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 increasingly 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, which limits their use in some applications. As a step towards addressing these limitations, the architecture described in this paper draws inspiration from research in cognitive systems. It embeds non-monotonic logical reasoning with incomplete commonsense domain knowledge, and inductive learning of previously unknown constraints on the states of the domain, to automatically focus and guide the construction of deep networks based on a small number of relevant training examples. We consider a robot reasoning about the stability and partial occlusion of configurations of objects in simulated images of an indoor domain as a motivating example of a dynamic domain with limited training examples. Experimental results indicate that in comparison with an architecture based just on deep networks, the proposed architecture improves reliability, and reduces the sample complexity and time complexity of training the deep networks.

I. INTRODUCTION

Consider an assistive robot tasked with clearing away a variety of toys that children have arranged in different configurations in different rooms. In such a dynamic domain, the robot’s 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 larger object placed on a smaller 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.

State of the art methods for scene understanding are based on deep network architectures. Although these methods have been reported to provide high accuracy for different pattern recognition and decision making tasks in robotics and AI, they 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. The architecture described in this paper draws inspiration from this research—it 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 illustrative example—and limit perceptual processing to that of 3D point clouds extracted from the scene. 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:

- Attempts to perform the estimation tasks by using a non-monotonic logical reasoning paradigm to reason 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 structures, 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 in our illustrative domain show a marked improvement in the accuracy and computational efficiency, while also providing insights about the interplay between reasoning and learning algorithms. Section II discusses related work, followed by the description of the proposed architecture in Section III. Experimental results are discussed in Section IV and the conclusions are in Section V.

II. RELATED WORK

Scene understanding is a key problem in computer vision and robotics, which includes the identification of relations between scene objects and a variety of estimation and prediction problems. Deep networks increasingly provide state of the art performance for many computer vision and control problems, e.g., object recognition [13] and for scene understanding problems. For instance, a Convolutional Neural Network (CNN) has been used to predict the stability of a tower of blocks [21, 20], and to predict the movement of an object sliding down an inclined surface and colliding with another object [35]. However, CNNs and other deep networks require a large number of labeled examples to learn the mapping from inputs to outputs. Also, they are computationally expensive, the operation of the learned networks are not easily interpretable, and it is difficult to transfer knowledge learned in one scenario or task to a related scenario or task [50]. In dynamic domains that make it difficult to obtain many labeled training examples, one popular approach is to use physics engines, e.g., in the context of using deep networks to predict the movement of objects in response to the application of external forces [10, 24, 34]. Physics engines have also been used to understand the physics of scenes [4].

One option to reduce the computational effort and the need for large, labeled datasets during training is to use prior domain knowledge [33]. For instance, an RNN architecture augmented by arithmetic and logical operations has been used to answer questions about the scene [25]. However, 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, reducing the effort in labeling training images [32]. However, this approach requires the constraints to be encoded manually as loss functions for each task. The structure of deep networks has also been used to constrain and simplify 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 knowledge/constraints over time.

Research in AI and robotics has provided many algorithms and architectures for addressing the limitations of existing approaches for scene understanding. For instance, theories of action and algorithms for reasoning with commonsense knowledge have been used with agents and robots [11]. For scene understanding, domain knowledge often includes the grounding (i.e., interpretation in the physical world) of spatial relations such as in, behind, and above. Measures related to the relative position of objects have been used to predict the successful application of actions in a new scenario [9], and researchers have explored reasoning about and learning spatial relations between objects [14, 22]. Deep networks have been used to infer spatial relations between scene objects from images and natural language expressions for applications such as manipulation [26], navigation [27] and HRI [29]. Researchers have also developed algorithms for learning domain knowledge. Early work used a first-order logic representation and incrementally refined the action operators [12]. More recent work combined Answer Set Prolog (ASP) with inductive learning to acquire domain knowledge represented as ASP programs [18], and integrated principles or non-monotonic logical reasoning and relational reinforcement learning to incrementally learn domain axioms [31]. As such, 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 [6, 17]. It can be viewed as building on early work on joint search through the space of hypotheses and observations [30], but such methods have not been fully explored for scene understanding.

In this paper, we assume that grounding of spatial relations is computed using an existing approach [23], and explore the complementary strengths of deep learning, non-monotonic logical reasoning with commonsense knowledge, and incremental learning of state constraints that govern the domain.

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. Training input includes the occlusion labels of objects and the stability labels of object configurations. An existing method is used to ground the spatial relations between objects [23]. 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 component object is unstable. A decision tree induction algorithm maps object attributes and spatial relations to the target classes, and branches with sufficient support among the training examples are used to construct axioms denoting the state constraints.
The learned constraints, commonsense domain knowledge, and the computed spatial relations are encoded in an ASP\[^2\] program. 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 ROIs to labels. During testing, the input image is either processed by ASP-reasoning or the learned CNN (i.e., decision trees are not used). The individual modules are described using the following illustrative domain.

