A Neuro-Symbolic ASP Pipeline for Visual Question Answering*

THOMAS EITER, NELSON HIGUERA, JOHANNES OETSCH and
MICHAEL PRITZ

Institute of Logic and Computation,
Vienna University of Technology (TU Wien), Austria
(*e-mails:* eiter@kr.tuwien.ac.at, higuera@kr.tuwien.ac.at, oetsch@kr.tuwien.ac.at,
pritz@kr.tuwien.ac.at)

Abstract

We present a neuro-symbolic visual question answering (VQA) pipeline for CLEVR, which is a well-known dataset that consists of pictures showing scenes with objects and questions related to them. Our pipeline covers (i) training neural networks for object classification and bounding-box prediction of the CLEVR scenes, (ii) statistical analysis on the distribution of prediction values of the neural networks to determine a threshold for high-confidence predictions, and (iii) a translation of CLEVR questions and network predictions that pass confidence thresholds into logic programmes so that we can compute the answers using an answer-set programming solver. By exploiting choice rules, we consider deterministic and non-deterministic scene encodings. Our experiments show that the non-deterministic scene encoding achieves good results even if the neural networks are trained rather poorly in comparison with the deterministic approach. This is important for building robust VQA systems if network predictions are less-than perfect. Furthermore, we show that restricting non-determinism to reasonable choices allows for more efficient implementations in comparison with related neuro-symbolic approaches without losing much accuracy.

KEYWORDS: answer-set programming, visual question answering, neuro-symbolic computation

1 Introduction

The goal in *visual question answering* (VQA) (Antol *et al.* 2015) is to find the answer to a question using information from a scene. A system must understand the question, extract the relevant information from the corresponding scene, and perform some kind of reasoning. *Neuro-symbolic approaches* are useful in this regard as they combine deep learning, which can be used for perception (e.g. object detection or natural language processing), with symbolic reasoning (Xu *et al.* 2018; Manhaeve *et al.* 2018; Yi *et al.* 2018; Yang *et al.* 2020; Basu *et al.* 2020; Mao *et al.* 2019). As the semantics of the employed reasoning formalism is known, the way in which an answer is reached is transparent.

We present a neuro-symbolic VQA pipeline for the CLEVR dataset (Johnson *et al.* 2017) that combines *deep neural networks* for perception and *answer-set programming*

(ASP) (Brewka *et al.* 2011) to implement the reasoning part. The system is publicly available at

https://github.com/Macehil/nesy-asp-vqa-pipeline

ASP offers a simple yet expressive modelling language and efficient solver technology. It is in particular attractive for this task as it allows to easily express non-determinism, preferences, and defaults. The scene encoding in the ASP programme makes use of *non-deterministic choice rules* for the objects predicted with high confidence by the network. This means that we do not only consider the prediction with the highest score, but also reasonable alternatives with lower ones. This allows our approach to make up for mistakes made in object classification in the reasoning component as the constraints in the programme exclude choices that do not lead to an answer.

For illustration, assume a scene with one red cylinder and the question "What shape is the red object?". Furthermore, assume that the neural network wrongly gives the cylinder a higher score for being blue than red. The ASP constraints enforce that an answer is derived. This entails that the right choice for the colour of the object must be red, and the correct answer "cylinder" is produced in the end.

While non-determinism improves the robustness of the VQA system, the downside is that, in case there are many object classes and the system is used with no restriction, it can negatively impact the reasoning performance in terms of run time. *The objective of this paper is to introduce a new method for restricting non-determinism that is sensitive to how well networks have been trained such that efficient reasoning is facilitated.*

While several datasets have been published to examine the strengths and weaknesses of VQA systems (Malinowski and Fritz 2014; Antol *et al.* 2015; Ren *et al.* 2015; Zhu *et al.* 2016; Johnson *et al.* 2017; Sampat *et al.* 2021), CLEVR is an ideal test bed for the purposes of this paper, since it is simple, well known, has detailed annotations describing the kind of reasoning each question requires, and focuses on basic object detection.

We in fact omit natural language processing for the VQA tasks because this would add a further variable and is not the focus of this work which is reasoning on top of object detection. Instead, we use *functional programmes* which are structured representations of the natural language questions already provided by CLEVR.

Our VQA pipeline for consists of the following stages:

(1) Training neural networks for object classification and bounding-box prediction of the CLEVR scenes using the object detection framework YOLOv3 (Redmon and Farhadi 2018).
(2) Statistical analysis on the distribution of prediction values of the neural networks to determine a threshold for high-confidence predictions defined as a function of mean and standard deviation of the distribution.
(3) Translating CLEVR questions and network predictions that pass confidence thresholds into logic programmes so that we can compute the answers using an ASP solver.

To determine what we regard as predictions of high confidence, we use statistical analysis on the distribution of network prediction values and disregard predictions that are below a threshold in Stage (2).

Our non-deterministic scene encoding approach is inspired by the neuro-symbolic systems NeurASP (Yang *et al.* 2020) and DeepProbLog (Manhaeve *et al.* 2018). DeepProbLog uses ProbLog, a probabilistic extension of Prolog, for reasoning. There, the output of the neural network is passed to the logic programme using neural-annotated disjunctions. NeurASP uses the same bridging idea, now called neural atoms, but uses ASP instead of ProbLog. Similar to our approach, neural atoms are translated into choice rules. NeurASP and DeepProbLog take multiple network predictions into account, in fact they are designed to consider all network predictions which are treated as a probability distribution. This can, as our experiments confirm, become a performance bottleneck if there are many object classes. Both systems feature *closed-loop reasoning*, that is, the outcome of the reasoning system can be back-propagated into the neural networks to facilitate better learning. Our pipeline is however uni-directional, as our goal is to explore the interplay between non-determinism and confidence thresholds regarding efficiency and robustness of the reasoning component. We leave the learning component for future work as a scalable implementation is not trivial in our setting and thus outside the scope of this paper.

