AQuA: ASP-Based Visual Question Answering

Kinjal Basu([✉]), Farhad Shakerin, and Gopal Gupta

The University of Texas at Dallas, Richardson, USA
{Kinjal.Basu,Farhad.Shakerin,gupta}@utdallas.edu

Abstract. AQuA (ASP-based Question Answering) is an Answer Set Programming (ASP) based visual question answering framework that truly "understands" an input picture and answers natural language questions about that picture. The knowledge contained in the picture is extracted using YOLO, a neural network-based object detection technique, and represented as an answer set program. Natural language processing is performed on the question to transform it into an ASP query. Semantic relations are extracted in the process for deeper understanding and to answer more complex questions. The resulting knowledge-base—with additional commonsense knowledge imported—can be used to perform reasoning using an ASP system, allowing it to answer questions about the picture, just like a human. This framework achieves 93.7% accuracy on CLEVR dataset, which exceeds human baseline performance. What is significant is that AQuA translates a question into an ASP query without requiring any training. Our framework for Visual Question Answering is quite general and closely simulates the way humans operate. In contrast to existing purely machine learning-based methods, our framework provides an explanation for the answer it computes, while maintaining high accuracy.

Keywords: Answer set programming · Visual question answering · Commonsense reasoning · Natural language understanding

1 Introduction

Answering questions about a given picture, or Visual Question Answering (VQA), is a long-standing goal of Artificial Intelligence research. To answer questions about a picture, humans generally first recognize the objects in the picture, then they reason with the questions asked using their commonsense knowledge. To be effective, we believe a VQA system should work in a similar way. Thus, to perceive a picture, ideally, a system should have intuitive abilities like object and attribute recognition and understanding of spatial-relationships. To answer questions, it must use reasoning. Natural language questions are complex and ambiguous by nature, and also require commonsense knowledge for their interpretation. Most importantly, reasoning skills such as counting, inference, comparison, etc., are needed to answer these questions.

To build VQA systems, researchers have created several datasets such as [3,12,13,22,25,33]. However, according to Johnson et al. [9] approaches based on neural network-based models tend to "cheat" while answering the questions by exploiting the correlations between word occurrences instead of truly understanding the content. CLEVR [9] is a recent VQA dataset of 100k rendered images along with complex and challenging questions over them. The images depict simple 3D shapes which makes recognition task easy, however, to answer the questions, a system should be able to perform logical reasoning over object attributes and their relationships.

The experiments conducted by *Johnson et. al.* in [9] suggest that approaches based on end-to-end training of neural network models perform poorly on CLEVR, no matter how sophisticated the architecture is. The state-of-the-art neural-based VQA systems incorporate convolutional layers, recurrent/LSTM models, and attention networks. Also, some models translate questions into intermediate functional units that are either programmed or are trainable neural modules.

Surveys of state-of-the-art neural-based VQA systems can be found elsewhere [27,30]. The main point to note is that none of these systems follow the methodology that humans use: recognize images via pattern recognition while answer questions through reasoning. In this paper we propose the AQuA framework that attempts to emulate a human for VQA. We strongly believe that AQuA's method is the most effective approach for VQA and does not suffer from the flaws (discussed later) found in all purely neural network-based approaches. What is more, AQuA can provide full justification for the answers it computes, something that is not possible for neural network-based methods because of their "black box" nature.

AQuA replicates a human's VQA behavior by incorporating commonsense knowledge and using ASP for reasoning. VQA in the AQuA framework employs the following sources of knowledge: (i) knowledge about objects extracted using the YOLO algorithm [21], (ii) semantic relations extracted from the question, (iii) query generated from the question, and (iv) commonsense knowledge. AQuA runs on the query-driven, scalable s(ASP) [15] answer set programming system that can provide a proof tree as a justification for the query being processed.

AQuA processes and reasons over raw textual questions and does not need any annotation or generation of function units such as what is employed by several approaches proposed for the CLEVR dataset [10,29,32]. Also, instead of predicting an answer, AQuA augments the parsed question with commonsense knowledge to truly understand it and to compute the correct answer (e.g., it understands that *block* means *cube*, or *shiny object* means *metal object*).

This paper makes the following novel contributions: (i) presents a fully explainable framework to handle VQA that outperforms existing neural-based systems in terms of accuracy and explainability, (ii) demonstrates that a general (i.e., not domain specific), scalable VQA system can be built without training, (iii) understands textual knowledge with the help of commonsense knowledge, and (iv) provides a method that guarantees a correct answer as long as the objects in the picture are recognized correctly and syntactic & semantic processing of the question yields a correct query.

