Non-monotonic Logical Reasoning to Guide Deep Learning for Explainable Visual Question Answering

Heather Riley
The University of Auckland, NZ
hril230@aucklanduni.ac.nz

Mohan Sridharan
University of Birmingham, UK
m.sridharan@bham.ac.uk

Abstract

Deep learning algorithms represent the state of the art for many problems in robotics and AI. However, they require a large labeled dataset, are computationally expensive, and the learned models are difficult to understand. Our architecture draws inspiration from research in cognitive systems to address these limitations. In the context of answering explanatory questions about scenes and an underlying classification task, our architecture uses non-monotonic logical reasoning with incomplete commonsense domain knowledge, and the features extracted from input images, to answer the input queries. Features from images not processed by such reasoning are mapped to the desired answers using a learned deep network model. In addition, previously unknown state constraints of the domain are learned incrementally and used for subsequent reasoning. Experimental results show that in comparison with an "end to end" deep architecture, our architecture significantly improves accuracy and efficiency of decision making.

1 Introduction

Deep networks represent the state of the art for many problems in robotics and AI. However, training these data-driven models requires many labeled training examples and considerable computational resources, which are not available in many domains. Also, it is difficult to interpret the behavior of the learned models, whereas humans may want to understand the decisions made by an automated reasoning or learning system. This "explainability" also helps designers improve the underlying algorithms.

In this paper, we consider Visual Question Answering (VQA) as a motivating example of a complex task requiring explainable reasoning and learning. Given an image of a scene, the objective is to answer explanatory questions, e.g., about objects and their relationships, or the outcomes of executing actions. Deep networks represent the state of the art for VQA, but exhibit the limitations described above. To address these limitations, we draw inspiration from research in cognitive systems, which indicates that explainable reasoning and learning can be achieved by jointly reasoning with incomplete domain knowledge and learning from experience. For VQA, our architecture uses Convolutional Neural Networks (CNNs) to extract concise visual features from image(s) of any given scene. It first attempts to answer the questions about the scene and an underlying classification problem using non-monotonic logical reasoning with the extracted features and incomplete commonsense domain knowledge. Feature vectors not classified by such reasoning train a decision tree classifier that is then used to answer questions about the classification. The decision tree's output and the feature vectors then train a Recurrent Neural Network (RNN) to answer the questions. Furthermore, feature vectors that are misclassified (or not classified) are used to learn constraints for subsequent reasoning.

For evaluation, we consider VQA while: (i) estimating the stability of configurations of simulated blocks; and (ii) recognizing traffic signs in a benchmark image dataset. We also consider a simulated robot computing and executing plans. We do not consider benchmark datasets and algorithms for VQA that focus on generalizing across domains, and do not support our architecture's capabilities. Our focus is very different; we want to explore the interplay between commonsense reasoning and learning for *explainable, reliable, and efficient scene understanding in any given domain*, especially when a large labeled dataset is not available. Experimental results show a significant improvement in accuracy, efficiency, and the ability to compute correct plans, in comparison with an architecture based only on deep networks.

2 Related work

Although deep networks represent state of the art for VQA (Malinowski et al., 2017) and other pattern recognition tasks, they are computationally expensive, require

Preliminary work. Under review by RAI-NR 2019. Do not distribute.

large, labeled datasets, and make it difficult to understand the internal representations, transfer knowledge, or identify bias. Methods have been developed to understand the operation of deep networks, e.g., by computing the contribution of each neuron in a CNN to the decision (Selvaraju et al., 2017), or using captions to explain answers to questions (Li et al., 2018). Methods have also been developed to understand the predictions of learning algorithms, e.g., by tracing predictions back to data (Koh & Liang, 2017).

The training data requirements (or data bias) of a deep network can be reduced by focusing on data relevant to the task(s) at hand. Examples for VQA include a stacked attention network that prioritizes relevant features (Yang et al., 2016), or a method that reduces data bias by associating questions with images that require different answers (Goyal et al., 2017). Learning for VQA has also been made more efficient by answering common questions using domain knowledge (Wang et al., 2017), and using physics engines that simulate domain knowledge (Wagner et al., 2018).