We compare NeurASP and the reasoning component of DeepProbLog with our approach on the CLEVR data. Indeed, limiting non-determinism of neural network outputs in ASP programmes to reasonable choices leads to a drastic performance improvement in terms of run time with only little loss in accuracy and is thus important for efficient reasoning. Furthermore, our experiments show that our system performs well even if the neural networks are trained rather poorly and predictions by the network are less-than perfect. This is important for robust reasoning as even well-trained networks can be negatively affected by noise or if settings like illumination change.

The remainder of this paper is organised as follows. We first review ASP and CLEVR in Section 2. Our VQA pipeline using ASP and confidence thresholds is detailed in Section 3. Afterwards, we present an experimental evaluation of our approach in Section 4, discuss further relevant related work in Section 5, and conclude in Section 6.

2 Background

We next provide preliminaries on ASP and background on the CLEVR dataset.

2.1 Answer-set programming

ASP is a declarative problem-solving paradigm, where a problem is encoded as a logic programme such that its *answer sets* (which are special models) correspond to the solutions of the problem and are computable using ASP solvers, for example, from `potassco.org` or `www.dlvsystem.com`. We just briefly recall important ASP concepts; and refer for more details to the literature (Brewka *et al.* 2011; Gebser *et al.* 2012).

An *ASP programme* is a finite set of *rules* r of the form

$$a_1 \,|\, \ldots \,|\, a_k \ :\!\!- \ b_1, \ldots, \ b_m, \ not \ c_1, \ldots, \ not \ c_n, \quad k, m, n \geq 0, \tag{1}$$

where all a_i, b_j, c_l are first-order atoms and *not* is *default negation*; we denote by $H(r) = \{a_1, \ldots, a_k\}$ and $B(r) = B^+(r) \cup \{not \ c_j \mid c_j \in B^-(r)\}$ the head and body of r, respectively, where $B^+(r) = \{b_1, \ldots, b_m\}$ and $B^-(r) = \{c_1, \ldots, c_n\}$. Intuitively, r

says that if all atoms in $B^+(r)$ are true and there is no evidence that some atom in $B^-(r)$ is true, then some atom in $H(r)$ must be true. If $m = n = 0$ and $k = 1$, then r is a *fact* (with :– omitted); if $k = 0$, r is a *constraint*.

An *interpretation* I is a set of ground (i.e. variable-free) atoms. It satisfies a ground rule r if $H(r) \cap I \neq \emptyset$ whenever $B^+(r) \subseteq I$ and $I \cap B^-(r) = \emptyset$; I is a model of a ground programme P if I satisfies each $r \in P$, and I is an *answer-set* of P if in addition no $J \subset I$ is a model of the Gelfond–Lifschitz reduct of P w.r.t. I (Gelfond and Lifschitz 1991).

Models and answer sets of a programme P with variables are defined in terms of the grounding of P (replace each rule by its possible instances over the Herbrand universe).

We will also use *choice rules* and *weak constraints*, which are of the respective forms

$$i\,\{a_1; \ldots ; a_n\}\,j :- b_1, \ldots, b_m, \ not\ c_1, \ldots, \ not\ c_n, \tag{2}$$

$$:\sim b_1, \ldots, b_m, \ not\ c_1, \ldots, \ not\ c_n.\ [w, t]. \tag{3}$$

Informally, (2) says that when the body is satisfied, at least i and at most j atoms from $\{a_1, \ldots, a_n\}$ must be true in an answer set I, while (3) contributes tuple t with costs w, which is an integer number, to a cost function, when the body is satisfied in I, rather than to eliminate I; the answer set I is optimal, if the total cost of all such tuples is minimal.

2.2 The CLEVR dataset

CLEVR (Johnson *et al.* 2017) is a dataset designed to test and diagnose the reasoning capabilities of VQA systems.[1] It consists of pictures showing scenes with objects and questions related to them; there are about ten questions per image. The dataset was created with the goal of minimising biases in the data, since some VQA systems are suspected to exploit them to find answers instead of actually reasoning about the question and scene information (Johnson *et al.* 2017).

Each CLEVR image depicts a scene with objects in it. The objects differ by the values of their attributes, which are *size* (big, small), *colour* (brown, blue, cyan, grey, green, purple, red, yellow), *material* (metal, rubber), and *shape* (cube, cylinder, sphere). Every image comes with a ground truth scene graph describing the scene depicted in it. Figure 1 contains three images from the CLEVR validation dataset with corresponding questions.

In CLEVR, questions are constructed using *functional programmes* that represent the questions in a structured format. These programmes are symbolic templates for a question which are instantiated with the corresponding values. For each such question template, there are one or more natural language sentences to which they are mapped. For illustration, the question "How many large things are either cyan metallic cylinders or yellow blocks?" from Figure 1 can be represented by the functional programme shown in Figure 2. There, function *scene*() returns the set of objects of the scene, the *filter_** functions restrict a set of objects to subsets with respective properties, *union*() yields the union of two sets, and *count*() finally returns the number of elements of a set. A

---

[1] https://cs.stanford.edu/people/jcjohns/clevr/.

