# Declarative Knowledge Distillation from Large Language Models for Visual Question Answering Datasets

**Thomas Eiter**[1] , **Jan Hadl**[1] , **Nelson Higuera**[1] , **Johannes Oetsch**[2]

[1]Institute for Logic and Computation, TU Wien, Favoritenstraße 9–11, 1040 Vienna, Austria
[2]Department of Computing, Jönköping University, Gjuterigatan 5, 551 11 Jönköping, Sweden

{thomas.eiter, jan.hadl, nelson.ruiz}@tuwien.ac.at, johannes.oetsch@ju.se

## Abstract

Visual Question Answering (VQA) is the task of answering a question about an image and requires processing multimodal input and reasoning to obtain the answer. Modular solutions that use declarative representations within the reasoning component have a clear advantage over end-to-end trained systems regarding interpretability. The downside is that crafting the rules for such a component can be an additional burden on the developer. We address this challenge by presenting an approach for declarative knowledge distillation from Large Language Models (LLMs). Our method is to prompt an LLM to extend an initial theory on VQA reasoning, given as an answer-set program, to meet the requirements of the VQA task. Examples from the VQA dataset are used to guide the LLM, validate the results, and mend rules if they are not correct by using feedback from the ASP solver. We demonstrate that our approach works on the prominent CLEVR and GQA datasets. Our results confirm that distilling knowledge from LLMs is in fact a promising direction besides data-driven rule learning approaches.

## 1 Introduction

*Visual question answering* (VQA) (Antol et al. 2015; Goyal et al. 2017) is a challenging problem with valuable applications (Barra et al. 2021; Lin et al. 2023); it is the task of providing an accurate answer for a question about a visual scene. This requires not just a joint understanding of vision and text, but also the ability to follow complex chains of reasoning operations.

*Neurosymbolic approaches to VQA* (Mao et al. 2019; Yi et al. 2018; Amizadeh et al. 2020; Eiter et al. 2022; Surís, Menon, and Vondrick 2023; Johnston, Nogueira, and Swingler 2023, etc.) use deep learning for perception, produce a symbolic representation of the input image and question, and then perform reasoning on this representation in a purely symbolic way. These approaches are interpretable, transparent, and can be extended easily due to their compositional structure. A promising direction in this regard is to use logic-based formalisms for the reasoning component. We are in particular interested in using Answer-Set Programming (ASP) (Brewka, Eiter, and Truszczynski 2011; Lifschitz 2019), a prominent knowledge representation framework, for the reasoning module of such systems. Besides concise representations, advantages are that non-determinism
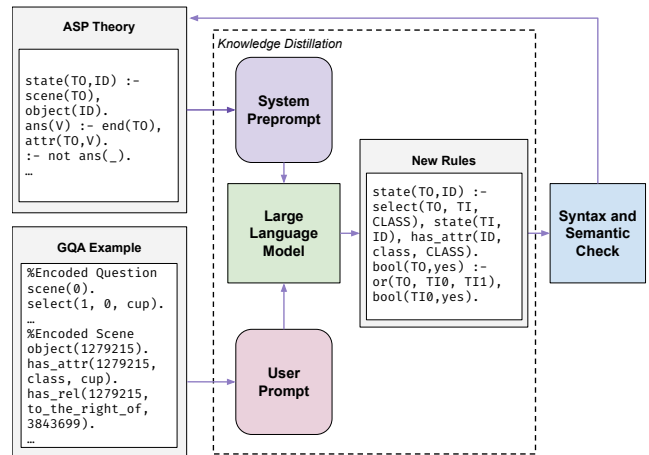


Figure 1: An overview of our knowledge distillation method.

allows for multiple answers if desired, ambiguity in the perception often can be resolved in the reasoning module (Eiter et al. 2022), and one can more easily add explanation capabilities (Eiter et al. 2023). Using ASP to augment VQA with reasoning capabilities is a topic that is indeed currently gaining traction and is used not only for VQA (Abraham, Alirezaie, and Raedt 2024; Eiter et al. 2023; Eiter et al. 2022; Basu, Shakerin, and Gupta 2020), but also for tasks such as segmentation of medical images (Bruno et al. 2021). The downside is that crafting the rules for such a component is not always easy and can be an additional burden on the developer.

We address this challenge by presenting an approach for *declarative knowledge distillation* from Large Language Models (LLMs) (Vaswani et al. 2017; Zhao et al. 2023) (see Fig. 1). The premise of this work is that we want to develop a modular solution for a VQA task for which we have, as is commonly the case, a dataset of questions, scenes, and answers at our disposal. Further, the reasoning component involves already an initial ASP theory that covers some but not all aspects of the VQA task. Our method is to prompt an LLM to extend this initial theory to meet the remaining requirements. Examples from a VQA dataset are used to guide the LLM, validate the results, and mend the generated rules if they are not correct using feedback from the ASP solver. More specifically, if we encounter an example from

the dataset for which the system does not yet give the correct answer, we prompt the LLM with a description of the scene, question, and expected answer to suggest rules to extend the current program. We then run an ASP solver to test the new program. If the output of the ASP solver does not match the expected answer, we hand this ouput and the expected answer to the LLM and prompt it to correct the ASP program accordingly. If the ASP solver gives a syntax error, we ask the LLM to fix it. If the test passes, we proceed with regression testing on previous examples to ensure that the system will still answer any question it could answer before the extension. We also experiment with presenting examples in batches to reduce the number of calls to the LLM.

