples of different arrangements of objects, and the robot has to reason with different descriptions of incomplete domain knowledge and the associated uncertainty. This may include qualitative descriptions of commonsense knowledge about the domain objects and relations between them, and *default* statements such as "structures with a large object placed on a small object are typically unstable" that hold in all but a few exceptional circumstances. In addition, the robot may use algorithms for sensing and navigation that model knowledge and the associated uncertainty quantitatively. Furthermore, human participants are unlikely to have the time and expertise to interpret sensor data or provide comprehensive feedback, and reasoning with the incomplete knowledge may result in incorrect or sub-optimal outcomes.

Deep network architectures and the associated algorithms represent the state of the art for scene understanding and many other pattern recognition and decision making tasks in robotics and AI. These algorithms require a large set of labeled training samples, are computationally expensive, and provide results that are not easily interpretable. Research in cognitive systems has shown that many of these challenges can be addressed by exploiting domain knowledge and the tight coupling between knowledge representation, reasoning and learning. Drawing inspiration from such cognitive systems, our architecture 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. We consider the scene understanding tasks of estimating the *partial occlusion* of objects and the *stability* of object configurations, based on limited training examples, in the context of the assistive robotics domain described above. To focus on the interplay between representation, reasoning, and learning, we focus on simulated images of scenes in this domain—Figure 1 shows an example—and limit perceptual processing to that of 3D point clouds corresponding to these scenes. We also assume that the robot knows the grounding (i.e., meaning in the physical world) of words such as "above" and "left_of" that describe basic geometric relations between domain objects. We then describe how the architecture:

Fig. 1: A simulated scene with toys. The robot has to reason about occlusion and stability of structures to reduce clutter.

- 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.
- Uses the labeled examples, i.e., images with occlusion labels for objects and stability labels for object struc-

[1]We use the terms "robot", "learner", and "agent" interchangeably.

tures, to train decision trees for incremental learning of previously unknown constraints governing domain states.
- 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.

Experimental results show a marked improvement in accuracy and computational efficiency in comparison with an architecture that only uses deep networks, while also providing insights about the interplay between reasoning and learning. Section II discusses related work and Section III describes our architecture. Experimental results and conclusions are discussed in Section IV and Section V respectively.

II. Related Work

Scene understanding is a key problem in computer vision and robotics. It includes the identification of relations between scene objects and a variety of estimation and prediction problems. Deep networks provide state of the art performance for such problems. For instance, a Convolutional Neural Network (CNN) has been used to predict the stability of a tower of blocks [20, 19], and to predict the movement of an object sliding down an inclined surface and colliding with another object [36]. However, CNNs and other deep networks require a large number of labeled examples and considerable computational resources to learn the mapping from inputs to outputs. Also, it is difficult to interpret the operation of the learned networks, or to transfer knowledge learned in one scenario or task to a related scenario or task [37]. Since labeled training examples are not readily available in many domains, researchers have used physics engines, e.g., in the context of training deep networks to predict the movement of objects in response to external forces [9, 24, 35], or to understand the physics of scenes [3]. Researchers have also used prior (domain) knowledge during training [34]. For instance, a recurrent neural network (RNN) architecture augmented by arithmetic and logical operations has been used to answer questions about scenes [25]. 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 [33]. 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 [28]. This approach, however, only makes limited use of the available knowledge, and does not revise the constraints over time.

Research in AI and robotics has provided algorithms and architectures for addressing the limitations described above. For instance, theories and algorithms have been developed to enable agents to reason with commonsense knowledge [10, 11]. For scene understanding, domain knowledge often includes the grounding of spatial relations such as *in* and *above*. Measures related to the relative position of objects have been used to predict the successful execution of actions in a new scenario [8], and methods have been developed to reason about and learn spatial relations between objects [13, 21]. Deep networks have been used to infer spatial relations between objects using images and natural language expressions, for applications such as manipulation [26], navigation [27] and human-robot interaction [29]. Researchers have also developed algorithms for learning domain knowledge. Examples include the incremental refinement of a first-order logic representation of action operators [12], the use of inductive learning to acquire domain knowledge represented as Answer Set Prolog (ASP) programs [17], and the integration of non-monotonic logical reasoning and relational reinforcement learning to incrementally acquire domain axioms [31]. 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 [5, 16]. It can be viewed as building on early work on search through the space of hypotheses and observations [30], but such methods have rarely been explored for scene understanding. Our architecture exploits the complementary strengths of deep learning, non-monotonic logical reasoning with commonsense knowledge, and incremental learning of constraints that govern the domain states, as described below.