2 Background

Answer Set Programming (ASP): An answer set program is a collection of rules of the form -

$$l_0 \leftarrow l_1,\ \ldots,\ l_m,\ not\, l_{m+1},\ \ldots,\ not\, l_n.$$

Classical logic denotes each l_i is a literal [4]. In an ASP rule, the left hand side is called the *head* and the right-hand side is the *body*. Constraints are ASP rules without *head*, whereas facts are without *body*. The variables start with an uppercase letter, while the predicates and the constants begin with a lowercase. We will follow this convention throughout the paper. The semantics of ASP is based on the stable model semantics of logic programming [5]. ASP supports *negation as failure* [4], allowing it to elegantly model common sense reasoning, default rules with exceptions, etc., and serves as the secret sauce for AQuA's sophistication.

s(ASP) System: s(ASP) [15] is a query-driven, goal-directed implementation of ASP. This is indispensable for automating commonsense reasoning, as traditional grounding and SAT-solver based implementations of ASP are not scalable. There are three major advantages of using s(ASP): 1. s(ASP) does not ground the program, which makes AQuA framework fast and scalable, 2. it only explores the parts of the knowledge base that are needed to answer a query, and 3. it provides justification tree for an answer.

YOLO Object Detection Model: Redmon et. al. [20,21] proposed a convolutional neural network architecture to detect multiple objects from different categories in a given image and to simultaneously draw bounding boxes around them. The novelty of this method lies in the fact that it combines the classification and regression tasks together by only using the extracted features from deep convolutional layers in a single step which makes real-time predictions possible.

The AQuA framework utilizes YOLO to extract characteristics of interest in the form of logical predicates from images. These predicates are then used by ASP engine to perform all sorts of reasoning on different relations between objects. In CLEVR domain these characteristics are shape, size, material, color, and spatial relationships between objects such as front, left, right, etc. Figure 3 shows an example of the predicates found by YOLO that are used by the s(ASP) engine to answer questions.

Stanford CoreNLP Tools: Stanford CoreNLP [14] is a set of tools for natural language processing. AQuA only uses *Parts-of-Speech (POS) tagger* and *Dependency Parser* from this set. Based on the context, POS tagger generates the necessary parts of speech such as noun, verb, adjective, etc., for each question. It identifies the question type (e.g., what, where, how) and disambiguated words (e.g., *block* as a *verb* vs. *block* as a *noun*). On the other hand, dependency graph provides the grammatical relations between words in the sentence. Dependency relations follow enhanced universal dependency annotation [24]. Figure 1 shows an example of POS tagging and dependency graph of a question.

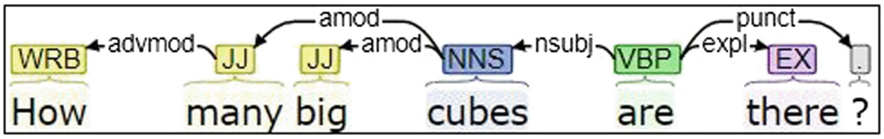

Fig. 1. Example of POS tagging and dependency graph

3 Technical Approach

In this section, we propose AQuA, a framework to perform visual question answering following a process used by us humans. AQuA represents knowledge using ASP paradigm and it is made up of five modules that perform the following tasks: (i) object detection and feature extraction using the YOLO algorithm [21], (ii) preprocessing of the natural language question, (iii) semantic relation extraction from the question, (iv) Query generation based on semantic analysis, and (v) commonsense knowledge representation. AQuA runs on the query-driven, scalable s(ASP) [15] answer set programming system that can provide a proof tree as a justification for a query being processed.

Figure 2 shows AQuA's architecture. The five modules are labeled, respectively, YOLO, Preprocessor, Semantic Relation Extractor (SRE), Query Generator, and Commonsense Knowledge.

3.1 Preprocessor

This module extracts information from the question by using Stanford CoreNLP parts-of-speech (POS) tagger and dependency graph generator. The lemma and the POS are fetched for every word along with the dependency graph of the sentence. Using this information, AQuA translates a question to a sequence of predicates that would encode the natural language question as an ASP query. Also, this information is used to infer the question type: *how_many*, *what_number*, *is_there*, etc., using which the answer types (such as *boolean*, *numeric*, *value*, etc.) are determined. The output of the Preprocessing module will be consumed by the Query Generator and the Semantic Relation Extraction (SRE) modules.

AQuA transforms natural language questions to a logical representation before feeding it to the ASP engine. This logical representation module is inspired by Neo-Davidsonian formalism [2], where every event is recognized with a unique identifier. Similarly, AQuA maintains an identifier for each word (a simple position index in the question). It identifies each object, even if there are two objects