Our approach bears similarities with methods from statistical-relational learning (SRL) (Raedt and Kersting 2017), as there the goal is also to find rules that generalise examples from a dataset. While SRL is data-driven, our method uses examples only to guide a knowledge distillation process via conditioning an LLM; we go into more detail in the related work section.

We use two VQA datasets to evaluate our knowledge distillation method: CLEVR (Johnson et al. 2017) and GQA (Hudson and Manning 2019). CLEVR uses synthetic scenes and is designed to challenge VQA systems with compositional questions that require breaking down the overall task into a tree of primitive operations that is evaluated recursively. On the other hand, GQA uses real images that depict complex visual scenes and diverse questions with a large number of possible answers that involve reasoning on the image scene graph, i.e., the objects in the scene, their attributes, and relations. To solve CLEVR, we use a recent VQA system (Eiter et al. 2023). For GQA, we use a similar system (Hadl 2023), which is able to perform VQA in a *zero-shot manner*, i.e., without training or fine-tuning of components to the current dataset. This is a significant advantage over related systems that require a costly training process. Both VQA systems use ASP for reasoning.

Our experiments show that LLMs can understand and produce ASP rules to implement new reasoning operators, and thus can effectively assist in the process of generating the reasoning component. This methodology offers a promising avenue for extracting domain-specific knowledge from LLMs in the realm of VQA and offers a viable alternative to data-driven rule learning approaches.

The remainder of the paper is organised as follows. We discuss related work in Section 2 and present the VQA approaches used in Section 3. Then, we introduce the knowledge distillation method in Section 4 and evaluate it in Section 5. Finally, Section 6 presents concluding remarks and pointers to future work.

## 2 Related Work

*Answer-Set Programming and LLMs.* Recently, LLMs have been explored in the context of ASP. Ishay, Yang, and Lee (2023) observed that LLM reasoning capabilities are shallow, but they can serve as a highly effective semantic parser to transform input into ASP representations. These are then used to solve logical puzzles. We also recently proposed to use LLMs to parse the question into ASP facts

in the context of VQA for images of graphs (Bauer et al. 2023). Rajasekharan et al. (2023) showed that the combination of LLMs with ASP in their STAR framework produces good results for natural language tasks that require qualitative reasoning, mathematical reasoning, and goal-directed conversation. Our work is different as we do not focus on semantic parsing but on the more challenging task of knowledge distillation, where we aim for a system that can produce sound logical rules capturing knowledge about a particular domain. Recent work by Zhu et al. (2023) explores the use of LLMs to learn rules from arithmetic and kinship relationships, yet the rules they learn do not contain variables and their semantics is informal.

*Statistical-relational learning.* Similar to our approach, methods from statistical-relational learning (SRL) (Raedt and Kersting 2017), in particular from inductive logic programming (ILP) (Muggleton 1991; Muggleton and Raedt 1994; Cropper et al. 2022), aim at producing rules from example data and a background theory. SRL has seen great advances in terms of scalability by, e.g., applying gradient-based boosting (Gutmann and Kersting 2006), and systems like ILASP (Law, Russo, and Broda 2020) and FastLAS (Law et al. 2020) provide means for inductively learning expressive ASP programs.

However, SRL takes a statistical and probabilistic learning perspective that is in essence data driven. Our method is orthogonal to that, as it does not aim at learning. On the surface, we also use a data set, but the role is very different as it guides a knowledge distillation process via conditioning an LLM. ILP uses a search-based approach where the solutions produced are correct and minimal under some criteria. A key aspect of many ILP systems are *mode declarations* that define the syntactic form of allowed rules to restrict the search space of possible programs. Mode declarations are in a formal language, and they tacitly assume an intuition about the form of the solution. For the distillation approach , we do not need that; we only elicit knowledge that is already present in the LLM, and the information in the prompt that instructs what rules we want is informal and in natural language. When prompting LLMs, rules are general by command—while optimality is not enforced, it may happen implicitly.

*Modular neurosymbolic VQA.* There are several VQA systems that feature a modular architecture which combines subsymbolic with symbolic components (Yi et al. 2018; Mao et al. 2019; Amizadeh et al. 2020; Eiter et al. 2022; Surís, Menon, and Vondrick 2023; Johnston, Nogueira, and Swingler 2023). Specifically, Yi et al. (2018) used a pipeline to extract a *scene graph* (a list of all objects detected in the image with their attributes and positions) from the image. They then translated the provided question into a structured representation of the reasoning steps, called *functional program*, and executed this program on the structural scene representation to obtain an answer. The authors showed excellent results on the popular CLEVR dataset (Johnson et al. 2017). This approach has been advanced with logic-based reasoning processes, e.g., by Differentiable First-Order Logic ($\nabla$-FOL) (Amizadeh et al. 2020), or by ASP (Eiter et al. 2022). These reasoning processes can consider not just the

most probable scene-graph prediction, but rather the entire vector of probabilities as output by the object detection and attribute/relation classifier networks that form the visual perception component of the VQA pipeline. Foundational models such as Vision-Language Models (VLMs) such as BLIP-2 (Li et al. 2023) and SimVLM (Wang et al. 2022) have become sufficiently strong through their pre-training regimes to generalise well to multiple different datasets. Approaches that use these VLMs as components are, e.g., ViperGPT (Surís, Menon, and Vondrick 2023), CodeVQA (Subramanian et al. 2023), and PnP-VQA (Tiong et al. 2022).

# 3 Background & VQA Methodology

In this section, we review the basics of the logic-based VQA approaches for the two datasets, GQA and CLEVR, that we are going to use for our evaluation. Both systems use ASP to derive answers from a symbolic scene representation. So we start by reviewing the basics of ASP next.