III. Proposed Architecture

Figure 2 is an overview of the proposed architecture, which takes as input RGB-D images of scenes with different object configurations. During training, the inputs include the occlusion labels of objects and the stability labels of object configurations in the images. Our prior work is used to ground the spatial relations between objects [22]. An object is considered to be occluded if the view of any minimal fraction of 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[2] 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 (ROIs), 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 of our architecture using the following illustrative domain.

Example 1: [Robot Assistant (RA)] A simulated robot analyzes images of scenes containing toys in different configu-

[2]We use "ASP" and "CR-Prolog" interchangeably.

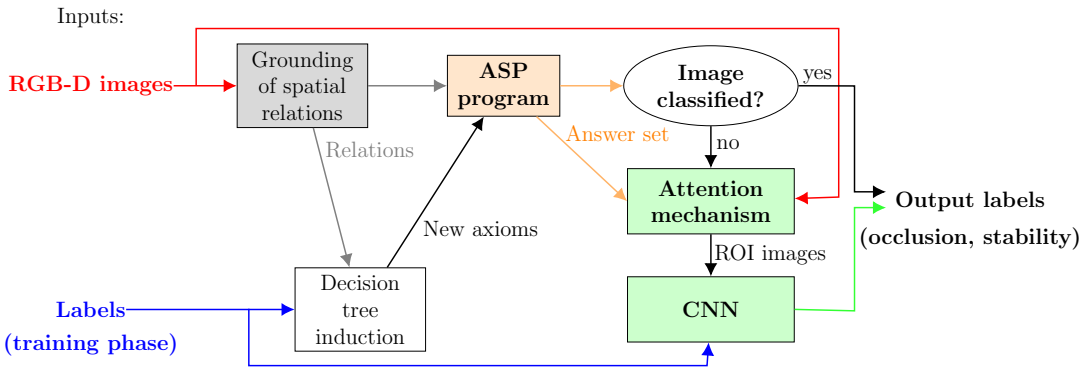


Fig. 2: Architecture combines the complementary strengths of non-monotonic logical reasoning, deep learning, and decision tree induction, to perform the scene understanding tasks reliably and efficiently.

rations. 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.

A. Knowledge Representation with ASP

To represent and reason with incomplete domain knowledge, we use ASP, a declarative language that can represent recursive definitions, defaults, causal relations, special forms of self-reference, and language constructs that occur frequently in non-mathematical domains, and are difficult to express in classical logic formalisms. ASP is based on the stable model semantics [10] and supports concepts such as *default negation* (negation by failure) and *epistemic disjunction*, e.g., unlike "$\neg$a", which implies that "*a is believed to be false*", "`not a`" only implies "*a is not believed to be true*". Each literal can be true, false or unknown and the *robot only believes that which it is forced to believe*. 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 reasoning with incomplete knowledge. ASP and other similar paradigms are often criticized for requiring considerable prior knowledge, and for being unwieldy in large, complex domains. However, modern ASP solvers support efficient reasoning in large knowledge bases with incomplete knowledge, and are used by an international research community in robotics [6] and other applications [7].