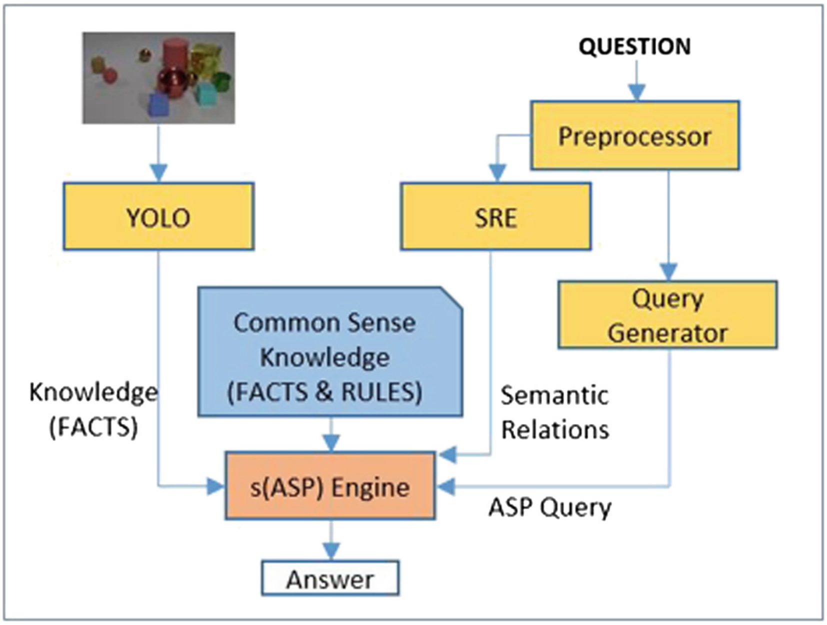

Fig. 2. System architecture

with the same name (e.g., *"How many matte blocks are behind the red block?"*, here question refers to two different blocks: the *queried object* and the *referenced object*).

3.2 SRE: Semantic Relations Extractor

Semantic relation labeling is the process of assigning relationship labels to two different phrases in a sentence based on the context. To extract semantic relations from the passage, we have created default logic rules with exceptions [26]. These rules utilize the POS and the dependency graph of the sentence from the Preprocessing step. To understand the CLEVR dataset questions, AQuA requires two types of semantic relations (i.e., *quantification* and *property*) to be extracted (if they exists) from the questions.

Quantification: In the AQuA framework, all the existential questions are considered as a special type of numeric comparison questions. Considering all the filters given in the question for the queried object, the ASP engine counts the number of queried object from the picture and compares the object count with the number given in the question. The compared number from the question is captured using *quantification* semantic relation in the form of *quantification(number, object)*. For example, *quantification(1, cube_4)* semantic relation (the suffix *4* is the identifier of the word *cube*) is extracted from the question

"*Are there any cubes?*". Quantification relation also helps in covering existential questions beyond CLEVR dataset such as *"Are there greater than five objects?"*

Property: In the CLEVR dataset questions, objects (POS: noun) are accompanied with none/one/many attribute values (POS: adjective). ASP engine uses these values as filters to search the object(s) in a given picture. Therefore, to capture this object-value pairs, AQuA extracts the *property* relationship in the form of *property(value, object)* using the *amod* relations expressed in the dependency graph. For example, AQuA automatically adds *property(big_4, ball_6)* and *property(red_5, ball_6)* semantic relation to the knowledge base for the question *"Is there a big red ball?"*

3.3 Query Generator

Based on the knowledge from a question, AQuA generates a list of ASP clauses with the query, which runs on the s(ASP) engine to find the answer. In general, questions with one-word answer are categorized into: (i) yes/no questions, and (ii) attribute/value questions. In particular, CLEVR question bank has yes/no questions in the form of existential questions (*Is there a red ball?*) and those that compare values (*Is the ball same color as the cube?*), and attribute/value questions in the form of counting objects (*How many red balls are there?*) and querying object attributes (*What is the color of the shiny cylinder?*). To handle each type of question, AQuA generates a set of ASP clauses. For example, the following set of clauses are generated to tackle the yes/no question: *"Is there a big ball?"*

```
(1) query(Q, A) :- question(Q), answer(Q, A).
(2) answer('is there a big ball ?', yes) :- find_ans ('is there a big ball ?').
(3) answer('is there a big ball ?', no) :- not find_ans('is there a big ball ?').
(4) find_ans(Q) :- question(Q), find_all_filters(ball, 5, L),
                   list_object(L, Ids), list_length(Ids, C),
                   quantification(N, ball_5), gte (C, N) .
(5) question('is there a big ball ?').
```

In the code above, clause (1) captures the answer of the question (Q) in the variable A. Clause (2) says that if *find_ans* predicate succeeds, return *yes* to the answer variable A from clause (1). Whereas, if *find_ans* fails, clause (3) produces *no* as an answer to the clause (1)'s answer variable A. If all the sub-goals from the body of the clause (4) succeed then *find_ans* becomes *true*, *false* otherwise. Lastly, fact (5) represents the natural language question.

Similar to the yes/no question, the following set of clauses are generated for the attribute/value question: *"What color is the cube ?"*

```
(1) query(Q, A) :- question(Q), answer(Q, A).
(2) answer('what color is the cube ?', A) :-
                        find_ans('what color is the cube ?', A).
(3) find_ans(Q, A) :- question(Q), find_all_filters(cube, 5, L),
                        list_object(L, Ids), get_att_val(Ids, color, A) .
(4) question('what color is the cube ?').
```

Here, clauses (1) and (4) are identical to clauses (1) and (5) for the yes/no question discussed above. Clause (2) stores the answer in variable A and passes it to clause (1). Clause (3) captures the answer in the head variable A that is defined in the body.