## 3.1 Answer-Set Programming

Answer-Set Programming (ASP) (Brewka, Eiter, and Truszczynski 2011; Lifschitz 2019) is a well-known approach to declarative problem solving, in which solutions to a problem are described by sets of logical rules. Efficient ASP solvers for evaluating the rules are readily available.[1]

For our concerns, an ASP program is a finite set $P$ of rules $r$ of the following form:

$$a :- \ b_1, \ldots, \ b_n, \ not \ c_1, \ldots, \ not \ c_n \quad m, n \geq 0$$

where $a$, all $b_i$, and all $c_j$ are atoms in a first-order predicate language, and $not$ stands for negation as failure (aka. weak negation). We allow that $a$ may be missing (viewed as falsity); then $r$ acts as a constraint. Intuitively, the rule means that whenever all $b_i$ are true and none of the $c_j$ can be shown to be true, then $a$ must be true. Some rules appear in Fig 2b.

The semantics of a ground (variable-free) ASP program is given in terms of answer sets, which are Herbrand models that satisfy a stability condition (Gelfond and Lifschitz 1988). A Herbrand interpretation of $P$ is a set $I$ of ground atoms in the language induced by $P$ (intuitively, the atoms that are true). Such an $I$ is a model of $P$ if for each rule $r$ in $P$ either $(i)$ $a \in I$ or $(ii)$ $\{b_1, \ldots, b_n\} \not\subseteq I$ or $(iii)$ $I \cap \{c_1, \ldots, c_n\} \neq \emptyset$; that is, $I$ satisfies $r$ viewed as implication in classical logic.

An interpretation $I$ is then an answer set of $P$, if $I$ is a $\subseteq$-minimal model of the program $P^I = \{r \in P \mid I$ satisfies neither $(ii)$ nor $(iii)\}$. Intuitively, $I$ must result by applying the rules $r$ whose bodies "fire" w.r.t. $I$ starting from facts.

The semantics of programs with variables is defined in terms of their groundings (uniform replacement of variables in rules with all possible ground terms).

ASP features further constructs such as choice rules (which allows to select among alternatives under cardinality bounds) and weak constraints (i.e., soft constraints expressing costs for an objective function that is minimised); notably, the latter allow for modeling numeric uncertainty and to single out the most likely from answer sets of a program or a range of most likely answer sets. For more details on ASP, we refer to (Brewka, Eiter, and Truszczynski 2011; Calimeri et al. 2020).

---

[1]We use clingo (v. 5.6.2 ) from https://potassco.org/.

## 3.2 Zero-Shot VQA for the GQA Dataset

Next, we explain our system for zero-shot VQA for the GQA dataset (Hudson and Manning 2019). It does not require any training, but relies on foundation models for processing the visual scene and ASP for deducing answers.
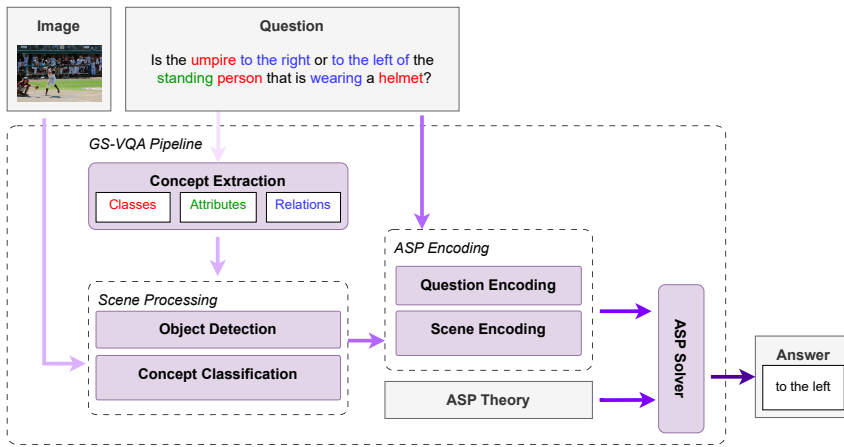
**The GQA dataset.** We use the state-of-the-art GQA dataset, which has been widely adopted in the recent literature (Amizadeh et al. 2020; Surís, Menon, and Vondrick 2023; Liang et al. 2020; Li et al. 2023). It contains over 22M open and binary questions that are complex in structure, involve a wide variety of reasoning skills, and have a large number (1 878) of possible answers. The questions cover more than 100 000 images from the Visual Genome dataset (Krishna et al. 2017) that present real-world scenes with a wide variety of object classes, attributes, and relations. GQA comes with two types of supplementary data that greatly aid in the development of our pipeline: First, each natural-language question from the test split comes with a functional representation of its required reasoning steps. Second, a Visual Genome scene graph is provided for every image in the test split of the dataset, which allows us to verify soundness of our ASP encoding under perfect visual information.

**Solving GQA.** Our system for solving GQA, shown in Fig 2a, resembles other modular neurosymbolic models consisting of modules for language, vision, and reasoning. We next describe these modules, a performance evaluation is relegated to the appendix. As mentioned, one of the advantages of using GQA is that we already have a functional representation that we can use for every question. Neurosymbolic systems have already shown that even classic neural networks such as the LSTM (Hochreiter and Schmidhuber 1997) are able to properly translate natural language into these representations. Moreover, LLMs can now be used for the task of semantic parsing, which may generalise much better than trained neural networks for specific datasets. We assume that the representation is given, and thus we only implement a script that translates it to our ASP representation. An example of this is shown in Fig. 2b under "Question Encoding". The predicates are linked by the numbers in the first argument, which represent steps to compose questions.