A domain's description in ASP comprises a *system description* $\mathcal{D}$ and a *history* $\mathcal{H}$. $\mathcal{D}$ comprises a *sorted signature* Σ and axioms. Σ includes *sorts* arranged hierarchically; *statics*, i.e., domain attributes that do not change over time; and *fluents*, i.e., domain attributes whose values can be changed. In the RA domain, sorts include $object$, $robot$, $size$, $relation$, $surface$, and $step$ for temporal reasoning. Statics include some object attributes such as $obj\_size(object, size)$ and $obj\_surface(obj, surface)$. 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.:

$$in\_hand(robot, object), \quad stable(object) \tag{1}$$

Actions of the RA domain include $pickup(robot, object)$ and $putdown(robot, object)$, and $holds(fluent, step)$ is a predicate implying that a particular fluent holds true at a particular timestep. The RA domain's axioms include:

$$holds(in\_hand(robot, object), I+1) \leftarrow occurs(pickup(robot, object), I) \tag{2a}$$

$$holds(obj\_relation(above, A, B), I) \leftarrow holds(obj\_relation(below, B, A), I) \tag{2b}$$

$$holds(obj\_relation(infront, A, B), I) \leftarrow holds(obj\_relation(behind, B, A), I) \tag{2c}$$

$$\neg occurs(pickup(robot, object), I) \leftarrow holds(in\_hand(robot, object), I) \tag{2d}$$

where Statement 2(a) describes a causal law, 2(b-c) describe constraints, and 2(d) describes an executability condition. 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".

$$\neg holds(stable(A), I) \leftarrow holds(obj\_relation(above, A, B), I), size(A, large), \ size(B, small), \ not\ holds(stable(A), I) \tag{3}$$

Finally $\mathcal{H}$ includes records of observations received and actions executed by the robot.

To reason with the incomplete domain knowledge, we construct the CR-Prolog program $\Pi(\mathcal{D}, \mathcal{H})$—please see our code repository [23]. Planning, diagnostics and inference tasks can then be reduced to computing *answer sets* of Π, which represent beliefs of the robot associated with Π [10]. We use SPARC [2] to compute answer set(s) of ASP programs.

B. Decision Tree Induction

Reasoning with incomplete knowledge can result in incorrect or suboptimal decisions. Our architecture includes a decision tree induction component to incrementally learn relevant knowledge in the form of axioms that represent state constraints. Specifically, separate decision trees are constructed in the RA domain for estimating stability and occlusion, using the spatial relations identified between pairs of objects, and the attributes of objects—the labels assigned to the leaf nodes are stable/unstable or occluded/not occluded. Some example decision trees are shown in Figures 3 and 4.

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 represents a precision higher than 95%, i.e., most examples correspond 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., 50%) are encoded in the ASP program for subsequent reasoning.

Consider the branches highlighted in gray in Figures 3 and 4, which can be translated into the following axioms:

$$stable(A) \leftarrow \ \neg obj\_relation(above, A, B) \tag{4a}$$

$$\neg occluded(A) \leftarrow \ \neg obj\_relation(behind, A, B) \tag{4b}$$

where Statement 4(a) implies that any object that is not above another object is stable, whereas Statement 4(b) says that an object is not occluded if it is not located behind another object. More elaborate axioms are created when other attributes (e.g., size and surface of objects) are considered. For example, the branch highlighted in gray and blue in Figure 3 translates to:

$$\begin{aligned} \neg stable(A) \ \leftarrow \ & obj\_relation(above, A, B), \\ & obj\_surface(B, irregular) \end{aligned} \tag{5}$$

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%). The lower threshold results in the discovery of additional axioms, but it also introduces noise. The underlying non-monotonic logical reasoning capability can be used to identify any inconsistencies caused by the inclusion of incorrect axioms—these errors can be corrected manually or through learning.

C. Attention Mechanism

The attention mechanism module is used only when ASP-based reasoning cannot assign labels to objects in the input image. 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. For instance, the head of Statement 4(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 5 defines some conditions under which an object is considered to be unstable. Both these statements will be considered by the attention mechanism if the robot's task is to estimate the stability of object configurations. In a similar manner, Statement 4(b) will only be explored further when the task is to examine the occlusion of objects.