Q: Is there a big brown object of the same shape as the green thing?
A: Yes

Q: How many large things are either cyan metallic cylinders or yellow blocks?
A: 0

Q: The tiny shiny cylinder has what color?
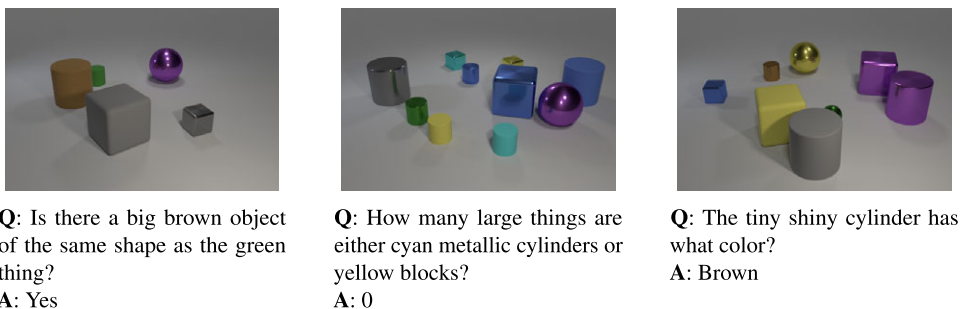**A**: Brown

Fig. 1. Three scenes and question-answer pairs from the CLEVR validation set.
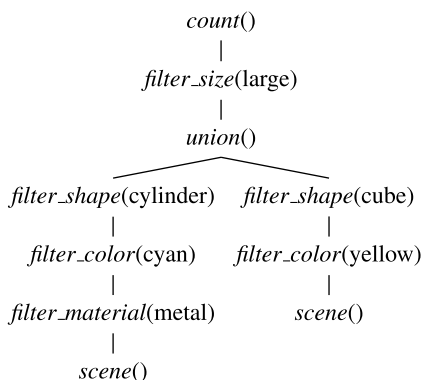

Fig. 2. CLEVR functional programme representing the question: "How many large things are either cyan metallic cylinders or yellow blocks?".
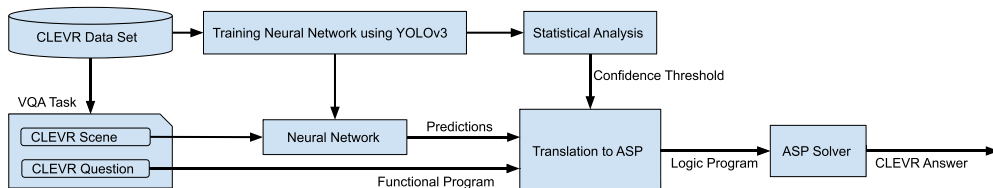

Fig. 3. Overview of our neuro-symbolic VQA pipeline.

detailed description of functional programmes in CLEVR can be found in the dataset documentation (Johnson *et al.* 2017).

3 The VQA pipeline

The architecture of our neuro-symbolic VQA pipeline that builds on object detection and ASP solving is depicted in Figure 3. A particular VQA task, which consists of a CLEVR scene and question, is translated to an ASP programme given predictions from a neural network for object detection, a functional programme, and a confidence threshold; by running an ASP solver, the answer to the CLEVR task is then figured out.

Before going into details of our VQA pipeline, we recapitulate the necessary stages of establishing it for the CLEVR dataset:

1. **Object detection**: we train neural networks for bounding-box prediction and object classification of the CLEVR scenes;
2. **Confidence thresholds**: we determine a threshold for network predictions that we consider to be of high confidence by statistical analysis on the distribution of prediction values of the neural networks;
3. **ASP encoding**: we translate CLEVR functional programmes that represent questions as well as network predictions that pass confidence thresholds into ASP programmes and use an ASP solver to compute the answers.

While the VQA tasks are designed to always have a unique answer, the ASP solver may give multiple results that correspond to alternative interpretations of the scene through the object detection network.

3.1 Object detection

We use YOLOv3 (Redmon and Farhadi 2018) for bounding-box prediction and object classification, adopting that the object detector's output is a matrix whose rows correspond to the bounding-box predictions in the input picture. Each bounding-box prediction is a vector of the form $(c_1, \ldots, c_n, x_1, y_1, x_2, y_2)$, where the pairs (x_1, y_1) and (x_2, y_2) give the top-left and bottom-right corner point of the bounding box, respectively, and $c_1, \ldots, c_n$ are *class confidence scores* with $c_i \in [0, 1]$ for $1 \leq i \leq n$; as customary, higher confidence scores represent higher confidence of a correct prediction. Each c_i represents the score for a specific combination of object attributes *size, colour, material* and *shape* and their respective values; we call this combination the *object class* of position i. For any object class c, let $\bar{c}$ be the list *size, shape, material, colour* of its attribute values. For example, assume c is the object class "large red metallic cylinder", then $\bar{c} = \text{large, cylinder, metallic, red}$. In total, there are $n = 96$ object classes in CLEVR.

Every row of the prediction matrix has also its own bounding-box confidence score. The number of bounding-box predictions of the object detection system depends on the *bounding-box threshold*, which is a hyper-parameter used to filter out rows with a low confidence score. For example, setting this threshold to 0.5 discards all predictions with confidence score below 0.5.

3.2 Confidence thresholds

Given class confidence scores $c_1, \ldots, c_n$ from an object detection prediction, we would like to focus on classifications that have reasonable high confidence and discard others with low confidence for the subsequent reasoning process. Using a fixed threshold hardly achieves this, since it does not take the distribution of confidence scores in the application area (or validation data for experiments) into account. Our approach solves this problem by fixing the threshold based on the mean and the standard deviation of prediction scores.