We next discuss the reasoning module, which uses as a fixed logical program. In a later section, we show how to use LLMs to extend such a logical theory to add functionality.

*The ASP theory.* The GS-VQA pipeline constructs symbolic encodings of both the input question and the input image, which we call *question encoding* and *scene encoding*, respectively. Similarly to a related approach (Eiter et al. 2022), we use ASP as the symbolic formalism for these encodings. This provides us not only with a mature ecosystem of tooling and solvers, but more importantly, allows us to capture the uncertainty in the class, attribute, and relation predictions of the scene-processing component.

The ASP theory consists of a set of rules that—in contrast to the question and scene encoding—do not change from question to question and encode the semantics of the reasoning operations that can appear as part of the question encoding.

| Image | Question |
|---|---|
| | Is the umpire to the right or to the left of the standing person that is wearing a helmet? |

**GS-VQA Pipeline**

**Concept Extraction**
Classes   Attributes   Relations

*Scene Processing*
Object Detection
Concept Classification

*ASP Encoding*
Question Encoding
Scene Encoding

ASP Theory

ASP Solver

Answer
to the left

(a) An overview over the full GS-VQA pipeline.

```
%Scene Encoding
object(o1).
has_obj_weight(o1, 1971).
has_attr(o1, class, person).
has_attr(o1, name, baseball_player).
{has_attr(o1, pose, standing)}.
...

%Question Encoding
scene(0).
select(1, 0, helmet).
relate(2, 1, person, wearing, subject).
filter_any(3, 2, standing).
choose_rel(4, 3, umpire, to_the_left_of,
        to_the_right_of, subject).
end(4).

%ASP Theory
state(TO,ID) :- select(TO, TI, C),
        state(TI, ID),
        has_attr(ID, _, C).
bool(TO,no) :- exist(TO, TI),
        not bool(TO,yes).
...
```

(b) Excerpts from the question encoding, scene encoding, and the ASP theory.

Figure 2: GS-VQA takes the image and question as input and uses a question-driven approach to generate a partial scene graph. We generate an ASP representation of both partial scene graph and question. These are then solved along an ASP theory to derive the correct answer.

Revisiting the rules in Figure 2b, there $T_i$ and $T_o$ are variables representing input/output step references, $ID$ represents an object id, $C$ a class, $A$ an attribute category, $V$ an attribute value, and $R$ a relation. The entire theory, described as part of the online repository, is solved alongside both scene and question encodings to produce an answer.

*Question-driven scene-graph generation.* The zero-shot nature of the pipeline presents a challenge for constructing a complete scene graph of visually complex scenes: due to the general-purpose nature, inference with the VLMs that GS-VQA uses for scene graph generation is far more resource intensive than adhoc trained models. As such, constructing a full scene graph in which likelihoods for all possible classes, attributes, and relations in the dataset are present for every object detected in the scene is untenable within a time bound that a human user might find acceptable.

To resolve this issue, the pipeline resorts to *"question-driven" partial scene-graph extraction*, where only information is extracted from the scene that is relevant for answering the question at hand. To this end, the *concept-extraction component* determines which object classes, attributes, and relations are relevant from the semantic representation of the input question. Conceptually, it takes a question in natural language and produces a tuple $(C, A, R)$, where $C$ is a set of classes, $A$ is a set of attribute categories, and $R$ is a set of relations. We use the functional representation that comes with every question in GQA to accomplish this. For the example question in Fig. 2a, the tuple is $(\{\texttt{helmet}, \texttt{umpire}, \texttt{person}\}, \{\texttt{pose}\}, \{\texttt{wearing}, \texttt{to\_the\_left\_of}, \texttt{to\_the\_right\_of}\})$.

Using the $(C, A, R)$ output tuple of the concept-extraction component, the *scene-processing component* has the task of extracting a question-driven partial scene graph. We represent the graph as a list $O = [o_1, \ldots, o_n]$ of objects $o_i = (id, s, B, c, A_o, R_o)$, each having a unique identifier $id$, a score $s$ between 0 and 1 denoting the confidence in the object detection from scene-processing, a bounding box $B$,

a class $c$, and sets $A_o$ and $B_o$ of attribute and relation likelihoods, respectively. The class $c$ is either in $C$ or a sub-class of maximal specificity of a class $c'$ in $C$. The latter ensures that an object cannot, e.g.,, be detected as just a "person", but must be detected as a maximally specific class like "baseball player". It also means that the scene graph contains only objects of classes that are deemed relevant to answering the question (hence the description as a "partial" scene graph). The set $A_o$ contains, for each possible value $v$ of each attribute category $a$ in $A$, a likelihood between 0 and 1 that $v$ applies to $a$ for the object. Finally, $R_o$ contains, for each relation $r \in R$ and each other detected object $o_j \neq o_i$, a likelihood between 0 and 1 that $r(o_i, o_j)$ applies. The scene graph $O = [o_1, o_2, \ldots, o_n]$ generated by when processing the scene is easily translated using a script into our ASP representation. Fig. 2b shows an excerpt of such representation under "Scene Encoding."

## 3.3 VQA for the CLEVR Dataset

The second dataset we consider is CLEVR (Johnson et al. 2017), which uses synthetic scenes but challenges VQA systems with more complex compositional questions. We use a VQA system for CLEVR (Eiter et al. 2022), that we have extended by an explanation component in recent work (Eiter et al. 2023). We revisit its basic functionality in the remainder of this section.

**The CLEVR dataset.** CLEVR was designed to test VQA system with compositional questions that involve making several reasoning steps to derive the correct answer. The dataset contains synthetically generated images with different objects in it. These objects vary in their shape (cube, cylinder, sphere), colour (brown, blue, cyan, gray, green, purple, red, yellow), size (big, small), and material (metal, rubber).