**Example 1:** [Robot Assistant (RA)] A simulated robot analyzes images of scenes with toys in different configurations to estimate occlusion of objects and stability of object structures, and to rearrange object structures so as to minimize clutter. Each object’s attributes include size (small, medium, large), surface (flat, irregular) and shape (cube, cylinder, duck), and the relation between objects can be (above, below, front, behind, right, left, close). The robot can move objects to achieve assigned goals. Domain knowledge includes axioms governing 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 Answer Set Prolog (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 [11] and supports concepts such as default negation (negation by failure) and epistemic disjunction, e.g., unlike “¬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 things that 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 the 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 [7] and other applications [8].

A domain’s description in ASP comprises a system description \(D\) and a history \(H\). \(D\) comprises a sorted signature \(\Sigma\) and axioms. \(\Sigma\) 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 \(\text{obj}_\text{size}(\text{object}, \text{size})\) and \(\text{obj}_\text{surface}(\text{obj}, \text{surface})\). The spatial relations \(\text{obj}_\text{relation}(\text{relation}, \text{object}, \text{object})\) between objects are fluents described in terms of their arguments' sorts, e.g., \(\text{obj}_\text{relation}(\text{above}, A, B)\) implies object \(A\) is above object \(B\). Note that the last argument in these relations is the reference object. Besides spatial relations, fluents describe other aspects of the domain, e.g.:

\[
\text{in\_hand}(\text{robot}, \text{object}), \quad \text{stable}(\text{object})
\]

while \(\text{pickup}(\text{robot}, \text{object})\) and \(\text{putdown}(\text{robot}, \text{object})\) are actions. Also, predicate \(\text{holds}((\text{fluent}, \text{step})\) implies that a particular fluent holds true at a particular timestep.

The axioms of \(D\) govern domain dynamics and include:

\[
\begin{align}
\text{holds}(\text{in\_hand}(\text{robot}, \text{object}), I + 1) & \leftarrow (2a) \\
\text{occurs}(&\text{pickup}(\text{robot}, \text{object}), I) \\
\text{holds}(\text{obj}_\text{relation}(\text{above}, A, B), I) & \leftarrow (2b) \\
&\text{holds}(\text{obj}_\text{relation}(\text{below}, B, A), I) \\
\text{holds}(\text{obj}_\text{relation}(\text{in\_front}, A, B), I) & \leftarrow (2c) \\
&\text{holds}(\text{obj}_\text{relation}(\text{behind}, B, A), I) \\
\text{¬occurs}(&\text{pickup}(\text{robot}, \text{object}), I) \\
&\text{holds}(\text{in\_hand}(\text{robot}, \text{object}), I)
\end{align}
\]
where Statement \(2a\) describes a causal law, (b-c) describe constraints, and (d) describes an executability condition. The spatial relations extracted from RGB-D images are converted to the facts used 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”.

\[
\neg \text{holds}(\text{stable}(A), I) \leftarrow \text{holds}(\text{obj\_relation}(\text{above}, A, B), I), \\
\text{size}(A, \text{large}), \text{size}(B, \text{small}), \quad (3) \\
\neg \text{not\_holds}(\text{stable}(A), I)
\]

Finally \(H\) includes records of observations received and actions executed by the robot.

To reason with the incomplete domain knowledge, we construct the CR-Prolog program \(II(D, H)\)—please see our code repository [2]. Planning, diagnostics and inference tasks can then be reduced to computing answer sets of II, which represent beliefs of the robot associated with II [11]. We use SPARC [5] to compute answer set(s) of ASP programs.

B. Decision Tree Induction

The spatial relations identified between pairs of objects and the attributes of objects are used to build decision trees for classification. In the RA domain, separate decision trees are built for estimating stability and occlusion, with the labels assigned to the leaf nodes being stable/unstable or occluded/not occluded respectively. One half of the available examples are used for training, while the other half is used for validating the axioms extracted from learned trees. We used an existing algorithm that constructs decision trees by computing the entropy (and thus information gain) of a split in the tree based on each attribute. Illustrative examples of decision trees are shown in Figures 3 and 4.

We use ensemble learning to construct decision trees and extract state constraints. Any branch of a decision tree in which the leaf represents a precision higher than 95%, i.e., most (if not all) examples correspond to a particular class, is used to construct axioms that are validated using the other half of the labeled examples. The validation process: (i) removes axioms without a minimum support; and (ii) compares the discovered axioms to reduce them to their most general version. Since the number of labeled examples is small, we reduce the effect of noise by repeating the learning and validation process a number of times (e.g., 100) and the axioms voted more than a minimum number of times (e.g., 50%) are encoded in the ASP program and used for subsequent reasoning.