More formally, given a list of prediction matrices $\boldsymbol{X}^1, \ldots, \boldsymbol{X}^m$, where any $\boldsymbol{X^i}$ is of dimension $N^i \times M$, N^i is the number of bounding-box predictions in the input image i, each of which is described by M features. We compute the mean μ and standard deviation σ for the maximum class confidence scores:

$$\mu = \frac{1}{\sum_{k=1}^{m} N^k} \sum_{k=1}^{m} \sum_{i=1}^{N^k} \max_{1 \le j \le n} (\boldsymbol{X}_{i,j}^k), \qquad (4)$$

$$\sigma = \sqrt{\frac{1}{\sum_{k=1}^{m} N^k} \sum_{k=1}^{m} \sum_{i=1}^{N^k} (\max_{1 \le j \le n} (\boldsymbol{X}_{i,j}^k) - \mu)^2}. \qquad (5)$$

We suggest computing these values on the validation dataset used in training the object detector. Then, we define the *confidence threshold* θ that determines what is considered a confident class prediction as follows:

$$\theta = max(\mu - \alpha \cdot \sigma, 0). \qquad (6)$$

We consider class predictions as sufficiently confident if their confidence score is not lower than the mean minus α many standard deviations. The value for α in Eqn. (6) is a parameter we call the *confidence rate*. It must be provided and should depend on how well the network is trained: for a fixed α, the number of class predictions that pass the threshold decreases if the network gets better trained as the standard deviation becomes smaller. For a fixed network, the number of class predictions that pass the threshold decreases if α decreases and increases otherwise.

3.3 ASP encoding

To solve VQA tasks, we rely on ASP to infer the right answer given the neural network output and a confidence threshold. We outline the details in the following.

3.3.1 Question encoding

The first step of our approach is to translate the functional programme representing a natural language question into an ASP *fact representation*. We illustrate this for the question "How many large things are either cyan metallic cylinders or yellow blocks?" in Section 2.2. The respective functional programme in Fig. 2 is encoded by the following ASP facts:

$$end(8).$$
$$count(8, 7).$$
$$filter\_large(7, 6).$$
$$union(6, 3, 5).$$
$$filter\_cylinder(3, 2). \qquad\qquad filter\_cube(5, 4).$$
$$filter\_cyan(2, 1). \qquad\qquad filter\_yellow(4, 0).$$
$$filter\_metal(1, 0).$$
$$scene(0).$$

The structure of the functional programme is encoded using indices that refer to output (first arguments) and input (remaining arguments) of the respective basic functions.

Let $\boldsymbol{X}$ be a prediction matrix and θ be a confidence threshold as described in Section 3.2. Recall that the output of the basic CLEVR function *scene()* corresponds to the objects detected in the scene which in turn correspond to the individual rows of $\boldsymbol{X}$.

For a row $\boldsymbol{X}_i$ with confidence class scores $c_1, ..., c_n$, set C_i contains every object class c_j with score greater or equal than θ. If no such c_j exists, then C_i contains the k classes with highest class confidence scores, where $k \in \{1, \ldots, 96\}$, is a fixed integer; intuitively, k is a fall-back parameter to ensure that some classes are selected if all scores are low.

For every row $\boldsymbol{X}_i$ with bounding-box corners (x_1, y_1) and (x_2, y_2), as well as $C_i = \{c_1, \ldots c_l\}$, we construct a choice rule of form

$$\{obj(O, i, \bar{c}_1, x_1, y_1, x_2, y_2); \ldots; obj(O, i, \bar{c}_l, x_1, y_1, x_2, y_2)\} = 1 :\!- scene(O).$$

Every object with sufficiently high confidence score will thus be considered for computing the final answer in a non-deterministic way. For every $c \in C_i$, we add the weak constraint

$$:\!\sim obj(O, i, \bar{c}, x_1, y_1, x_2, y_2) \; [w_c, i],$$

where the weight w_c is defined as $\min(-1000 \cdot \ln(s), 5000)$, and s is the class confidence score for c in $\boldsymbol{X}_i$. This approach, which comes from the NeurASP implementation, achieves that object selections are penalised by a weight which corresponds to the object's class confidence score. Resulting answer sets can thus be ordered according to the total confidence of the involved object predictions. We refer to this encoding as *non-deterministic scene encoding*, but we also consider the special case of a *deterministic scene encoding* where each C_i holds only the single object class with the highest confidence score.

3.3.3 Encoding of the basic CLEVR functions

We next present encodings for the remaining CLEVR functions.

Filter rules: The CLEVR filter rules restrict sets of objects. We only present the rule for *filter_color*(yellow); the rules for the other colours, materials, and shapes are analogous:

$$obj(T, I, \ldots, \text{yellow}, \ldots) :\!- filter\_yellow(T, T_1), obj(T_1, I, \ldots, \text{yellow}, \ldots).$$

The variables T and T_1 are used to indicate output resp. input references; I represents the object identifier. We omit arguments irrelevant for the particular filter functions.

Count rule: Function *count()* returns the number of elements of a given set. We encode it as follows:

$$int(T, V) :\!- count(T, T_1), \; \#count\{I : obj(T_1, I, ...)\} = V.$$

Here, $\#count$ is an ASP *aggregate function* that computes the numbers of object identifiers referenced by variable T_1.