The selected axioms are used to identify ROIs in the image. More specifically, 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. For instance, to estimate stability in Figure 1, the attention mechanism should consider the stack comprising the red cube, white cylinder and the green ball, since they satisfy the relevant relation $above$—the other two objects (duck and pitcher) can be disregarded. Any image may contain multiple such ROIs, and each ROI may have multiple objects.

D. 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, and 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.

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.

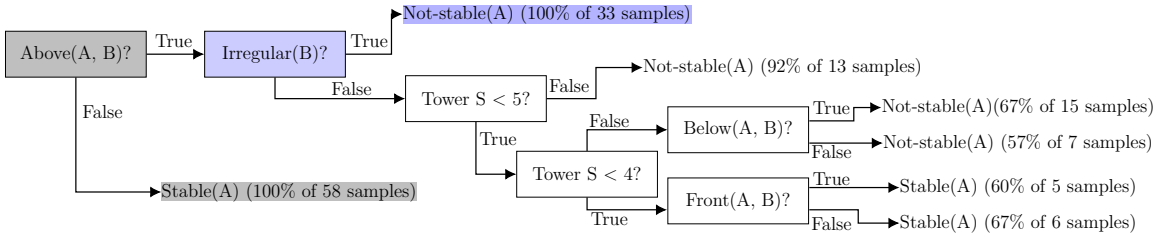


Fig. 3: Example of a decision tree constructed for stability estimation using some labeled examples. Highlighted branches are used to construct previously unknown axioms.

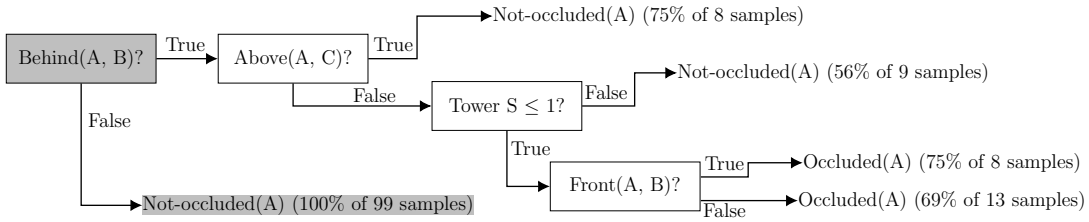


Fig. 4: Example of a decision tree constructed for occlusion estimation using some labeled examples. Highlighted branch is used to construct previously unknown axiom.

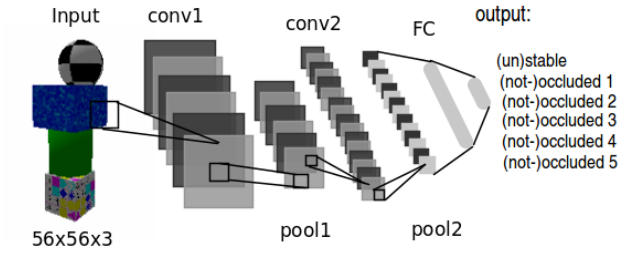


Fig. 5: Lenet architecture.

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 feed-forward 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. 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.

In this paper, we explored two CNN architectures: (i) Lenet [18], initially proposed for recognizing hand-written digits; and (ii) Alexnet [15], which has been widely used since it provided 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 5 is a pictorial representation of this network. As described later in Section IV-A, 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 [14] in TensorFlow [1] 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 [23].

IV. Experimental Setup and Results

In this section, we describe the experimental setup and the results of experimental evaluation of our architecture.

A. Experimental Setup

To simulate experiments in a dynamic domain in 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. 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) [4]. We considered three different arrangements of these objects:

- **Towers**: images containing $2-5$ 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 $2-4$ objects stacked on each other, with the rest $(1-3)$ 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) 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, 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). 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.

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.