3.4 Commonsense Knowledge

Similar to a human, AQuA requires commonsense knowledge to correctly compute answers to questions. For the CLEVR dataset questions, AQuA needs to have commonsense knowledge about properties (e.g., color, size, material), directions (e.g., left, front), and shapes (e.g., cube, sphere). AQuA will not be able to understand question phrases such as *'... red metal cube ...'*, unless it knows *red* is a color, *metal* is a material, and *cube* is a shape. In some cases, AQuA requires deeper knowledge to understand the questions. For example, phrase like *'... shiny object ...'* requires two step inferences: (i) *shiny* refers to *metal*, and, (ii) *metal* is a *material*. Commonsense knowledge can be categorized in two different parts based on the type: (i) commonsense facts, which are grounded and always true. (ii) commonsense rules, which are required for reasoning tasks (i.e., counting, spatial reasoning, etc.).

Commonsense Facts: For CLEVR, commonsense facts are of two types: (i) attribute values (e.g., *red* is a *color*, *cube* is a *shape*), and, (ii) similar values (e.g., *block* and *cube*). Thus, these facts are represented using two types of relational predicates:

is_property(V, A): This relationship stores the attribute value pair, V and A. For example, *red* is a *color* will be represented as *is_property(red, color).*

is_similar(X1, X2): This relationship stores two similar words, *X1* and *X2*, one of which can be used in place of the other without changing the meaning. For example, *big* is similar to *large* is represented as *is_similar(big, large).*

We humans incrementally augment/refine our commonsense knowledge from information we encounter in everyday life. Thus, we add/remove facts as our knowledge evolves. If we want to add another shape (such as *pyramid*), we only need to add another *is_property* relation. The system will automatically incorporate this knowledge and be able to handle questions involving *pyramids.*

Commonsense Rules: Commonsense rules deal with all the reasoning tasks that AQuA needs to handle. Depending on the CLEVER dataset question, AQuA executes several sub-tasks (e.g., list-length, list-union, numeric-comparison such as greater-than-equal, less-than) underneath the primary reasoning tasks (e.g., sorting a list of object from left to right based on its spatial position) required to answer a question. Following is an example of a set of rules required to filter a list of objects that match a specific attribute-value pair. The code is straightforward as it recursively traverses the list.

```
(1) filter(_, _, [], []).
(2) filter(Att, Val, [Id | T1], [Id | T2]) :- property(Id, Att, Val),
                                               filter(Att, Val, T1, T2).
(3) filter(Att, Val, [Id | T1], T2) :- not property(Id, Att, Val),
                                       filter(Att, Val, T1, T2).
```

3.5 ASP Engine

ASP engine is the brain of our system. All the knowledge (image representation, commonsense knowledge, semantic relations) and the query in ASP syntax are executed using the query-driven s(ASP) system. Depending on the query it only explores part of the knowledge which is needed to compute the answer. Also, s(ASP) does not do unnecessary grounding which saves lots of memory. Due to this benefit, we are able to test all the questions in the same iteration without facing any memory overflow issues.

An AQuA query in s(ASP) is always of the form *"?- query (Q, A)"* where Q represents the question and A the computed answer.

4 Experiments and Results

AQuA is a novel, general framework for visual QA. Questions are answered in this framework through the use of commonsense reasoning rather than with machine learning. If the questions are correctly parsed and correctly converted into a query, the answer computed by AQuA is always correct. Unlike other neural-based systems, AQuA does not need training over questions. Thus, the question bank provided in the dataset for testing is of no use to AQuA. Instead, the validation QA set is directly used to compute an answer and compare it to the actual answer given in the validation QA dataset to generate the results.

We tested our AQuA framework on the CLEVR dataset. The validation set for CLEVR contains 149,991 questions and 15,000 images. We simplified the testing process slightly by limiting the question length to 15 words. We built this system for natural language questions answering over the CLEVR image set using the AQuA framework without focusing on any particular question type. As mentioned earlier, we also did not use the functional programs/units given for each question in the validation set since AQuA performs its own parsing and semantic analysis. We ran AQuA on 45,157 questions that fit the selection criteria imposed (e.g., questions with 15 words or less) to generate ASP queries. An accuracy of **93.7%** was achieved with 42,314 correct answers. This performance is beyond the average human accuracy [9].

Table 1. Performance results

Question type		Accuracy (%)	
Exist		**96**	
Count		**91.7**	
Compare value	Shape	87.42	**92.89**
	Color	94.32	
	Size	92.17	
	Material	96.14	
Compare integer	Less than	97.7	**98.05**
	Greater than	98.6	
	Equal	NA*	
Query attribute	Shape	94.01	**94.39**
	Color	94.87	
	Size	93.82	
	Material	94.75	

* Equality questions are minuscule in number so currently ignored.