Cognitive systems research indicates that reliable, efficient, and explainable reasoning and learning can be achieved by jointly reasoning with domain knowledge and learning from experience. Methods that refine first-order logic representations of action operators do not support commonsense reasoning or merging of new, unreliable information Gil (1994). Non-monotonic logics such as Answer Set Prolog (ASP) address these limitations in different applications Erdem et al. (2016). ASP has been combined with inductive learning to acquire domain knowledge Law et al. (2018), and combined with probabilistic representations for reasoning Baral et al. (2009). Approaches based on classical first-order logic are not expressive enough, e.g., modeling uncertainty by attaching probabilities to logic statements is not always meaningful. Logic programming methods, by themselves, do not support all desired capabilities such as efficient incremental learning of knowledge and real-time reasoning with large probabilistic components. Frameworks have ben developed to address these problems using principles of step-wise refinement, e.g., reasoning with tightly-coupled transition diagrams at different resolutions Sridharan et al. (2019), or combining commonsense reasoning with active learning and relational reinforcement learning to acquire knowledge Sridharan & Meadows (2018).

Using VQA as a motivating example, and building on work in cognitive systems and our prior work Riley & Sridharan (2018a), our architecture combines the complementary strengths of reasoning with commonsense knowledge, inductive learning of knowledge, and deep learning.

3 Architecture

Figure 1 is an overview of our VQA architecture, which embeds commonsense reasoning with incomplete knowledge, and inductive learning, in a deep network architecture. CNN-based feature extractors are trained to extract feature vectors from images of scenes. For each feature vector, an attempt is first made to classify it and explain the decision using non-monotonic logical reasoning. If this method fails, a decision tree is trained to classify the feature vector and explain the outcome. If logical reasoning is used for classification, it is also used to answer explanatory questions about the scene. If a decision tree is used for classification, an RNN is trained to map the decision tree output, image features, and the query, to the answer. Furthermore, decision-tree induction with training data and existing knowledge identifies previously unknown state constraints used for subsequent reasoning. We hypothesize that this architecture will make learning more time and sample efficient, and make decisions more interpretable. Due to space limitations, we briefly describe the components below.

We use three domains for evaluation. The **Structure Stability (SS)** domain (top left, Figure 2) has 2500 images of structures of simulated blocks from a physics-based simulator; the objective ois to classify structures as being stable or unstable, and to answer explanatory questions, e.g., "why is this structure unstable?" and "what should be done to make this structure stable?". The **Traffic Sign (TS)** domain (bottom left, Figure 2) uses the BelgiumTS benchmark dataset Timofte et al. (2013) with ≈ 7000 real-world images of 62 traffic signs. The objective is to classify the signs and answers questions such as "what is the sign's message?" and "how should the driver respond to this sign?". The third domain (used for planning) is introduced later.

3.1 Feature Extraction using CNNs

Input images are mapped to concise features. The selection of features is based on domain expertise, e.g., features of the SS domain include number of blocks in structure, whether the structure is on a lean etc, and features of the TS domain include primary and secondary colors and symbols, shape of the sign etc. For each feature, a simple CNN was trained and additional layers added until training accuracy converged. For more complex features, previously trained CNN models can be fine-tuned. The code for this component is in our online repository Riley & Sridharan (2018b).

3.2 Classification using Non-monotonic Logical Reasoning or Decision Trees

A class label is assigned to the extracted feature vector using one of two methods: (i) non-monotonic logical reasoning; or (ii) a learned decision tree.

ASP Reasoning with Commonsense Knowledge: ASP is a declarative language based on stable model semantics. Each literal can be true, false or unknown, and the agent reasoning with domain knowledge does not believe anything that it is not forced to believe. ASP can represent recursive definitions, defaults, causal relations, and