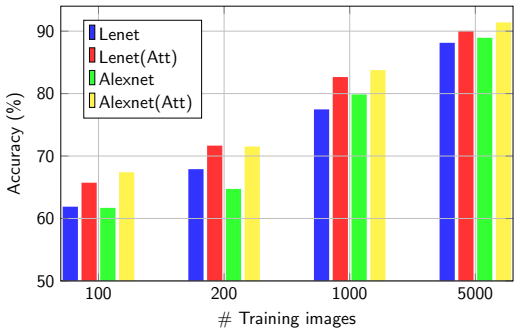


Fig. 6: Accuracy of Lenet and Alexnet with and without commonsense reasoning and the attention mechanism. The number of background images were fixed at 100. Our architecture improves accuracy in comparison with the baselines.

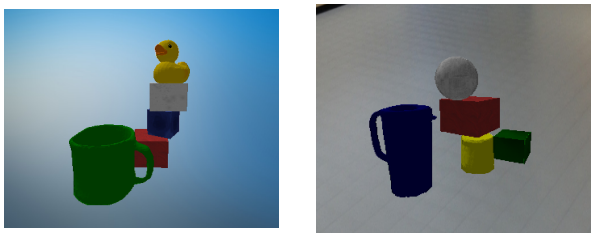
B. Experimental Results

The first set of experiments was designed as follows, with results summarized in Figure 6:

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;
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; and
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 6. The baseline CNNs used the training dataset without attention mechanism or commonsense reasoning, with results plotted as "Lenet" and "Alexnet" in Figure 6.

Figure 6 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 an 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 7 shows two examples of the improvement provided by the attention mechanism. In Figure 7a, 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 7b, both networks 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



(a) (b)

Fig. 7: 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 cylinder.

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 6. 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 Lenet and Lenet(Att) with different number of training examples (100 and 5000) and different number of backgrounds (10, 30, and 100). As shown in Figure 8, with Lenet and 100 training samples, accuracy degrades from $\approx 65\%$ for 10 different background images to $\approx 62\%$ when we have 100 different background images. The degradation is smaller, i.e., $\approx 1\%$, for 5000 training examples with number of backgrounds varying from $10 - 100$. For 1000 different backgrounds (not shown in Figure 8), the drop in accuracy was $\approx 2\%$. These results indicate that a baseline (Lenet) 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 network. With Lenet(Att), on the other hand, Figure 8 shows that there is hardly any change in accuracy regardless of the change in the number of background images—the attention mechanism minimizes the effect of variations in the background by focusing attention on just the relevant ROIs in the image.

The second set of experiments was designed as follows, with results summarized in Figure 9:

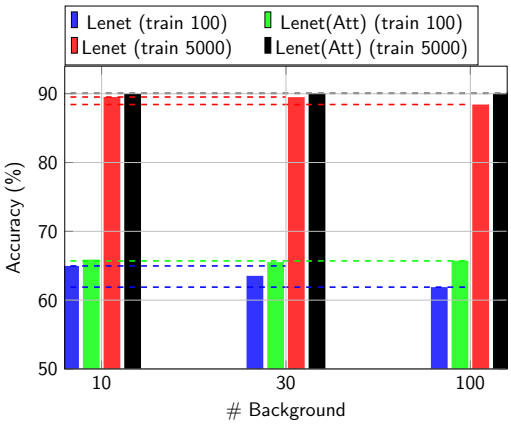


Fig. 8: Effect of changing the number of backgrounds on the accuracy of the Lenet and Lenet(Att) networks for 100 and 5000 training images. Without the attention mechanism and commonsense reasoning, increasing the number of backgrounds reduces the classification accuracy.

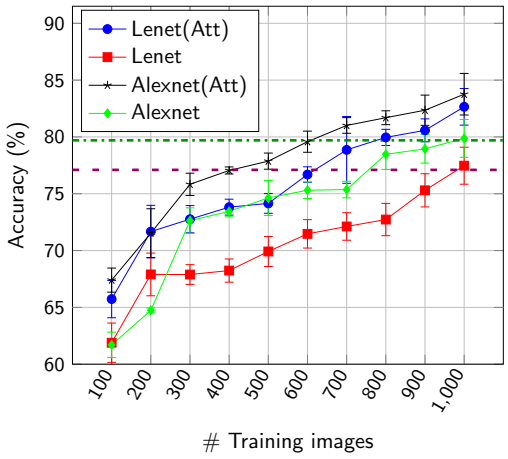


Fig. 9: Accuracy of Lenet and Alexnet with and without the attention mechanism and commonsense reasoning. The number of background images was fixed at 100. Any desired accuracy is achieved with a smaller training set.