We have extensively studied the 2,843 questions that produced erroneous results. 2,092 questions out of 2,843 do not match the correct answer and other 751 questions throw ASP exceptions. Our manual analysis showed that mismatch happens mostly because of errors caused by the YOLO module: failing to detect a partially visible object, wrongly detecting a shadow as an object, wrongly detecting two overlapping objects as one, etc. Eliminating these errors through manual intervention resulted in another 2,626 questions out of the 2,843 questions being answered correctly. Only 217 incorrectly answered questions remained. Further analysis indicated that these could be attributed to wrong parsing or oversimplified spatial reasoning. As an example of parsing error, *block* sometime is parsed as a *verb* instead of a *noun*. With respect to oversimplification of spatial reasoning, note that objects in CLEVR have 3D shapes, but we only considered X and Y coordinates to calculate relative positioning of referenced objects (e.g., for *behind the block* concept). In the future, we can eliminate such errors by using more sophisticated spatial reasoning.

Even with these errors, our results are very promising as AQuA outperforms many state-of-the-art ML based methods on the CLEVR dataset illustrating the advantage of our approach that uses both machine learning and reasoning. Quantitative results for each question type are summarized in Table 1.

5 Example

To illustrate the AQuA framework further, we next discuss a full-fledged VQA example showing the data-flow and intermediate outputs from each step.

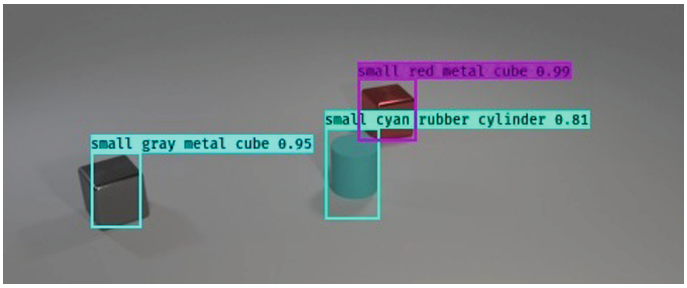
small red metal cube 0.99
small cyan rubber cylinder 0.81
small gray metal cube 0.95

Fig. 3. Object detection using YOLO.

Image: Figure 3 shows an image from the validation set. Objects in the image are detected and labeled with its attribute. The decimal number at the end of each label shows the correctness probability of that object using the YOLO model.

Object representation: All the detected objects from the Fig. 3 are converted to ASP facts. Every object has a unique id number. Five attributes are detected for each object: *shape, color, material, size, and coordinates.*

```
object(1, cylinder, cyan, rubber, small, 246, 185).
object(2, cube, red, metal, small, 270, 130).
object(3, cube, gray, metal, small, 79, 191).
```

Question: Now, from the question set we have the following question for the image shown in Fig. 3.

'Is there a matte thing in front of the metallic thing behind the gray cube?'

Parsed Output: The question is parsed using the Stanford CoreNLP parser to get the *lemma, POS, and dependency graph.* Figure 4 shows the parser's output.

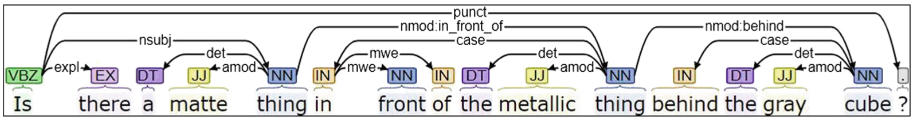
punct
nmod:in_front_of
nsubj
case
nmod:behind
case
det
det
det
expl
amod
mwe
mwe
amod
amod
VBZ EX DT JJ NN IN NN IN DT JJ NN IN DT JJ NN .
Is there a matte thing in front of the metallic thing behind the gray cube ?

Fig. 4. POS tagging and dependency graph.

Semantic Relations: The following semantic relations are next extracted by applying default ASP rules. The word index is given for each word followed by a '_' (underscore). The lemma *'thing'* occurs twice in the question each occurrence representing a different object. Thus, the two occurrences are given distinct identifiers.

```
quantification(1, thing_5).
property(matte_4, thing_5).
property(metallic_10, thing_11).
property(gray_14, cube_15).
```

Commonsense Knowledge: The following figure shows the necessary commonsense knowledge needed to understand this question.

```
is_property(cube, shape).
is_property(cylinder, shape).
is_property(metal, material).
is_property(rubber, material).
is_property(small, size).
is_property(red, color).
is_property(cyan, color).
is_property(gray, color).
is_similar(matte, rubber).
```

Query: This question involves transitive referential reasoning: (i) *matte thing in front of the metallic thing*, and (ii) *metallic thing behind the gray cube*. Our framework can handle any level of transitivity. The query generated for this question is shown below. For these type of questions, a human will always start reasoning from the final referenced object. Similarly, our automatically generated query will perform spatial reasoning starting out from the *cube*. After resolving all the reference relations, it checks whether the desired object exists or not.

```
find_ans(Q) :- question(Q), get_all_id(L1), find_all_filters(cube, 15, L2),
                filter_all(L2, L1, [H0|T0]), get_behind_list(H0, L3),
                find_all_filters(thing, 11, L4), filter_all(L4, L3, [H1|T1]),
                get_front_list(H1, L5), find_all_filters(thing, 5, L6),
                filter_all(L6, L5, Ids), list_length(Ids, C),
                quantification(N, thing_5), gte(C, N).
```

Answer: The given answer matches with actual answer, which is *yes*.