CLEVR questions require, e.g., identifying objects, counting, filtering for attributes, comparing attributes, and spatial reasoning. They are formulated in natural language, but, as for GQA, a functional representation is also provided that

can be directly parsed into ASP facts. For illustration, the question "How many large things are either cyan metallic cylinders or yellow blocks?" then becomes

```
end(8). count(8,7). filter_large(7,6). union(6,3,5).
filter_cylinder(3,2). filter_cyan(2,1).
filter_metal(1,0).
filter_cube(5,4). filter_yellow(4,0). scene(0).
```

which encodes an execution tree of operations to derive the answer, where indices refer to output (first argument) and input (remaining arguments) of the respective operations.

**Solving CLEVR.** The architecture of the system for CLEVR is similar to the one for solving GQA. In fact, the latter system was inspired by the design of the former. The scenes in CLEVR are however less complex, and we use the popular object-detection framework YOLOv5[2] to identify all objects in the image, their attributes, and further bounding-box information.

As for GQA, ASP is used for the reasoning module. More specifically, we use a uniform ASP encoding that describes how to derive the answer for a question and a scene, both given as translations into ASP facts, by step-wise evaluating the execution tree of operations from the question encoding. More information, code, and the full ASP encoding can be found online.[3]

## 4  Knowledge Distillation Method

Our VQA approaches use a hand-coded ASP theory that correctly computes the answer to any question in the dataset, given a correct representation of both the question and the image. The ASP encoding is constructed around a fixed dataset. If the dataset is extended, rules need to be modified or added to handle new examples. In general, these new rules must be crafted by a human; here, an LLM based system can come to aid and provide automated support. To this end, we aim for a reasoning module that manages the theory by being able to recognise which examples it can handle and which it cannot, and in the latter case, by adding rules in a way such that the example can be solved. We propose a method of *declarative knowledge distillation*, where the model we distil from is an LLM, and the knowledge that is distilled from it is represented in ASP rules.

### 4.1  Preprompt

We first present a preprompt that instructs the LLM to only return ASP rules that extend an initial theory Init. Theory Init is a partial encoding for the task at hand that we want to extend. Our preprompt consists of several components:

1. **Introduction**: We present the setting of VQA and clarify that we have already parsed both scene and question into correct representations.
2. **Language Syntax**: We describe the syntax of the language we use to represent questions and scenes, in our case ASP.

3. **Scene and Question Explanation**: We explain the representation that is used for the scene graph and questions and give examples.
4. **Answer Format**: We describe the format of the answers to the questions.
5. **Initial Theory**: We present the initial theory Init that must be updated as necessary.
6. **Task Explanation**: Finally, we explain the input the LLM will receive from now on and the expected response (see Listing 1).

The input after the preprompt are question/scene/answer tuples $(Q, S, A)$ in the language of ASP. The task is the following: when our system recognises that the current theory cannot handle the instance presented, then the LLM is prompted to add rules such that the correct answer can be derived. The expected LLM output is a list of ASP rules.

```
Your task is to keep the ASP theory
updated with rules that allows us
to answer questions.
We provide an initial theory that
can handle some instances.
The prompt input will consist of one or
more questions in the ASP representation.

Strictly follow these guidelines:
1. Only output the new ASP Rules.
2. Do not add facts as rules.
3. New rules should be as general
as possible, i.e., have a low number of
constants and high number of variables.
4. Do not output any natural language.
```

Listing 1: Excerpt from the Task Explanation part of our preprompt.

### 4.2  Rule Distillation Algorithm

With the task explained to the LLM by the preprompt, we can start to present examples from the dataset. For each question/scene/answer tuple $p = (Q, S, A)$, we do the following steps (and repeat them for the same example at most $r$ times if not successful):

1. **Prompting:** We prompt the LLM with $p$ as additional input, and we get as a response $R$.
2. **Solving:** We concatenate $R$ with the initial theory to get theory Res. Then, we run an ASP solver on theory Res alongside the instance pair $(Q, S)$.
   (a) **Syntax Check:** If we get a syntax error, we pass the error message to the LLM and prompt it to revise $R$ accordingly. We try this at most $m$ times.
   (b) **Semantic Check:** We check whether the answer we get from the solver is correct, i.e., coincides with $A$. If not, we pass the actual answer and the expected answer to the LLM and task it to update $R$; we try this at most $m$ times.
3. **Regression Testing:** To avoid that adding rules to the theory renders past examples incorrect, we test Res on all previously seen examples. Only if all tests pass we keep Res, otherwise we disregard the extension $R$.

The algorithm has two parameters, $r$ is the number of retries per example, and $m$ is the number of retries for mending incorrect rules (defaults are $r = m = 1$). Mending rules is potentially expensive as it requires more calls to the LLM; it can be turned off if preferred.

**Example Sampling Strategies.** VQA datasets contain millions of instances, and going blindly through them can make the distillation process ineffective. Choosing a small but representative sample can yield better results faster. We propose two strategies to group instances:

- *Predicate Count*: We group all the instances by the number of predicate occurrences that appear in the ASP question representation. For this, we create a dictionary whose keys are the numbers of predicates and the contents are questions with such length.
- *Predicate Relevance*: Here, we group examples based on the predicates that appear in the question representation. We first create a dictionary whose keys are all the predicates that appear in any question representation. Then we populate the dictionary with questions where the key predicate appears in the question representation.