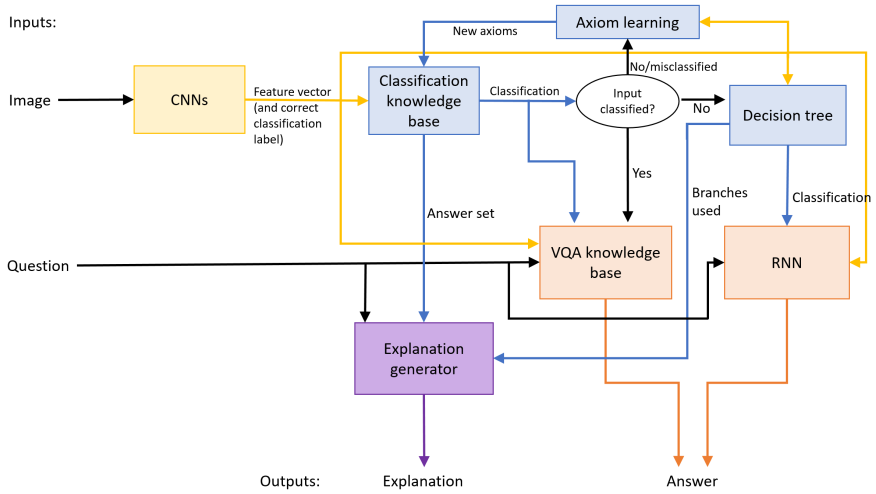

Figure 1: Overview of our architecture's components.

Figure 2: Illustrative domains: (top left) blocks in SS domain; (bottom left) traffic sign in TS domain; (right) simulated scenario in the RA domain.

language constructs difficult to express in classical logic formalisms Gelfond & Kahl (2014). ASP supports *default negation*, *epistemic disjunction*, and non-monotonic logical reasoning, i.e., it can revise previously held conclusions based on new evidence, which aids in the recovery from errors made by reasoning with incomplete knowledge.

A domain description in ASP has a *system description* $\mathscr{D}$ and a *history* $\mathscr{H}$. $\mathscr{D}$ has a *sorted signature* Σ and axioms. Σ has *basic sorts*, *statics*, i.e., domain attributes whose values do not change, *fluents*, i.e., domain attributes whose values can change over time, and *actions*. Basic sorts include *structure*, *color*, and *size* for SS domain; and *main_color*, *other_color*, *main_symbol* etc for TS domain; both domains have *step* for temporal reasoning. Statics and fluents model domain attributes, e.g., $num\_blocks(structure, num)$ and $stable(structure)$ in SS domain, and $primary\_symbol(sign, main\_symbol)$ and $primary\_color(sign, main\_color)$ in TS domain. Axioms of $\mathscr{D}$ govern dynamics; in our domains, they include:

$$stable(S) \leftarrow num\_blocks(S,2), \neg structure\_type(S, lean)$$

$$sign\_type(TS, no\_parking) \leftarrow primary\_color(TS, blue), primary\_symbol(TS, blank), cross(TS),\ shape(TS, circle)$$

History $\mathscr{H}$ is usually a record of fluents observed to be true or false at a particular time step, and the occurrence of an action at a particular time step. This notion is expanded to include default statements that are true in all but certain exceptional circumstances. e.g., "structures with two blocks of the same size are usually stable". For robotics examples, please see Sridharan et al. (2019).

Reasoning is achieved by translating the domain representation to a program $\Pi(\mathscr{D}, \mathscr{H})$ in CR-Prolog, a variant of ASP. Each *answer set* of $\Pi(\mathscr{D}, \mathscr{H})$ represents the beliefs of an agent associated with Π. Planning and diagnostics are reduced to computing answer sets of ASP programs. ASP programs for our domains are in our repository Riley & Sridharan (2018b). For the classification task in our domains, relevant literals in the answer set provide the class label and an explanation for this label. The accuracy of the decisions made depends on the accuracy and extent of the knowledge encoded, but encoding comprehensive domain knowledge is difficult. The decision of what (and how much) knowledge to encode is made by the designer.

Decision Tree Classifier: If ASP-based inference cannot assign class labels, the feature vector is mapped to a class label using a decision tree classifier learned from labeled data. Non-leaf nodes of the tree split the feature vector examples based on values of particular features. Each such node is also associated with samples that satisfy the values of the features along the path from the root node, with the leaf nodes representing class labels. We use a standard implementation of a decision tree classifier based on the Gini measure of information gain. Note that this tree's search space is limited since it only considers samples that could not be classified by ASP-based reasoning.

3.3 Answering Explanatory Questions

Existing software, controlled vocabulary, and templates of language models and parts of speech, are used to transcribe

questions to text and a relational representation, and to generate answers as text that may be converted to speech.