1) The Lenet network was trained with training datasets containing between $100 - 1000$ images, in step-sizes of 100. Separate datasets were created for testing;
2) The datasets after applying the attention mechanism were derived from the original datasets from Step-1. 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 a deep network, and the results are plotted as "Lenet(Att)" and "Alexnet(Att)" in Figure 9. The baseline CNN network used the training dataset without the commonsense reasoning or attention mechanism.

Figure 9 shows that using the attention mechanism and

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

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

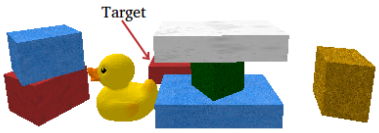


Fig. 10: Illustrative image used for planning experiments with and without the learned axioms.

reasoning with commonsense knowledge helps achieve any desired level of accuracy with much fewer training examples. The purple dashed (horizontal) line in Figure 9 indicates that the baseline Lenet needs $\approx$ 1000 images to reach an accuracy of 77%, whereas Lenet(Att) reduces this number to $\approx$ 600. A similar difference is observed between Alexnet and Alexnet(Att) for $\approx$ 80% accuracy—see the dark green dash-dotted (horizontal) line in Figure 9. In other words, the use of commonsense knowledge helps train deep networks with fewer examples, reducing both the computational requirements and storage requirements. These results support hypothesis **H2**.

Finally, the third set of experiments was designed as follows, with results summarized in Table I:

1) Ten sets of 50 labeled images were created, as described in Section IV-A;
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—see Section III-B;
3) The precision and recall for the unknown axioms, e.g., Statements 4(a), 4(b), and 5, but excluding defaults (i.e., with threshold of 0.95), are summarized as "unknown (normal)" in Table I;
4) The precision and recall for the unknown default statements, e.g., Statement 3, (i.e., with threshold of 70%) are summarized as "unknown (default)" in Table I;

Table I 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 is understandable because it is challenging to distinguish between defaults and their exceptions. Although we do not describe it here, reasoning with commonsense knowledge and decision trees also provides (at least partial) explanations for the decisions made by the architecture.

Finally, we ran experiments in which the robot computed minimal plans to pickup 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 the transition diagram. For instance, the goal in one set of experiments was to clear the large red box partially hidden behind the blue, green and white boxes, and the duck in Figure 10. With all the axioms the robot found three plans, all of which were correct. However, with some axioms missing, the robot found as many as 64 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**.

V. Discussion and Conclusions

Deep network architectures and algorithms represent the state of the art for many tasks in robotics and AI. However, they require large training datasets and considerable computational resources, and make it difficult to understand their operation. The architecture described in this paper 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.

The work described here opens up multiple directions for future research. First, we will explore the interplay between reasoning and learning to better understand the operation of deep network models. We have already shown in other work that the use of relational logical structures makes it easier to explain the decisions, the underlying knowledge and beliefs, and the experiences that informed these beliefs [32]. Second, we will expand the learning capabilities of our architecture. Although we have only discussed the learning of state constraints in this paper, we have (in other work) explored the use of relational reinforcement learning and limited human feedback for learning different types of domain knowledge [31]. Furthermore, we will explore the use of this architecture on robots assisting humans in more complex domains.

Acknowledgements

This work was supported in part by the Asian Office of Aerospace Research and Development award FA2386-16-1-4071. All opinions and conclusions described in this paper are those of the authors.

References

[1] M Abadi, A Agarwal, P Barham, E Brevdo, Z Chen, C Citro, GS Corrado, A Davis, J Dean, M Devin, et al. TensorFlow: large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016. URL https://arxiv.org/abs/1603.04467.