We then sample examples from the group created for the chosen strategy with a parameter $k$. For the former, the total number of examples is $k$ multiplied by the number of keys in the dictionary, while for the latter, the total number of examples is exactly $k$.

**Batch Optimization.** We present prompt instances one by one, which results in one LLM call for each example. This is not very efficient and can misguide the LLM into implementing rules that only solve that particular example, yet our aim is to have general rules that can handle a considerable portion of the dataset. Considering that LLMs have increasingly larger context sizes, we also investigate the option of presenting prompt instances in batches, where each batch contains up to $b$ singular instances. We observe that the scene representation is usually much larger than the question representation and only contains a small number of predicates, and the variance comes more from the constants in them. Considering this, whenever we use our batch strategy, we present the prompt instance $p = \{Q_1, \ldots, Q_b\}$, where we include only the question representation in the $Q_i$ and drop the scene representations. Now the LLM must create rules general enough to pass the semantic check for all the examples in the batch. However, there is an expected trade-off between batch size and accuracy: With large $b$, the semantic check and regression testing might be too strict for the LLM to produce rules that correctly cover all examples in one shot.

## 5 Knowledge Distillation Experiments

We conduct a series of experiments to evaluate our knowledge distillation method on GQA and CLEVR to answer the following research questions:[4]

**(R1)** Given an ASP-based VQA system and a VQA task, can our approach extend the ASP reasoning component to deal

with questions that require reasoning operations/steps that are not yet implemented?

**(R2)** What LLMs are suitable for our method?

**(R3)** Can our method cope with the challenge of removing, either randomly or in a more controlled way, increasingly large parts from an initial complete theory?

**(R4)** What are the effects of the more resource-friendly batch processing variant and mending switched off for our method?

The evaluation platform is a workstation with an Intel Core i7-12700K CPU, 32GB of RAM, and an NVIDIA GeForce RTX 3080 Ti GPU with 12GB of video memory. All experiments were run 5 times. We include average accuracy, standard deviation, as well as the minimum and maximum value obtained. For reproducibility, we logged all our parameters, random seeds, and input prompts.

**LLM selection.** Before going into the details of our knowledge distillation experiments, we describe how we selected LLMs for further consideration.

We ran the ASP programs of our VQA systems on examples from GQA and CLEVR. Then we selected ca. $45k$ examples where the answer was correctly calculated for GQA, and then divided this set into a training and a test suite of ca. $35k$ and $10k$ instances, respectively. For CLEVR, we selected $50k$ examples and split them into $35k$ for training and $15k$ for testing.[5]

For the selected examples, we ran preliminary experiments on a large array of LLMs, both local and online API-based ones. Local models, such as GPT4ALL (Anand et al. 2023) "wizardlm-13b", showed very poor performance when prompted to produce ASP rules. Some API models, like "mistral-medium", where too slow in coming up with a response for our purposes and were thus excluded.

We selected the three top performers, which were GPT-4 (OpenAI 2023), GPT-3.5, and Mistral (Jiang et al. 2023); more specifically, the models used are "gpt-4-1106-preview", "gpt-3.5-turbo-1106" and "mistral-small", respectively.

**Experiment 1.** Our first experiment is designed to address **(R1)** and **(R2)**. We start with the complete ASP encoding for each VQA approach presented in Section 3 and remove all rules that mention a particular predicate $P$ that occurs in some question representation. We then prompt the LLM with examples that contain $P$ in their question representation to repair the theory. This simulates a scenario where the initial ASP theory is not yet capable to address all requirements of the VQA task and needs to be extended.

We use the predicate ordering strategy with a sample size of $k = 10$. As we only sample questions where $P$ is present, the number of examples used for each run is $k$.

The results for GQA and CLEVR are shown in Table 1. Column Init $\setminus P$ shows the accuracy of the initial ASP encoding with a predicate $P$ removed. The drop in accuracy varies depending on the number of questions affected and the role of the predicate that was removed (e.g., "select" results in a deeper drop). The other columns show the performance of

---

[4]The code for reproducing our experiments is available as an online repository: https://github.com/pudumagico/KR2024.

[5]The expression "training set" refers here to the examples used when running the knowledge distillation method.

| $P$ | Init $\setminus P$ | GPT-4 | GPT-3.5 | Mistral |
|---|---|---|---|---|
| query | 48.84 | $97.67 \pm 18.05$ $(89.16, 98.92)$ | $70.02 \pm 19.36$ $(48.84, 85.53)$ | — |
| exist | 86.36 | $99.75 \pm 00.50$ $(98.86, 99.98)$ | $87.65 \pm 02.41$ $(86.36, 91.95)$ | $89.68 \pm 04.20$ $(86.36, 95.66)$ |
| or | 92.18 | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $93.03 \pm 01.90$ $(92.18, 96.44)$ | $93.20 \pm 01.66$ $(92.18, 96.02)$ |
| filter | 81.60 | $98.21 \pm 00.40$ $(97.49, 98.40)$ | $83.15 \pm 03.47$ $(81.60, 89.37)$ | $81.70 \pm 00.24$ $(81.60, 82.14)$ |
| choose_attr | 92.12 | $95.98 \pm 05.37$ $(88.73, 99.83)$ | $93.73 \pm 01.36$ $(92.31, 95.83)$ | $92.12 \pm 00.01$ $(92.12, 92.15)$ |
| verify_rel | 93.72 | $98.60 \pm 01.11$ $(96.73, 99.43)$ | — | — |
| select | 9.53 | $99.94 \pm 00.07$ $(99.87, 100.0)$ | $27.42 \pm 40.01$ $(9.53, 99.01)$ | — |
| negate | 98.59 | $98.54 \pm 00.20$ $(98.59, 98.74)$ | — | — |
| relate | 56.89 | $69.38 \pm 12.50$ $(56.89, 85.25)$ | — | — |
| two_different | 98.94 | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $99.39 \pm 00.55$ $(98.94, 100.0)$ | — |
| two_same | 98.83 | $99.99 \pm 00.00$ $(99.99, 100.0)$ | $99.05 \pm 00.53$ $(98.83, 100.0)$ | — |