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

\[
\text{stable}(A) \leftarrow \neg \text{obj\_relation}(\text{above}, A, B) \quad (4a) \\
\neg \text{occluded}(A) \leftarrow \neg \text{obj\_relation}(\text{behind}, A, B) \quad (4b)
\]

where Statement \(4a\) implies that any object that is not above another object is stable, whereas Statement \(4b\) 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:

\[
\neg \text{stable}(A) \leftarrow \text{obj\_relation}(\text{above}, A, B), \\
\text{obj\_surface}(B, \text{irregular})
\]

which states that an object is unstable if it is located above another object with an irregular surface. Similar extensions to axioms are found when we also consider the shape of objects.

The architecture is also able to discover default knowledge that holds in all but a few exceptional circumstances. To find these axioms by allowing for some exceptions, we reduced the threshold for selecting a branch of a tree from 95% to a smaller value (e.g., 70%), which results in the discovery of additional axioms. As we will see later, this lower threshold results in noisy estimates of axioms.

C. Attention Mechanism

The attention mechanism module of the architecture is used only when ASP-based reasoning cannot assign labels to objects in the input image. It first identifies each axiom in the ASP program whose head corresponds to a relation/fluent of interest. For instance, Statement \(3a\) represents a state constraint in which the head, which implies that an object is stable, holds true in any state in which all the relations in the body of the axiom are satisfied. Statement \(5\) defines 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 \(3b\) establishes a connection between occlusion and the spatial relation behind. This axiom and relation will only be explored further when the task is to examine the occlusion of objects.

Next, the selected axioms are used to identify regions of interest (ROIs) in the image. More specifically, for each axiom considered to be of interest, the body of the axiom provides the relations to be used to constrain the input image and identify regions of interest (ROIs) that should be considered for further processing; the remaining image regions are unlikely to provide any useful information and thus need not be analyzed further. For instance, in Figure 1, the attention mechanism should consider the stack comprising the red cube, white cylinder and the green ball, since they satisfy the relation above that is relevant to stability estimation—the other two objects (duck and pitcher) can be disregarded. Any image may contain multiple such ROIs, and each ROI may have multiple objects. In summary, the attention mechanism:

- Directs attention to images that the agent cannot process by reasoning with domain knowledge; and
- Identifies ROIs in these images for further analysis by removing regions that do not contain any information relevant to the task at hand.

D. Convolutional Neural Networks

The ROIs identified by the attention mechanism serve as input to a deep network—we use two variants of CNN. Recall that pixels of any such ROI contains information directly
In the context of images, convolutional layers extract useful attributes to model the mapping from inputs to outputs. For instance, the initial layers may extract lines and arches, whereas the subsequent layers may compose complex shapes such as squares and circles. In the context of 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. The pooling layers limit the number of parameters, control overfitting, and reduce the image resolution such that each pixel corresponds to the mean or maximum of neighboring pixels.

In this paper, we considered two CNN architectures: the simple Lenet [19], initially proposed for recognizing handwritten digits; and the widely used Alexnet [16], which provided best results on the Imagenet (benchmark) challenge in 2012. 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 $28 \times 28$ gray-scale input images and the ten-class softmax output layer used in the original implementation for classifying digits, we consider $56 \times 56$ 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, we consider ROIs with up to five objects in the experimental studies. 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, $227 \times 227$ RGB images were used as input and the output classes determined the target variables estimating occlusion.

**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.
and stability—once again, we have five outputs estimating occlusion to consider ROIs with up to five objects, and one output for stability of the scene. Due to the multi-class labeling problem, the sigmoid activation function was used in both networks. We used the Adam optimizer \cite{Kingma2015} in TensorFlow \cite{tensorflow2015-whitepaper} 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 \cite{our-repo}.

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) \cite{Balakrishnan2012}. These objects are arranged in three different scenarios:

- **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/simple or no occlusion. Lighting, orientation, camera distance and 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 suitable ROIs from images in the original dataset (not processed by ASP-based reasoning), by identifying relevant axioms and relations in the ASP program. Pixels in these images that are outside the ROI are cropped off. CNNs trained using these two datasets were compared for different amounts of training data and for networks varying in complexity. As stated earlier, occlusion is estimated for each object (five outputs) and stability is estimated for the structure (1 output). The experiments were designed to test the following hypotheses:

- **H1** Reasoning with commonsense domain knowledge and the associated attention mechanism improves the accuracy of deep networks;
- **H2** Reasoning with commonsense domain knowledge and the associated 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 performance measure was primarily 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.

B. Experimental Results

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

1) Labeled images containing objects were generated, as described in the Section IV-A.
2) 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;
3) The dataset after the application of the attention mechanism was derived from the original dataset in step-2. The selection of images as well as the pixels from each image was based on the target task and relations of interest; and
4) The dataset created in step-3 was 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 CNN networks used the training dataset without ROI extraction or commonsense reasoning, with results plotted as “Lenet” and “Alexnet” in Figure 6.