If ASP-based reasoning is able to classify the image feature vector, it is also used to answer questions about the underlying scene. To provide such answers, we revise the signature and axioms of $\mathscr{D}$, e.g., sorts such as *query_type* and *answer_type*, relations to represent abstract attributes, and axioms to reason with these attributes and construct answers. The answer set(s) of the corresponding program $\Pi(\mathscr{D},\mathscr{H})$ are computed and parsed to extract relevant literals that form the answer. If the decision tree is used to classify the image feature vector, an LSTM network-based RNN is trained to answer the questions based on the feature vector, class label, and a vector representing the transcribed query. To build the RNN, we start with one hidden layer and add more layers until the accuracy converges. In our domains, the RNN had as many as $26-30$ hidden layers.The code used is in our repository Riley & Sridharan (2018b).

3.4 Learning State Constraints

In many domains, the encoded knowledge is incomplete or changes over time, resulting in incorrect or sub-optimal decisions, e.g., a traffic sign can be misclassified. Our architecture supports incremental learning of domain knowledge, specifically using decision tree induction to learn state constraints. In the context of VQA, we first identify training examples that are not classified or are misclassified based on existing knowledge, and built a decision tree. Next, we identify paths in the tree supported by a sufficient number of examples; these correspond to partial state descriptions and class labels that occur frequently. These paths are used to create candidate constraints. We then generalize the candidate axioms to remove over-specifications, e.g., the first two axioms (below) generalize to the third one:

$$\neg stable(S) \textbf{ if } num\_blocks(S,3),\ base(S,wide),\ struc\_type(S,lean)$$
$$\neg stable(S) \textbf{ if } num\_blocks(S,3),\ base(S,narrow),\ struc\_type(S,lean)$$
$$\neg stable(S) \textbf{ if } num\_blocks(S,3),\ struc\_type(S,lean)$$

The candidate axioms are validated by adding them to the ASP program and testing that they do not violate any of the relevant training examples.

3.5 Planning with Domain Knowledge

We also extend reasoning to planning in the **Robot Assistant (RA)** domain, in which a simulated robot observes the domain, moves to deliver messages to people, and answers explanatory questions. Figure 2(right) shows a simulated scenario. In other work, we have coupled ASP-based reasoning with probabilistic reasoning to account for the uncertainty in sensing and actuation Sridharan et al. (2019). Here, we temporarily abstract away the probabilistic models of uncertainty, focus on the interplay between reasoning and learning, and evaluate the effect of added noise.

To support planning in the RA domain, we construct a Σ with sorts such as *place*, *robot*, and *object*; fluents such as $loc(agent,place)$ and $messagestatus(mid,person,status)$; statics such as $next\_to(place,place)$; and actions such as $move(robot,place)$ and $deliver(robot,message\_id,person)$. For ease of explanation, we assume that the locations of people are determined by external sensors, and the locations of objects are statics. Axioms of $\mathscr{D}$ include:

$$move(rob_1,L) \textbf{ causes } loc(rob_1,L)$$
$$loc(P,L) \textbf{ if } work\_place(P,L),\ not\ \neg loc(P,L)$$
$$\textbf{impossible } move(rob_1,L) \textbf{ if } loc(rob_1,L)$$

to encode causal laws, constraints, and executability conditions. After adding a goal and helper axioms, answer sets of $\Pi(\mathscr{D},\mathscr{H})$ include a plan of actions, and missing constraints can be learned as described in Section 3.4.

4 Experimental Setup and Results

We experimentally evaluated four hypotheses, i.e., that our architecture (**H1**) outperforms an architecture based on just deep networks for classification and VQA with small training datasets; (**H2**) provides intuitive answers to explanatory questions; (**H3**) uses learned constraints to improve the ability to answer questions; and (**H4**) supports planning and uses learned axioms to improve plan quality. Hypotheses $H1$, $H2$ and $H3$ are evaluated in the SS and TS domains in the context of VQA; $H4$ is evaluated in the RA domain in the context of planning and VQA. Accuracy was used as the primary performance measure. Accuracy was measured by: (a) comparing the assigned labels with the ground truth labels for classification; and (b) heuristically computing whether the answer mentions all image attributes relevant to the question posed (for VQA); relevance was established by a human expert, one of the authors of this paper. Plan quality was measured as the ability to compute minimal and correct plans that achieves the goal on execution. Two-thirds of the available data is used to train the deep networks and other models, using the remaining data for testing. For each image, we randomly chose from the suitable questions for training and testing, and report the average of multiple such trials. Also, all claims are statistically significant.