Justification: As stated earlier, because of its query driven nature, s(ASP) will automatically generate a justification for any answer it computes [15]. A justification essentially is an explanation of how the answer is computed. Unfortunately, details are omitted due to lack of space.

6 Discussion

Our goal is to create a VQA framework that can answer questions based on logical reasoning, in a manner similar to how humans answer questions. Humans use pattern recognition for understanding pictures (akin to using neural networks to detect and localize objects in a picture) and commonsense reasoning for answering question about them. We believe that this is the most effective way for VQA. Our experiments demonstrate that if knowledge from the picture is represented correctly, and the question is parsed/processed properly, the AQuA framework will not fail in obtaining the correct answer. Our goal is to reach 100% accuracy (simulate an unerring human). The object detection model we use has around **5.3%** error rate. Upgrading to state-of-the-art neural-based object detection models [8,32] can increase our accuracy to nearly 100%. Similarly, if the error rate for Stanford CoreNLP tools is reduced, we can reach even higher accuracy.

AQuA has many other advantages as well, which other machine learning-based methods do not. As explained earlier, the AQuA framework provides explanation for a computed answer. Explainability is an important consideration for any intelligent system. The AQuA framework is also *interpretable*. Unlike neural network based methods, our AQuA framework is completely transparent. This means that the system can be well understood, debugged and improved. The AQuA framework is quite general, i.e., it can be applied to any type of VQA with reasonable effort. Thus, our framework can be used for any other visual question answering domain with minimal changes, unlike current machine learning approaches. A particular machine learning model can only work well in a specific narrowed down domain like CLEVR. But, that model cannot be used in any other similar domain without training it again with new data or without transferring the learning from another model. In contrast, the AQuA framework based on commonsense reasoning can be applied to other domains without much change.

Also, our approach is incremental in nature. AQuA can be easily expanded with more question types which will lead to greater accuracy. On the contrary, expanding the capabilities of a machine learning system often requires hyper-parameter tuning, which often results in reduced accuracy.

Table 2. Question type wise summarized result from various state-of-the-art neural-network based model for CLEVR

Method	Count	Exist	Compare number	Compare attribute	Query attribute	Overall
Humans [10]	86.7	96.6	86.4	96.0	95.0	92.6
CNN+LSTM+SAN [10]	59.7	77.9	75.1	70.8	80.9	73.2
N2NMN [7]	68.5	85.7	84.9	88.7	90.0	83.7
Dependency Tree [1]	81.4	94.2	81.6	97.1	90.5	89.3
CNN+LSTM+RN [23]	90.1	97.8	93.6	97.1	97.9	95.5
IEP [10]	92.7	97.1	98.7	98.9	98.1	96.9
CNN+GRU+FiLM [19]	94.5	99.2	93.8	99.0	99.2	97.6
DDRprog [29]	96.5	98.8	98.4	99.0	99.1	98.3
MAC [8]	97.1	99.5	99.1	99.5	99.5	98.9
TbD+reg+hres [16]	97.6	99.2	99.4	99.6	99.5	99.1
NS-VQA [32]	99.7	99.9	99.9	99.8	99.8	99.8

7 Contribution and Related Works

The main contribution of this paper is an effective, efficient, and robust framework to handle any type of visual question answering. This framework is interpretable, expandable, scalable and explainable.

Significant research has been done on VQA, however, most of the recent work has been based on using neural network technology. For example, Stacked Attention Networks (CNN + LSTM + SA) [31], Relation Networks (CNN + LSTM + RN) [23], and Feature-wise Linear Modulation (CNN + GRU + FiLM) [19] combine the CNN-extracted image features with LSTM-extracted question features and pass them through multi-layer perceptron network. N2NMN [7], Dependency Tree [1] and TbD+reg+hres [16] assemble a graph of trained neural modules on the fly, each responsible for performing a single unit of computation to answer a question. IEP [10], DDRprog [29] and NS-VQA [32] construct intermediate functional units that unlike N2NMN are handcrafted programs. The latter incorporates segmentation techniques to achieve more accurate vision results. MAC [8] proposes differentiable reasoning units of recurrent neural network that would decompose the reasoning task to multiple small steps. A common issue in all these systems is that they heavily rely on supervised training which is computationally expensive and requires huge amount of annotated data. Moreover, the black-box nature of neural networks does not allow any justification as to why a model arrives at a certain answer. With the exception of TbD+reg+hres, which provides partial transparency via visualization of attention regions, other proposals are not explainable. In addition, a mechanism to incorporate commonsense knowledge is completely absent from all neural-based proposals.