(a) Results for GQA.

| $P$ | Init $\setminus P$ | GPT-4 | GPT-3.5 | Mistral |
|---|---|---|---|---|
| exist | 79.63 | $99.48 \pm 00.70$ $(98.72, 100.0)$ | $87.82 \pm 11.11$ $(98.72, 100.0)$ | $80.21 \pm 06.36$ $(72.16, 90.02)$ |
| unique | 29.19 | $97.67 \pm 18.05$ $(89.16, 98.92)$ | — | — |
| count | 98.01 | $99.60 \pm 00.88$ $(98.01, 100.0)$ | $98.01 \pm 01.12$ $(98.01, 98.40)$ | — |
| equal_integer | 96.61 | $99.92 \pm 00.17$ $(99.61, 100.0)$ | $97.80 \pm 01.26$ $(96.61, 99.60)$ | — |
| and | 93.67 | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $97.46 \pm 03.46$ $(93.67, 100.0)$ | — |
| relate_left | 84.73 | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $96.18 \pm 07.63$ $(84.73, 100.0)$ | $94.14 \pm 08.02$ $(84.73, 100.0)$ |
| filter_large | 68.54 | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $87.41 \pm 17.23$ $(68.54, 100.0)$ | $81.12 \pm 17.23$ $(68.54, 100.0)$ |
| query_shape | 72.23 | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $94.44 \pm 12.43$ $(72.23, 100.0)$ |
| same_color | 94.79 | $99.36 \pm 00.87$ $(98.41, 100.0)$ | $100.0 \pm 00.00$ $(100.0, 100.0)$ | $97.07 \pm 02.70$ $(94.79, 100.0)$ |

(b) Results for CLEVR.

Table 1: Results for the knowledge distillation method when attempting to restore Init after all rules that mention a predicate $P$ are removed.

| $s(\%)$ | Init | GPT-4 | GPT-3.5 |
|---|---|---|---|
| 10 | 26.57 | $94.67 \pm 02.21$ $(89.71, 95.67)$ | — |
| 20 | 63.54 | $75.56 \pm 11.86$ $(63.55, 90.22)$ | $66.14 \pm 07.78$ $(63.55, 89.48)$ |
| 50 | 7.17 | $47.48 \pm 15.26$ $(30.25, 71.64)$ | $24.43 \pm 12.28$ $(07.18, 46.88)$ |

(a) Results for GQA.

| $s(\%)$ | Init | GPT-4 | GPT-3.5 |
|---|---|---|---|
| 10 | 46.61 | $70.76 \pm 04.17$ $(66.59, 75.62)$ | $50.30 \pm 03.68$ $(46.62, 53.98)$ |
| 20 | 9.66 | $44.66 \pm 30.58$ $(14.16, 97.30)$ | $32.03 \pm 11.86$ $(13.44, 45.70)$ |
| 50 | 0.0 | $23.90 \pm 03.30$ $(18.06, 27.84)$ | $10.19 \pm 19.59$ $(00.00, 49.36)$ |

(b) Results for CLEVR.

Table 2: Knowledge distillation results when attempting to restore a complete ASP theory after a percentage $s$ of rules is randomly removed.

| $b$ | Light | Medium | Heavy |
|---|---|---|---|
| Init | 0.0 | 0.0 | 6.24 |
| 1 | $56, 26 \pm 10.23$ $(34.54, 61.28)$ | $81.45 \pm 05.07$ $(76.86, 87.91)$ | $83.85 \pm 02.49$ $(81.38, 87.77)$ |
| 2 | $32.71 \pm 04.31$ $(25.72, 43.15)$ | $79.83 \pm 03.42$ $(75.11, 83.03)$ | $74.32 \pm 02.91$ $(75.86, 80.54)$ |
| 5 | $16.62 \pm 05.28$ $(10.51, 17.59)$ | $69.68 \pm 31.12$ $(24.18, 82.19)$ | $84.25 \pm 04.59$ $(78.93, 89.48)$ |
| 10 | — | $15.38 \pm 12.30$ $(11.62, 31.75)$ | $84.75 \pm 04.20$ $(80.64, 90.85)$ |

(a) Results for GQA.

| $b$ | Light | Medium | Heavy |
|---|---|---|---|
| Init | 0.0 | 5.56 | 20.80 |
| 1 | $84.68 \pm 26.42$ $(38.23, 100.0)$ | $86.97 \pm 04.35$ $(83.89, 90.05)$ | $95.40 \pm 03.83$ $(91.25, 98.81)$ |
| 2 | $75.4 \pm 33.78$ $(27.84, 99.88)$ | $18.68 \pm 04.50$ $(15.67, 26.09)$ | $88.51 \pm 04.46$ $(83.37, 91.25)$ |
| 5 | $17.06 \pm 29.55$ $(00.00, 51.19)$ | $17.79 \pm 03.00$ $(15.67, 19.92)$ | $94.39 \pm 03.71$ $(91.33, 98.52)$ |
| 10 | — | — | $89.88 \pm 09.04$ $(77.68, 98.81)$ |

(b) Results for CLEVR.