Rules for set operations: The two set operation functions in CLEVR are intersection and union. We present respective ASP rules for each of them:

$$obj(T, I, \ldots) :\!-\ and(T, T_1, T_2),\ obj(T_1, I, \ldots),\ obj(T_2, I, \ldots)\,.$$
$$obj(T, I, \ldots) :\!-\ or(T, T_1, T_2),\ obj(T_1, I, \ldots)\,.$$
$$obj(T, I, \ldots) :\!-\ or(T, T_1, T_2),\ obj(T_2, I, \ldots)\,.$$

Uniqueness constraint: The CLEVR function *unique*() is used to assert that there is exactly one input object, which is then propagated to the output. We encode this in ASP using a constraint to eliminate answer sets violating uniqueness and a rule for propagation:

$$:\!-\ unique(T, T_1),\ obj(T_1, I, \ldots),\ obj(T_1, I', \ldots),\ I \neq I'\,.$$
$$obj(T, \ldots) :\!-\ unique(T, T_1),\ obj(T_1, \ldots)\,.$$

Spatial-relation rules: Several CLEVR functions allow to determine objects in a certain spatial relation with another object. We present the rule for identifying all objects that are left relative to a given reference; the rules for right, front, and behind, are analogous:

$$obj(T, I, \ldots) :\!-\ relate\_left(T, T_1, T_2), I \neq I', X_1 < X_1',$$
$$obj(T_1, I, \ldots, X_1, \ldots), obj(T_2, I', \ldots, X_1', \ldots)\,.$$

Exist rule: The *exist*() rule in CLEVR returns true if the referenced set of objects is not empty, and it returns false otherwise using default negation. Respective ASP rules look as follows:

$$bool(T, \text{true}) :\!-\ exist(T, T_1),\ obj(T_1, \ldots)\,.$$
$$bool(T, \text{false}) :\!-\ exist(T, T_1),\ not\ bool(T, \text{true})\,.$$

Query rules: Query functions allow to return an attribute value of a referenced object. We present a rule to query for the size of an object; the rules for colour, material, and shape look similar:

$$size(T, Size) :\!-\ query\_size(T, T_1), obj(T_1, \ldots, Size, \ldots)\,.$$

Same-attribute-relation rules: Similar to the spatial-relation functions, same-attribute-relation rules allow to select sets of objects if they agree on a specified attribute with a specified reference object. We illustrate the ASP encoding for the size attribute, the ones for colour, material and shape are defined with the necessary changes:

$$obj(T, I, \ldots) :\!-\ same\_size(T, T_1, T_2),\ obj(T_1, I, \ldots, Size, \ldots),$$
$$obj(T_2, I', \ldots, Size, \ldots), I \neq I'\,.$$

Integer-comparison rules: CLEVR supports the common relations for comparing integers like "equals", "less-than", and "greater-than". We present the ASP encoding for "equals":

$$bool(T, \text{true}) :\!-\ equal\_integer(T, T_1, T_2),\ int(T_1, V),\ int(T_2, V)\,.$$
$$bool(T, \text{false}) :\!-\ equal\_integer(T, T_1, T_2),\ not\ bool(T, \text{true})\,.$$

Attribute-comparison rules: To check whether two objects have the same attributes, like size, colour, material, or shape, CLEVR provides attribute-comparisons rules. The one for size can be represented in ASP as follows, the others are defined analogously:

$$bool(T, \text{true}) :\!- \; equal\_size(T, T_1, T_2), \; size(T_1, V), \; size(T_2, V) \,.$$
$$bool(T, \text{false}) :\!- \; equal\_size(T, T_1, T_2), \; not \; bool(V, \text{true}) \,.$$

In addition to the rules above, we also use rules to derive the $ans/1$ atom that extracts the final answer for the encoded CLEVR question from the output of the basic function at the root of the computation:

$$ans(V) :\!- \; end(T), size(T, V) \,. \qquad\qquad ans(V) :\!- \; end(T), shape(T, V) \,.$$
$$ans(V) :\!- \; end(T), color(T, V) \,. \qquad\qquad ans(V) :\!- \; end(T), bool(T, V) \,.$$
$$ans(V) :\!- \; end(T), material(T, V) \,. \qquad ans(V) :\!- \; end(T), int(T, V) \,.$$
$$:\!- \; not \; ans(\_) \,.$$

The last constraint enforces that at least one answer is derived.

Putting all together, to find an answer to a CLEVR question, we translate the corresponding functional programme into its fact representation and join it with the rules from above. Each answer set then corresponds to a CLEVR answer founded in a particular choice for scene objects. For the deterministic, resp., non-deterministic encoding, at most one, resp., multiple answer sets are possible; no answer set means imperfect object recognition. In case of multiple answer sets, we use answer-set optimisation over the weak constraints to determine the most plausible solution.

4 Experiments on the CLEVR dataset

Recall that the parameters of our approach are (i) the bounding-box threshold for object detection, (ii) the confidence rate α for computing the confidence threshold as distance from the mean in terms of standard deviations, and (iii) k as a fall-back parameter for object-class selection. We experimentally evaluated our approach on the CLEVR dataset to study the effects of different parameter settings. In particular, we study

- different bounding-box thresholds and training epochs for object detection,
- how the deterministic scene encoding compares to the non-deterministic one, and
- runtime performance of our approach in comparison with NeurASP and ProbLog.

For the non-deterministic scene encoding, we consider different settings for α and set $k = 1$.

We restricted our experiments to a sample of 15000 CLEVR questions as the systems NeurASP and ProbLog would exceed the memory limits on the unrestricted dataset.

All experiments were carried out on an Ubuntu (20.04.3 LTS) system with a 3.60GHz Intel CPU, 16GiB of RAM, and an NVIDIA GeForce GTX 1080 GPU with 8GB of memory installed.