Unlike AQuA, in many practical question-answering situations, it is impossible to understand the semantics of the question without making background assumptions from a commonsense knowledge base. Bypassing the natural language processing part, neural-based models are trained on domain-specific questions to map a question to a sequence of functional units. Thus, these models are not generalized enough to be trivially applied to other domains. Furthermore, they even fail to answer questions from the same domain in case of natural language sophistication (e.g., syntactic variation, co-referenced words)

Table 2 shows the summarized result for various state of the art neural network models for CLEVR dataset. First, neural-based methods for VQA have several deficiencies: they require training even for the natural language part (questions), models are inscrutable blackboxes, impossible to debug, etc. Use of training not only involves expensive computation, it requires laborious annotation of the data for questions. In contrast, the AQuA framework emulates methods employed by humans and answers questions through commonsense reasoning while providing explanation for the answer. Second, many of the methods such as NS-VQA [32] have more than 99% accuracy, however, these results are misleading as these methods are not general. They construct intermediate functional units that are handcrafted programs. In contrast, no such handcrafting is needed in AQuA. Commonsense knowledge is accumulated and can be transferred from one domain to another. Third, neural-based systems are not composable in the sense that if we introduce a new shape in the CLEVR dataset (e.g., a pyramid), then they will have to be retrained from scratch. In contrast, in AQuA all we have to do is to add small amount of new knowledge (e.g., a pyramid is also a shape) for the system to answer questions about the new object added.

8 Future Work

One obvious future work will be to be able to handle a broader class of questions, i.e., be able to parse them and turn them into ASP queries that are correctly answered. Also, as discussed earlier, we can use better, state-of-the-art neural network models for object detection that will almost remove every error from the picture processing end.

All the natural language or visual question answering systems suffer from NLP errors such as parsing issues (wrong parsing). We have used Stanford CoreNLP parser, which shows many wrong POS tagging and dependency graph errors. Researchers have created many other state-of-the-art parsers like Spacy [6], dependency parsing from AllenNLP [11], etc. In the future, we can have a voting mechanism between different parsers thereby increasing our accuracy.

For CLEVR dataset, only a limited amount of commonsense knowledge is needed: only few commonsense facts (about *color*, *shape*, *material*, *size*, and *similarities*) and commonsense reasoning rules (about *counting*, *comparing*, *sorting*, and *spatial relationships*) are needed. Reasoning rules are general and can be used for other datasets. However, commonsense facts are very specific to this dataset. Currently, we are manually putting the commonsense facts in ASP format. As future work, we plan to automate the commonsense facts generation process: based on the context, AQuA will fetch the knowledge from online sources such as WordNet [17], ConceptNet [28], etc., and will represent it in ASP format [18]. This will make the system more robust and flexible. Adding more commonsense knowledge and improving the ability to handle a broader class of questions is akin to a human acquiring more knowledge and learning to answer more complex questions. In this regard our approach is identical to how humans do VQA and, indeed, we are exploring practical and more advanced applications of the AQuA framework.

9 Conclusion

We presented an ASP-based VQA framework called AQuA and applied it to the CLEVR dataset with excellent results. AQuA automatically transforms visual knowledge to logical facts using the YOLO model. We also showed how AQuA automatically crafts the ASP query from a natural language question. We used the query driven s(ASP) engine to perform commonsense reasoning to obtain the answer. Our approach always finds the correct answer, if adequate knowledge is present and the natural language analysis works correctly. The AQuA framework not only gives promising results on the CLEVR dataset, it also enjoys other benefits such as explainability, generalizability, and interpretability.

Acknowledgement. We are indebted to Dhruva Pendharkar for his early work on natural language question answering. Thanks also to Sarat Varanasi for discussion and help. Authors gratefully acknowledge support from NSF grants IIS 1910131 and IIS 1718945.