Execution Example 1 *[Question Answering, TS domain]* Consider a scenario in the TS Domain with the following exchange for a particular input (test) image.

- Classification question: "what is the sign's message?"
- Architecture's answer: "uneven surfaces ahead".
- When asked to explain the reason for this answer, the architecture identifies the features extracted: (i) triangle-shaped; (ii) main color is white and border color is red; (iii) no background image; (iv) bumpy-road symbol.

- ASP-based inference with domain knowledge and literals of image features is unable to classify the sign.
- Extracted features were processed using the trained decision tree, which only used the sign's colors to assign class label. Colors are normally insufficient for classification, but the decision tree is only trained to classify signs that cannot be classified using existing knowledge.
- The decision tree output, feature vector, and question, were processed by trained RNN to provide the answer.

For other examples such as the image of SS domain in top left of Figure 2, domain knowledge is sufficient for classification and answering questions.

4.1 Experimental Results: VQA + Learn Axiom

To evaluate $H1$ and $H2$, we ran trials in which we varied the size of the training dataset, and compared the accuracy of our architecture with a baseline CNN-RNN architecture. Due to space constraints, we only summarize VQA accuracy in Figure 3. We observe that our architecture is better than the baseline architecture based on just deep networks for small training datasets. Classification accuracy (not shown) increases with the size of the training set but VQA accuracy does not because it also depends on the complexity of the questions. The accuracy improvement is more pronounced in the more complex (TS) domain.

Next, we designed an ASP program for the SS domain with eight axioms related to stability, randomly chose four to be removed, and examined the ability to learn these axioms and use them for classification and VQA, with number of labeled training examples ranging from 100 to 2000. Since the TS domain has many more axioms and labeled examples, each experimental trial examines the effect of removing a quarter of the axioms (randomly), with the number of training examples varying from 100 to 4000. Results averaged over 30 such trials are summarized in Figure 4; the blue ("Original KB") bars represent baseline and the orange ("Learned KB") bars show results with the learned axioms. Our approach incrementally learns previously unknown axioms, and using axioms improves VQA (and classification) accuracy significantly; these results support $H3$.

4.2 Experimental Results: Learn Axiom + Plan

We evaluated the ability to learn axioms and use them for planning in the RA domain. The robot had to use domain knowledge to plan, classify, and answer questions. Results (100 trials) indicate a VQA accuracy of 82% with just 500 labeled images. We first examine an execution trace.

Execution Example 2 *[Question Answering, RA Domain]* The robot has to deliver messages from John to Sally, and return to John to answer questions.

- The robot was initially in John's office. The computed plan had the robot move through the library and the kitchen to Sally's office, deliver the message to Sally, and return to John's office through the same route.
- During plan execution, the robot captures and processes images of the scenes. After returning to John's office, the robot discusses plans, observations, and beliefs with the humans. Some statements in the exchange:
  John's question: "is Sally's location cluttered?"
  Robot's answer: "Yes".
  When asked, robot provides an **explanation** for this decision: "Sally is in her office. Objects observed are Sally's chair, desk, and computer, and a cup, chair, and plate. The room is cluttered because the cup, chair and plate are not usually in that room."

Next, we evaluated the ability to learn and use axioms. In this domain, there is default knowledge about the initial locations of people (i.e., their office) and objects, unless the defaults are negated by other knowledge or observations. Including such knowledge allows the robot to efficiently compute minimal and correct plans, e.g., when trying to deliver messages to a particular person. However, this default knowledge may not be known in advance and may change with time. In all our trials, our approach was able to accurately and efficiently learn unknown information about such defaults and their exceptions.