4.1 Object-detection evaluation

For object detection, we used an open-source implementation of YOLOv3 (Redmon and Farhadi 2018).[2] The system was trained on 4000 CLEVR images with bounding-box

---

[2] https://github.com/eriklindernoren/PyTorch-YOLOv3.

Table 1. *Precision and recall for object detection on CLEVR scenes*

Epochs/Threshold	25/0.25	50/0.25	200/0.25	25/0.50	50/0.50	200/0.50
Recall	71.23%	96.39%	99.20%	44.41%	89.06%	98.03%
Precision	83.77%	98.27%	99.58%	97.17%	99.28%	99.82%

annotations, as suggested in related work (Yi *et al.* 2018). We used models trained for 25, 50, and 200 epochs, resp., to obtain different levels of training for the neural networks. For the bounding-box thresholds, we considered two settings, namely 0.25 and 0.50. Table 1 summarises the results of the evaluation of how the networks of different training quality perform for detecting the objects in the CLEVR scenes. We report on precision and recall, which are defined as usual in terms of true positives (TP), false positives (FP), and false negatives (FN). A *TP* (resp., *FP*) is a prediction that is correct (resp., incorrect) w.r.t. the scene annotations in CLEVR. An *FN* is an object that exists according to the scene annotations, but there is no corresponding prediction.

As expected, our results show that the total number of FP and FN decreases for the better trained YOLOv3 models. Naturally, a low bounding-box threshold yields more FP detections, while the number of FN decreases. Setting the bounding-box threshold to a higher value usually leads to fewer FP but also more FN.

4.2 Question-answering evaluation

We used the ASP solver `clingo` (v. 5.5.1) to compute answer sets.[3] Table 2 sheds light on the impact of the training level and bounding-box thresholds of the models on question answering for the deterministic and non-deterministic scene encodings. Our system yields either correct, incorrect, or no answers to the CLEVR questions, and we report respective rates. The non-deterministic scene encoding outperformed the deterministic approach for all settings of training epochs and bounding-box thresholds, and the rate of correct answers increases with larger α. The differences are considerable if the networks are trained rather poorly and and become small or even disappear for well-trained ones. It thus seems beneficial to consider more than one prediction of the object detection system if network predictions are less than perfect. Also, lower bounding-box thresholds lead to more correct results in all cases, especially for the deterministic encoding. Hence, being too selective in the object detection is counterproductive.

4.3 Comparison with NeurASP and ProbLog

The related systems NeurASP and DeepProbLog also embody the idea of non-determinism for object classifications, but they do not incorporate a mechanism to restrict object classes to ones with high confidence like in our approach; this can drag down performance considerably. We conducted different experiments to investigate the impact of limiting the number of object classes as in our approach on runtimes and accuracy of the questions answering task in comparison with the aforementioned related systems.

---

[3] https://potassco.org/clingo/.

Table 2. *Results for CLEVR question answering*

Epochs/Threshold		25/0.25	50/0.25	200/0.25	25/0.50	50/0.50	200/0.50
Deterministic Scene Encoding							
	Correct	64.39%	92.93%	97.01%	42.33%	82.79%	95.05%
	Wrong	17.60%	3.06%	1.11%	29.55%	8.65%	2.22%
	No answer	18.01%	4.01%	1.88%	28.12%	8.56%	2.73%
Non-Deterministic Scene Encoding							
$\alpha = 0.5$	Correct	74.12%	93.13%	96.94%	72.36%	92.54%	96.71%
	Wrong	12.29%	2.60%	1.01%	13.21%	2.98%	1.27%
	No answer	13.59%	4.27%	2.05%	14.43%	4.48%	2.02%
$\alpha = 1.0$	Correct	76.17%	93.13%	96.94%	74.29%	92.54%	96.71%
	Wrong	12.53%	2.60%	1.01%	13.47%	2.98%	1.27%
	No answer	11.30%	4.27%	2.05%	12.24%	4.48%	2.02%
$\alpha = 1.5$	Correct	80.61%	93.13%	96.94%	78.83%	92.55%	96.71%
	Wrong	13.02%	2.60%	1.01%	14.00%	2.98%	1.27%
	No answer	6.37%	4.27%	2.05%	7.17%	4.47%	2.02%
$\alpha = 2.0$	Correct	84.09%	93.17%	96.94%	82.65%	92.56%	96.71%
	Wrong	13.52%	2.60%	1.01%	14.53%	2.98%	1.27%
	No answer	2.39%	4.23%	2.05%	2.82%	4.46%	2.02%

The choice rules in NeurASP always contain the 96 CLEVR object classes, whose scores come from the YOLOv3 network. In addition, we also used a setup for NeurASP where only the highest prediction is considered while the probabilities of all other atoms are set to 0. While the former setting is more similar to our approach, the latter is the one used by Yang *et al.* (2020) in their object detection example. Table 3 summarises the results on question answering accuracy, and Table 4 shows the total runtime for the different systems under consideration on the 15,000 questions. NeurASP outperforms our approach in terms of correct answers as it does not restrict the number of atoms for the choice rules. However, this comes at a price as runtimes are much longer, which can be explained by the inflation of the search space due to the unrestricted choice rules. Also, the rate of incorrect answers is higher for NeurASP while our approach will more often remain agnostic when in doubt. For the non-deterministic encoding of our approach, we observe a similar jump in runtimes for $\alpha = 2.5$ as more object classes are included in the choice rules.