References

1. Cao, Q., Liang, X., Li, B., Li, G., Lin, L.: Visual question reasoning on general dependency tree. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7249–7257 (2018)
2. Davidson, D.: Inquiries into Truth and Interpretation: Philosophical Essays, vol. 2. Oxford University Press, Oxford (2001)
3. Gao, H., Mao, J., Zhou, J., Huang, Z., Wang, L., Xu, W.: Are you talking to a machine? Dataset and methods for multilingual image question. In: NIPS 2015, pp. 2296–2304 (2015)
4. Gelfond, M., Kahl, Y.: Knowledge Representation, Reasoning, and the Design of Intelligent Agents: The Answer-Set Programming Approach. Cambridge University Press, Cambridge (2014)
5. Gelfond, M., Lifschitz, V.: The stable model semantics for logic programming. In: ICLP/SLP, vol. 88, pp. 1070–1080 (1988)
6. Honnibal, M., Montani, I.: spaCy 2: natural language understanding with bloom embeddings, convolutional neural networks and incremental parsing. **7** (2017, to appear)
7. Hu, R., Andreas, J., Rohrbach, M., Darrell, T., Saenko, K.: Learning to reason: end-to-end module networks for visual question answering. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 804–813 (2017)
8. Hudson, D.A., Manning, C.D.: Compositional attention networks for machine reasoning. arXiv preprint arXiv:1803.03067 (2018)
9. Johnson, J., Hariharan, B., van der Maaten, L., Fei-Fei, L., Lawrence Zitnick, C., Girshick, R.: CLEVR: a diagnostic dataset for compositional language and elementary visual reasoning. In: IEEE CVPR 2017, pp. 2901–2910 (2017)
10. Johnson, J., et al.: Inferring and executing programs for visual reasoning. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2989–2998 (2017)
11. Joshi, V., Peters, M., Hopkins, M.: Extending a parser to distant domains using a few dozen partially annotated examples. arXiv preprint arXiv:1805.06556 (2018)
12. Krishna, R., et al.: Visual genome: connecting language and vision using crowdsourced dense image annotations. Int. J. Comput. Vision **123**(1), 32–73 (2017)
13. Malinowski, M., Fritz, M.: A multi-world approach to question answering about real-world scenes based on uncertain input. In: NIPS 2014, pp. 1682–1690 (2014)
14. Manning, C.D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S.J., McClosky, D.: The Stanford CoreNLP natural language processing toolkit. In: ACL System Demonstrations, pp. 55–60 (2014)
15. Marple, K., Salazar, E., Gupta, G.: Computing stable models of normal logic programs without grounding. arXiv:1709.00501 (2017)
16. Mascharka, D., Tran, P., Soklaski, R., Majumdar, A.: Transparency by design: closing the gap between performance and interpretability in visual reasoning. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 4942–4950 (2018)

17. Miller, G.A.: WordNet: a lexical database for english. Commun. ACM **38**(11), 39–41 (1995). https://doi.org/10.1145/219717.219748
18. Pendharkar, D., Gupta, G.: An ASP based approach to answering questions for natural language text. In: Alferes, J.J., Johansson, M. (eds.) PADL 2019. LNCS, vol. 11372, pp. 46–63. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-05998-9_4
19. Perez, E., et al.: FiLM: visual reasoning with a general conditioning layer. In: AAAI (2018)
20. Redmon, J., Divvala, S.K., Girshick, R.B., Farhadi, A.: You only look once: unified, real-time object detection. In: CVPR, pp. 779–788. IEEE Computer Society (2016)
21. Redmon, J., Farhadi, A.: YOLOv3: an incremental improvement. arXiv preprint arXiv:1804.02767 (2018)
22. Ren, M., Kiros, R., Zemel, R.: Exploring models and data for image question answering. In: NIPS 2015, pp. 2953–2961 (2015)
23. Santor, A., et al.: A simple neural network module for relational reasoning. In: NIPS 2017, pp. 4967–4976 (2017)
24. Schuster, S., Manning, C.D.: Enhanced English universal dependencies: an improved representation for natural language understanding tasks. In: LRED 2016, pp. 2371–2378 (2016)
25. Shah, S., Mishra, A., Yadati, N., Talukdar, P.P.: KVQA: knowledge-aware visual question answering. In: AAAI (2019)
26. Shakerin, F., Salazar, E., Gupta, G.: A new algorithm to automate inductive learning of default theories. TPLP **17**(5–6), 1010–1026 (2017)
27. Shrestha, R., Kafle, K., Kanan, C.: Answer them all! Toward universal visual question answering models. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 10472–10481 (2019)
28. Speer, R., Chin, J., Havasi, C.: ConceptNet 5.5: an open multilingual graph of general knowledge. In: Proceedings AAAI, pp. 4444–4451 (2017)
29. Suarez, J., Johnson, J., Li, F.F.: DDRprog: a CLEVR differentiable dynamic reasoning programmer. arXiv preprint arXiv:1803.11361 (2018)
30. Wu, Q., Teney, D., Wang, P., Shen, C., Dick, A., van den Hengel, A.: Visual question answering: a survey of methods and datasets. Comput. Vis. Image Underst. **163**, 21–40 (2017)
31. Yang, Z., He, X., Gao, J., Deng, L., Smola, A.J.: Stacked attention networks for image question answering. CVPR, pp. 21–29 (2015)
32. Yi, K., et al.: Neural-symbolic VQA: disentangling reasoning from vision and language understanding. In: NIPS 2018, pp. 1031–1042 (2018)
33. Yu, L., Park, E., Berg, A.C., Berg, T.L.: Visual madlibs: fill in the blank image generation and question answering. arXiv preprint arXiv:1506.00278 (2015)