[2] Evgenii Balai, Michael Gelfond, and Yuanlin Zhang. Towards Answer Set Programming with Sorts. In *International Conference on Logic Programming and Nonmonotonic Reasoning*, Corunna, Spain, September 15-19, 2013. URL https://link.springer.com/chapter/10.1007/978-3-642-40564-8_14.

[3] Peter W Battaglia, Jessica B Hamrick, and Joshua B Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences*, page 201306572, 2013. URL https://www.pnas.org/content/110/45/18327.full.

[4] Berk Calli, Aaron Wallsman, Arjun Singfh, and Siddhartha S. Srinivasa. Benchmarking in Manipulation Research. *IEEE Robotics and Automation Magazine*, (September):36–52, 2015. URL https://ieeexplore.ieee.org/document/7254318.

[5] J. Y. Chai, Q. Gao, L. She, S. Yang, S. Saba-Sadiya, and G. Xu. Language to Action: Towards Interactive Task Learning with Physical Agents. In *International Joint Conference on Artificial Intelligence (IJCAI)*, Stockholm, Sweden, July 13-19, 2018. URL https://www.ijcai.org/proceedings/2018/0001.pdf.

[6] Esra Erdem and Volkan Patoglu. Applications of Action Languages to Cognitive Robotics. In *Correct Reasoning*. Springer-Verlag, Berlin, 2012. URL https://link.springer.com/chapter/10.1007/978-3-642-30743-0_16.

[7] Esra Erdem, Michael Gelfond, and Nicola Leone. Applications of Answer Set Programming. *AI Magazine*, 37(3): 53–68, 2016. URL https://www.aaai.org/ojs/index.php/aimagazine/article/view/2678.

[8] Severin Fichtl, Dirk Kraft, Norbert Krüger, and Frank Guerin. 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), Hamburg*. Citeseer, 2015.

[9] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015. URL https://arxiv.org/abs/1511.07404.

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

[11] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. Morgan Kaufmann, San Francisco, USA, 2004.

[12] Yolanda Gil. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. In *International Conference on Machine Learning*, pages 87–95, New Brunswick, USA, July 10-13, 1994. URL https://www.sciencedirect.com/science/article/pii/B9781558603356500192.

[13] Philipp Jund, Andreas Eitel, Nichola Abdo, and Wolfram Burgard. Optimization Beyond the Convolution: Generalizing Spatial Relations with End-to-End Metric Learning. In *ICRA*, 2018. URL https://ieeexplore.ieee.org/abstract/document/8460220.

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. URL https://arxiv.org/abs/1412.6980.

[15] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. URL https://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[16] John E. Laird, Kevin Gluck, John Anderson, Kenneth D. Forbus, Odest Chadwicke Jenkins, Christian Lebiere, Dario Salvucci, Matthias Scheutz, Andrea Thomaz, Greg Trafton, Robert E. Wray, Shiwali Mohan, and James R. Kirk. Interactive Task Learning. *IEEE Intelligent Systems*, 32(4):6–21, 2017. URL https://ieeexplore.ieee.org/abstract/document/8012335.

[17] Mark Law, Alessandra Russo, and Kysia Broda. The Complexity and Generality of Learning Answer Set Programs. *Artificial Intelligence*, 259:110–146, 2018. URL https://www.sciencedirect.com/science/article/pii/S000437021830105X.

[18] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. URL https://ieeexplore.ieee.org/document/726791.

[19] Adam Lerer, Sam Gross, and Rob Fergus. Learning physical intuition of block towers by example. *arXiv preprint arXiv:1603.01312*, 2016. URL https://arxiv.org/abs/1603.01312.

[20] Wenbin Li, Aleš Leonardis, and Mario Fritz. Visual stability prediction and its application to manipulation. *arXiv preprint arXiv:1609.04861*, 2016. URL https://arxiv.org/abs/1609.04861.

[21] Oier Mees, Nichola Abdo, Mladen Mazuran, and Wolfram Burgard. Metric learning for generalizing spatial relations to new objects. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3175–3182, 2017. URL https://ieeexplore.ieee.org/abstract/document/8206149.