We could not use DeepProbLog for CLEVR directly as annotated disjunctions that depend on a variable number of objects in a scene are not supported and would require some extensions. Instead, we evaluated a translation to ProbLog, as DeepProblog's inference component is essentially the one of Problog (Manhaeve *et al.* 2018). Recall that for NeurASP and DeepProbLog, neural network outputs are interpreted as probability distributions. While NeurASP does not strictly require this and works also in our setting, ProbLog is less lenient and network outputs need to be normalised so that the sum of their scores does not exceed 1. This does however not change results as only the relative

Table 3. *Comparisons with NeurASP and ProbLog*

Epochs/Threshold	25/0.25	50/0.25	200/0.25	25/0.50	50/0.50	200/0.50
NeurASP						
Correct	86.09%	96.74%	98.53%	84.87%	96.17%	98.20%
Wrong	13.88%	3.21%	1.45%	15.09%	3.77%	1.77%
No answer	0.03%	0.05%	0.03%	0.04%	0.06%	0.03%
NeurASP (best prediction)						
Correct	75.63%	95.78%	98.33%	53.12%	88.21%	96.87%
Wrong	23.51%	4.14%	1.63%	38.17%	11.30%	3.06%
No answer	0.87%	0.08%	0.03%	8.71%	0.49%	0.07%
ProbLog						
Correct	83.84%	96.25%	98.33%	81.97%	95.27%	97.85%
Wrong	14.46%	2.66%	1.14%	15.45%	3.13%	1.34%
No answer	1.70%	1.09%	0.53%	2.57%	1.59%	0.81%

Table 4. *Comparison of total VQA runtimes (in seconds)*

Epochs/Threshold		25/0.25	50/0.25	200/0.25	25/0.50	50/0.50	200/0.50
NeurASP		9149	4375	4364	8750	4234	4694
NeurASP (best prediction)		3114	3628	3575	1245	3174	3577
ProbLog		6235	6894	6463	5311	5883	6138
Deterministic Scene Encoding		84	89	87	81	89	85
Non-Det. Scene Encoding with	$\alpha = 0.5$	112	123	130	109	124	131
	$\alpha = 1.5$	141	125	126	140	125	133
	$\alpha = 2.0$	154	126	130	165	125	135
	$\alpha = 2.5$	3656	127	128	3541	123	128

order of the object-class scores is relevant for determining the most plausible answer. For the case of the unrestricted 96 object classes, computing results on our hardware was infeasible. We thus considered only the three object classes with highest confidence scores for every bounding-box prediction in our experiments. Results on runtimes and accuracy are therefore lower bounds for the unrestricted case. The picture for ProbLog is quite similar to that of NeurASP: While additional predictions help for question answering to some extent, this is at the cost of a considerable increase in runtime.

Overall, the experiments further support our belief that non-determinism is useful for neuro-symbolic VQA systems and suitable mechanisms to restrict it do reasonable choices allows for more efficient implementations.

5 Further related work

Purely deep-learning-based approaches (Yang *et al.* 2016; Lu *et al.* 2016; Jabri *et al.* 2016) led to significant advances in VQA. Some systems rely on attention mechanisms to focus on certain features of the image and question to retrieve the answer (Yang *et al.* 2016;

Lu *et al.* 2016). Jabri *et al.* (2016)) achieved good results by framing a VQA task as a classification problem. Some VQA systems are however suspected to not learn to reason but to exploit dataset biases to find answers, as described by Johnson *et al.* (2017).

Besides these purely data driven attempts, there are also systems which incorporate symbolic reasoning in combination with neural-based methods for VQA (Yi *et al.* 2018; Basu *et al.* 2020; Mao *et al.* 2019). The system proposed by Yi *et al.* (2018) consists of a scene parser, which retrieves object level scene information from images, a question parser, which creates symbolic programmes from natural language questions, and a programme executor that runs these programmes on the abstract scene representation. Our system is akin to this system, but we use ASP for scene representation as well as question encoding, and our programme executor is an ASP solver. A similar system architecture appears in the approach by Mao *et al.* (2019) with the difference that scene and question parsing is jointly learned from image and question-answer pairs, whereas the components of Yi *et al.*'s system are trained separately. This means that annotated images are not necessary for training, which makes the system more versatile. The approach of Basu *et al.* (2020) builds like ours on ASP. They use object-level scene representations and parse natural language questions to obtain rules which represent the question. The answer to a question is given by the answer set for the image-question encoding which is combined with commonsense knowledge. However, their approach is not amenable to non-determinism for the scene encoding in order to deal with competing object classifications as we do.

Riley and Sridharan (2019) present an integrated approach for deep learning and ASP for representing incomplete commonsense knowledge and non-monotonic reasoning that also involves learning and programme induction. They apply their approach to VQA tasks with explanatory questions and are able to achieve better accuracy on small data sets than end-to-end architectures. As in our work, they use neural networks to extract features of an image. We focus however more narrowly on the interface between the neural network outputs and the logical rules and turn network outputs into choice rules to further improve robustness, which has not been considered by Riley and Sridharan.

6 Conclusion

We have introduced a neuro-symbolic VQA pipeline for the CLEVR dataset based on a translation from CLEVR questions in the form of functional programmes to ASP rules, where we proposed a non-deterministic and a deterministic approach to encode object-level scene information. Notably, non-determinism is restricted to network predictions that pass a confidence threshold determined by statistical analysis. It takes the variance of prediction quality into account and can be adjusted by the novel confidence rate parameter α, which supports control of non-determinism, resp. disjunctive information.

Our experiments confirm that, on the one hand, non-determinism is important for robustness of the reasoning component especially if the neural networks for object classification are not well-trained or predictions are negatively affected by other causes. On the other hand, unrestricted non-determinism as featured by related neuro-symbolic systems can pose a performance bottleneck. Our method of using a confidence threshold is a viable compromise between quality of question answering and efficiency of the reasoning component. The insight that it makes sense to deal with uncertainty at the level

of the reasoning component is in fact not restricted to ASP and therefore also provides directions to further improve related approaches.