Finally, we ran 100 paired trials to explore the impact of learned axioms on planning. In each trial, we randomly chose a particular goal and initial conditions, and measured the ability to computer minimal and correct plans before and after learning previously unknown axioms. The validity of a plan is established by executing it in simulation. Results obtained without the learned axioms were computed as a ratio of the results with the learned axioms. Before axiom learning, the robot often explored an incorrect location (e.g., for a person) based on other considerations (e.g., distance to the room) and ended up having to replan. After learning the axioms, the robot eliminated irrelevant paths in the transition diagram from further consideration; we observe a (statistically) significant improvement in performance. For instance, in the absence of the learned axioms, the robot computes four times as many plans taking more than six times as much time in any given trial (on average) as when the learned axioms were used for reasoning. Even the time taken to compute each plan is significantly higher in the absence of the learned axioms.

5 Discussion and Conclusions

For many critical problems in robotics and AI, explainability can help identify errors, design better algorithms, and improve trust in automated reasoning and learning algorithms. In this paper, we considered VQA as a motivating example of such a problem that requires explainability in reasoning and learning. Deep networks represent state of the art for VQA, but they are computationally expensive, re-

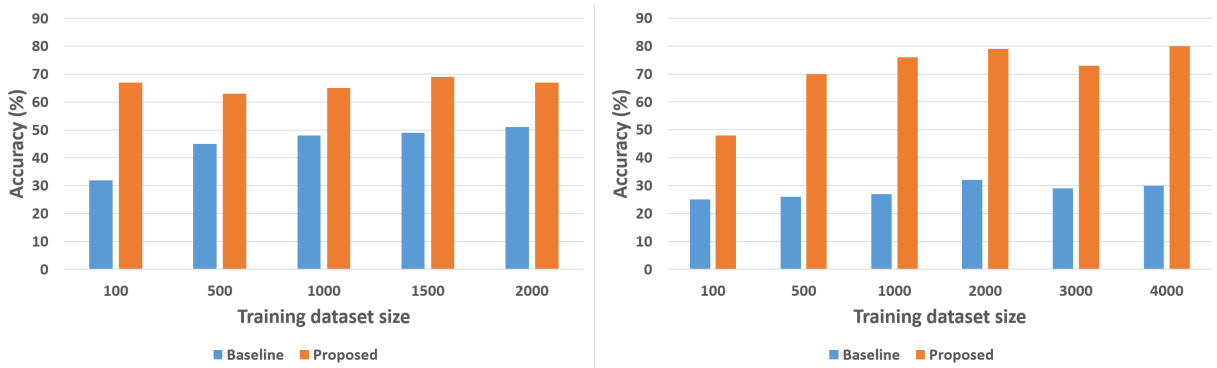

Figure 3: VQA accuracy as a function of the number of training samples in the SS domain (left) and TS domain (right).

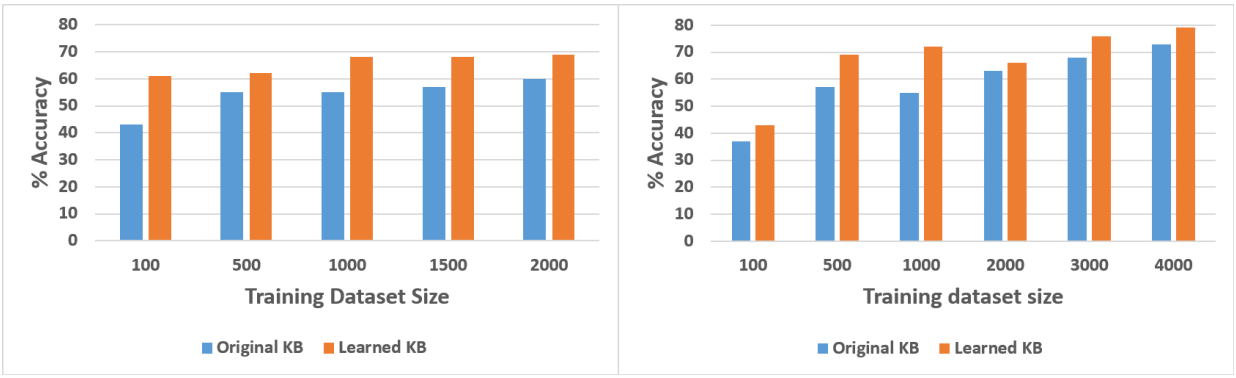

Figure 4: Comparison of VQA accuracy with and without axiom learning in the SS domain (left) and TS domain (right). Reasoning with the learned axioms improves accuracy.