Figure 3 indicates that our architecture, which integrates commonsense reasoning with deep learning, improves the accuracy of the Lenet and Alexnet networks for the joint estimation of stability and occlusion in scenes. We notice that training and testing the deep networks with only those images that cannot be processed by commonsense reasoning enables these networks to focus their attention on the relevant parts of the images, resulting in better performance, i.e., higher accuracy. We notice that the benefits are more pronounced when the training dataset is smaller, but there is significant improvement in performance at all training dataset sizes considered in our experiments. These results thus supports hypothesis H1.

Figure 7 shows two examples illustrating the improvement provided by the use of the attention mechanism. In Figure 7A,
Fig. 6: Accuracy of Lenet and Alexnet CNNs with and without the commonsense reasoning and the attention mechanism. The number of background images were fixed at 100 in these trials. Our architecture improves accuracy.

Fig. 7: Examples of test images for the Lenet CNN: (a) Both Lenet and Lenet(Att) detected the occlusion of the red cube by the green mug, but only the latter correctly predicted the tower’s instability; and (b) Both predicted the instability of the tower, but only the Lenet(Att) detected the obstruction caused by the yellow cylinder on the green cube.

both the Lenet and the Lenet(Att) networks were able to recognize the occlusion of the red cube caused by the green mug, but only the latter, which uses the attention mechanism in conjunction with the 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 the Lenet(Att) network was able to identify the occlusion/obstruction of the green cube caused by the yellow can. The classification errors are most probably because a similar example had not been observed during the training phase—this is a common limitation of deep architectures. The attention mechanism eliminates the analysis of unnecessary parts of images and focuses attention on the relevant parts, resulting in a more targeted network that provides better classification accuracy. For this example, the CNNs were trained with 1000 images, and the two test scenes were not seen during training.

Note that the number of different backgrounds (selected randomly) was fixed at 100 for the experimental results summarized in Figure 6. For different number of training examples, this varied the relation between the number of training examples and the backgrounds. 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 due to a number of factors such as lighting, viewpoint, orientation etc. To analyze the effect of different backgrounds, we explored the use of the the Lenet architecture with different number of training examples (100 and 5000) and different number of backgrounds (30, 50, and 100). As shown in Figure 8, varying the background does have an impact on the accuracy, which degraded from \( \approx 65\% \) for one background per 10 images to \( \approx 62\% \) when this ratio increases to 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 almost \( \approx 2\% \). 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 reduces performance for the regular Lenet architecture. On the other hand, the proposed attention mechanism minimizes the effect of background on classification performance, providing almost the same accuracy regardless of the variations in the background.

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

1) Labeled images containing objects were created, as described in Section IV-A
2) The Lenet network was trained with training datasets containing between 100 – 1000 images, in step-sizes of 100. Separate set of scenes was created for testing;
3) The dataset after applying the attention mechanism
was derived from the original dataset from step-2. 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

4) The dataset created in step-3 was used to train and test a deep network, and the results are plotted as “Lenet(Att)” in Figure 9. The baseline CNN network used the training dataset without the commonsense reasoning or ROI extraction.

In these experiments, we used Lenet and not Alexnet because the former was observed to provide performance comparable to the latter but with much less computational effort.

Figure 9 shows that the proposed attention mechanism supported by reasoning with commonsense knowledge achieves a desired level of accuracy with much fewer training examples. For instance, the orange dashed line in Figure 9 indicates that the baseline Lenet needs \( \approx 1000 \) images to reach an accuracy of 77\%, whereas our architecture reduces this number to \( \approx 600 \). In other words, the deep networks can be trained with fewer examples because the commonsense knowledge is exploited, reducing both the computation and storage requirements. These results support hypothesis \( \text{H2} \).

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 (excluding defaults and with threshold of 95\%), e.g., Statements 3(a), 3(b), and 3(c) are summarized as “unknown (normal)” in Table I.
4) The precision and recall for the unknown default statements (with threshold of 70\%), e.g., Statement 3 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 architecture’s performance.

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 white box and the duck in Figure 10. 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. All these results support hypothesis \( \text{H3} \).

V. CONCLUSION

Deep network architectures and algorithms are providing state of the art performance for many pattern recognition 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 combines the principles of reasoning with incomplete commonsense domain knowledge, and decision tree induction, with deep learning. In the context of estimating occlusion of objects and the stability of object configurations in simulated images, we observe that the proposed architecture improves the accuracy and
computational efficiency of the deep network architectures. Future work will further examine the performance of this architecture in other (more complex) domains, and examine the explainability of the observed performance.

REFERENCES