While CLEVR is well-suited for the purposes of this work, the scenes are quite simple and do not require advanced features of ASP like expressing common-sense knowledge. For future work, we intend to apply our approach also to other datasets, especially ones that do not use synthetic scenes and where object classification might be harder, resulting in increased uncertainty. There, using additional domain knowledge and representing it with ASP could come in handy. Furthermore, we plan to extend our pipeline to closed-loop reasoning, that is, using the output of the ASP solver also in the learning process.

References

ANTOL, S., AGRAWAL, A., LU, J., MITCHELL, M., BATRA, D., ZITNICK, C. L. AND PARIKH, D. VQA: Visual question answering. In *Proceedings of the 2015 IEEE International Conference on Computer Vision* (*ICCV*) 2015, pp. 2425–2433. IEEE.

BASU, K., SHAKERIN, F. AND GUPTA, G. AQuA: ASP-based visual question answering. In *Proceedings of the 22nd International Symposium on Practical Aspects of Declarative Languages* (*PADL 2020*) 2020, vol. 12007. Lecture Notes in Computer Science. Springer, 57–72.

BREWKA, G., EITER, T. AND TRUSZCZYŃSKI, M. 2011. Answer set programming at a glance. *Communications of the ACM*, *54*, 12, 92–103.

GEBSER, M., KAMINSKI, R., KAUFMANN, B. AND SCHAUB, T. 2012. *Answer Set Solving in Practice*, volume 6 of *Synthesis Lectures on Artificial Intelligence and Machine Learning*. Morgan & Claypool Publishers.

GELFOND, M. AND LIFSCHITZ, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, *9*, 3–4, 365–385.

JABRI, A., JOULIN, A. AND VAN DER MAATEN, L. Revisiting visual question answering baselines. In *Proceedings of the 14th European Conference on Computer Vision* (*ECCV 2016*) 2016, vol. 9912. Lecture Notes in Computer Science. Springer, 727–739.

JOHNSON, J., HARIHARAN, B., VAN DER MAATEN, L., FEI-FEI, L., ZITNICK, C. L. AND GIRSHICK, R. B. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) 2017. IEEE, 1988–1997.

LU, J., YANG, J., BATRA, D. AND PARIKH, D. Hierarchical question-image co-attention for visual question answering. In *Advances in Neural Information Processing Systems* (*NIPS 2016*) 2016, vol. 29. Curran Associates, Inc., 289–297.

MALINOWSKI, M. AND FRITZ, M. A multi-world approach to question answering about real-world scenes based on uncertain input. In *Advances in Neural Information Processing Systems* (*NIPS 2014*) 2014, vol. 27. Curran Associates, Inc., 1682–1690.

MANHAEVE, R., DUMANCIC, S., KIMMIG, A., DEMEESTER, T. AND RAEDT, L. D. DeepProbLog: Neural probabilistic logic programming. In *Advances in Neural Information Processing Systems* (*NeurIPS 2018*) 2018, vol. 31, 3753–3763.

MAO, J., GAN, C., KOHLI, P., TENENBAUM, J. B. AND WU, J. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. In *Proceedings of the 7th International Conference on Learning Representations* (*ICLR 2019*) 2019.

REDMON, J. AND FARHADI, A. 2018. YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767, abs/1804.02767.*

REN, M., KIROS, R. AND ZEMEL, R. Exploring models and data for image question answering. In *Advances in Neural Information Processing Systems* (*NIPS 2015*) 2015, vol. 28. Curran Associates, Inc., 2953–2961.

RILEY, H. AND SRIDHARAN, M. 2019. Integrating non-monotonic logical reasoning and inductive learning with deep learning for explainable visual question answering. *Frontiers in Robotics and AI, 6:125.*

SAMPAT, S. K., KUMAR, A., YANG, Y. AND BARAL, C. CLEVR_HYP: A challenge dataset and baselines for visual question answering with hypothetical actions over images. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies* (*NAACL-HLT 2021*) 2021. Association for Computational Linguistics, 3692–3709.

XU, J., ZHANG, Z., FRIEDMAN, T., LIANG, Y. AND VAN DEN BROECK, G. A semantic loss function for deep learning with symbolic knowledge. In *Proceedings of the 35th International Conference on Machine Learning* (*ICML 2018*) 2018, vol. 80. Proceedings of Machine Learning Research. PMLR, 5502–5511.

YANG, Z., HE, X., GAO, J., DENG, L. AND SMOLA, A. Stacked attention networks for image question answering. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) 2016, 21–29.

YANG, Z., ISHAY, A. AND LEE, J. NeurASP: Embracing neural networks into answer set programming. In *Proceedings of the 29th International Joint Conference on Artificial Intelligence* (*IJCAI 2020*) 2020. International Joint Conferences on Artificial Intelligence Organization, 1755–1762.

YI, K., WU, J., GAN, C., TORRALBA, A., KOHLI, P. AND TENENBAUM, J. Neural-symbolic VQA: Disentangling reasoning from vision and language understanding. In *Advances in Neural Information Processing Systems* (*NeurIPS 2018*) 2018, vol. 39. Curran Associates, Inc., 1039–1050.

ZHU, Y., GROTH, O., BERNSTEIN, M. AND FEI-FEI, L. Visual7w: Grounded question answering in images. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition* (*CVPR*) 2016, 4995–5004.