quire large training datasets, and make it difficult to support explainability. Inspired by research in cognitive systems, our architecture couples representation, reasoning and interactive learning, and exploits the complementary strengths of deep learning, non-monotonic logical reasoning with commonsense knowledge, and decision tree induction. Experimental results on benchmark datasets and simulated images indicate that in comparison with baseline deep networks, our architecture provides: (i) better accuracy, sample efficiency and time complexity on classification problems; (ii) more reliable answers to explanatory questions; and (iii) support for learning unknown state constraints. Future work will further explore the use of reasoning with commonsense knowledge to direct and better understand the operation of deep network architectures, and evaluate our architecture in more complex domains.

Acknowledgements

This work was supported in part by the US Office of Naval Research Science of Autonomy award N00014-17-1-2434, and the Asian Office of Aerospace Research and Development award FA2386-16-1-4071. Opinions and conclusions reported in this paper are those of the authors.

References

Baral, C., Gelfond, M., & Rushton, N. (2009). Probabilistic Reasoning with Answer Sets. *Theory and Practice of Logic Programming*, *9*, 57–144.

Erdem, E., Gelfond, M., & Leone, N. (2016). Applications of Answer Set Programming. *AI Magazine*, *37*, 53–68.

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

Gil, Y. (1994). Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. *International Conference on Machine Learning* (pp. 87–95).

Goyal, Y., Khot, T., Stay, D. S., Batra, D., & Parikh, D. (2017). Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. *International Conference on Computer Vision and Pattern Recognition* (pp. 6325–6334).

Koh, P. W., & Liang, P. (2017). Understanding Black-box Predictions via Influence Functions. *International Conference on Machine Learning (ICML)* (pp. 1885–1894).

Law, M., Russo, A., & Broda, K. (2018). The Complexity and Generality of Learning Answer Set Programs. *Artificial Intelligence*, *259*, 110–146.

Li, Q., Fu, J., Yu, D., Mei, T., & Luo, J. (2018). *Tell-and-Answer: Towards Explainable Visual Question Answering using Attributes and Captions*. Technical report, `https://arxiv.org/abs/1801.09041`.

Malinowski, M., Rohrbach, M., & Fritz, M. (2017). Ask Your Neurons: A Deep Learning Approach to Visual Question Answering. *International Journal of Computer Vision*, *125*, 110–135.

Riley, H., & Sridharan, M. (2018a). Non-monotonic Logical Reasoning and Deep Learning for Explainable Visual Question Answering. *International Conference on Human-Agent Interaction*.

Riley, H., & Sridharan, M. (2018b). Software for paper. `https://github.com/hril230/masters_code`.

Selvaraju, R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization. *International Conference on Computer Vision* (pp. 618–626).

Sridharan, M., Gelfond, M., Zhang, S., & Wyatt, J. (2019). REBA: A Refinement-Based Architecture for Knowledge Representation and Reasoning in Robotics. *Journal of Artificial Intelligence Research*, *65*, 87–180.

Sridharan, M., & Meadows, B. (2018). Knowledge Representation and Interactive Learning of Domain Knowledge for Human-Robot Collaboration. *Advances in Cognitive Systems*, *7*, 69–88.

Timofte, R., Mathias, M., Benenson, R., & Gool, L. V. (2013). Traffic Sign Recognition - How far are we from the Solution? *International Joint Conference on Neural Networks (IJCNN)* (pp. 1–8).

Wagner, M., Basevi, H., Shetty, R., Li, W., Malinowski, M., Fritz, M., & Leonardis, A. (2018). Answering Visual *What-If* Questions: From Actions to Predicted Scene Descriptions. *Visual Learning and Embodied Agents in Simulation Environments (VLEASE) Workshop at ECCV*.

Wang, P., Wu, Q., Shen, C., van den Hengel, A., & Dick, A. R. (2017). Explicit Knowledge-based Reasoning for Visual Question Answering. *International Joint Conference on Artificial Intelligence*.

Yang, Z., He, X., Gao, J., Deng, L., & Smola, A. J. (2016). Stacked Attention Networks for Image Question Answering. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 21–29).