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

[23] Tiago Mota and Mohan Sridharan. Software related to the paper, 2019. https://github.com/tmot987/RSS2019.

[24] Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. "What happens if.." Learning to Predict the Effect of Forces in Images. In *European Conference on Computer Vision*, pages 269–285. Springer, 2016. URL https://link.springer.com/chapter/10.1007/978-3-319-46493-0_17.

[25] Arvind Neelakantan, Quoc V Le, and Ilya Sutskever. Neural programmer: Inducing latent programs with gradient descent. *arXiv preprint arXiv:1511.04834*, 2015. URL https://arxiv.org/pdf/1511.04834.pdf.

[26] Rohan Paul, Jacob Arkin, Nicholas Roy, and Thomas Howard. Efficient Grounding of Abstract Spatial Concepts for Natural Language Interaction with Robot Manipulators. In *Robotics: Science and Systems*, Ann Arbor, USA, June 18-22, 2016. URL http://www.roboticsproceedings.org/rss12/p37.pdf.

[27] Andrzej Pronobis and Rajesh Rao. Learning Deep Generative Spatial Models for Mobile Robots. In *RSS Workshop on Spatial-Semantic Representations in Robotics*, Cambridge, USA, July 16, 2017.

[28] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Tim Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017. URL http://papers.nips.cc/paper/7082-a-simple-neural-network-module-for-relational-reasoning.

[29] Mohit Shridhar and David Hsu. Grounding Spatio-Semantic Referring Expressions for Human-Robot Interaction. In *RSS Workshop on Spatial-Semantic Representations in Robotics*, July 2017.

[30] Herbert A. Simon and Glenn Lea. Problem Solving and Rule Induction: A Unified View. In *Knowledge and Cognition*, pages 15–26. Lawrence Eribaum, Oxford, UK, 1974.

[31] Mohan Sridharan and Ben Meadows. Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration. *Advances in Cognitive Systems*, 7:77–96, December 2018. URL http://www.cogsys.org/papers/ACSvol6/article13.pdf.

[32] Mohan Sridharan and Ben Meadows. Theory of explanations for human-robot collaboration. In *AAAI Spring Symposium on Story-Enabled Intelligence*, Stanford, USA, March 25-27, 2019.

[33] Russell Stewart and Stefano Ermon. Label-free supervision of neural networks with physics and domain knowledge. In *AAAI*, volume 1, pages 1–7, 2017. URL http://phys.csail.mit.edu/papers/16.pdf.

[34] Niko Sünderhauf, Oliver Brock, Walter Scheirer, Raia Hadsell, Dieter Fox, Jürgen Leitner, Ben Upcroft, Pieter Abbeel, Wolfram Burgard, Michael Milford, et al. The limits and potentials of deep learning for robotics. *The International Journal of Robotics Research*, 37(4-5):405–420, 2018. URL https://journals.sagepub.com/doi/abs/10.1177/0278364918770733.

[35] Misha Wagner, Hector Basevi, Rakshith Shetty, Wenbin Li, Mateusz Malinowski, Mario Fritz, and Ales Leonardis. Answering Visual *What-If* Questions: From Actions to Predicted Scene Descriptions. In *Visual Learning and Embodied Agents in Simulation Environments (VLEASE) Workshop at ECCV*, Munich, Germany, September 9, 2018. https://arxiv.org/abs/1809.03707.

[36] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Advances in neural information processing systems*, pages 127–135, 2015. URL https://papers.nips.cc/paper/5780-galileo-perceiving-physical-object-properties-by-integrating-a-physics-engine-with-deep-learning.

[37] Renqiao Zhang, Jiajun Wu, Chengkai Zhang, William T Freeman, and Joshua B Tenenbaum. A comparative evaluation of approximate probabilistic simulation and deep neural networks as accounts of human physical scene understanding. *arXiv preprint arXiv:1605.01138*, 2016. URL https://arxiv.org/abs/1605.01138.