Table 3: Results for the knowledge distillation method when using batch sizes $b$ and the different initial theories Light, Medium and Heavy.

the considered LLMs; here and in other tables, "—" means no improvement. By a large margin, GPT-4 is the most suit-able LLM for this task for both datasets. With GPT-4, we could obtain rules that improve over Init for every predicate

| $P$ | Init $\setminus P$ | GPT-4 | GPT-3.5 | Mistral |
|---|---|---|---|---|
| query | 48.84 | $99.02 \pm 00.04$ (99.02, 99.03) | $55.90 \pm 15.80$ (48.84, 84.17) | — |
| exist | 86.36 | $99.09 \pm 01.76$ (95.94, 99.97) | $87.83 \pm 03.29$ (86.36, 93.73) | $86.60 \pm 0.05$ (86.36, 87.57) |
| or | 92.18 | $100.0 \pm 00.00$ (100.0, 100.0) | $93.03 \pm 01.90$ (92.18, 96.44) | $94.41 \pm 3.35$ (92.18, 99.73) |
| filter | 81.60 | $98.56 \pm 00.58$ (98.47, 98.63) | $82.38 \pm 0.078$ (81.92, 82.50) | $84.99 \pm 7.59$ (81.60, 98.59) |
| choose_attr | 92.12 | $98.65 \pm 02.65$ (93.91, 99.88) | $95.16 \pm 04.26$ (92.03, 99.84) | $92.08 \pm 0.06$ (91.98, 92.12) |
| verify_rel | 93.72 | $95.74 \pm 03.49$ (93.72, 99.08) | $94.30 \pm 01.17$ (93.72, 96.06) | — |
| select | 9.53 | $81.69 \pm 36.81$ (9.53, 100.0) | $28.94 \pm 39.82$ (9.53, 100.0) | — |
| negate | 98.59 | $98.72 \pm 00.02$ (98.59, 99.12) | — | — |
| relate | 56.89 | $58.02 \pm 02.19$ (57.54, 61.91) | $57.29 \pm 00.09$ (56.89, 58.91) | — |
| two_different | 98.94 | $100.0 \pm 00.00$ (100.0, 100.0) | — | — |
| two_same | 98.83 | $99.60 \pm 00.02$ (99.50, 100.0) | $99.09 \pm 00.05$ (98.83, 100.0) | — |

Table 4: Ablation study for GQA: Results when attempting to restore Init after removing rules for $P$ without mending step.

| $s(\%)$ | Init | GPT-4 | GPT-3.5 |
|---|---|---|---|
| 10 | 26.57 | $83.06 \pm 23.26$ (36.61, 95.80) | — |
| 20 | 63.54 | $55.46 \pm 14.84$ (26.99, 69.84) | $38.67 \pm 21.00$ (08.48, 64.96) |
| 50 | 7.17 | $36.38 \pm 10.62$ (18.15, 48.55) | $04.21 \pm 03.42$ (00.00, 7.94) |

Table 5: Ablation study for GQA: Attempting to restore theory $T$ after $s$ percent of rules were randomly removed without mending.

| $b$ | Light | Medium | Heavy |
|---|---|---|---|
| Init | 0.0 | 0.0 | 6.24 |
| 1 | $51, 43 \pm 08.56$ (42.72, 59.85) | $80.41 \pm 05.34$ (74.54, 85.01) | $76.70 \pm 00.76$ (75.82, 77.19) |
| 2 | $27.06 \pm 09.60$ (21.09, 38.14) | $77.70 \pm 05.42$ (75.25, 83.92) | $77.60 \pm 03.85$ (73.41, 81.00) |
| 5 | $15.23 \pm 02.28$ (12.60, 16.49) | $60.16 \pm 34.17$ (21.08, 84.45) | $86.28 \pm 10.84$ (27.93, 94.71) |
| 10 | — | $19.55 \pm 07.41$ (13.03, 27.62) | $73.93 \pm 06.50$ (66.42, 77.69) |

Table 6: Ablation study for GQA: Results when using batch sizes $b$ and different initial theories without mending step.

$P$. For the other models, there the quality of suggested repairs differs largely. GPT-3.5 is also capable of repairing Init but to a lesser extent than GPT-4, as it does not produce any new rules for some predicates. When it does, the accuracy is lower than the one of GPT-4. The Mistral model performs similarly to GPT-3.5 when it finds the correct rules. This, however, happened less often and the gain in accuracy then is minimal.

**Experiment 2.** This experiment is designed to address **(R3)**. It is a challenge experiment where we took the complete ASP encodings and then removed a random sample of $s$ percent of the rules from it. For GQA, the ASP theory consists of 60 rules; for CLEVR, the program comprises 72 rules. After rendering this theory incomplete, we prompted the LLM to restore it. We tested this setting for $s = \{10, 20, 50\}$ on the two best models from the first experiment, i.e., GPT-4 and GPT-3.5. We used the length ordering strategy with $k = 2$, which yields 20 examples per run, and $r = 1$ retries.

The results are given in Table 2. For GQA, GPT-4 is quite capable of producing ASP rules that improve the accuracy of the initial theory. However, GPT-3.5 starts to falter for GQA, as the gain in accuracy is dramatically reduced for $s = 20, 50$, and no gain is reported when $s = 10$. In the case of GQA, the initial accuracy of the theory with $s = 10$ is higher than for $s = 20$. Even though we remove more rules in the latter, some rules affect larger portions of the questions.

**Experiment 3.** This experiment tests our batch optimization approach and aims at answering **(R4)**. We consider batch sizes $b = 2, 5, 10$ and the length ordering strategy with $k = 11$, which yields 100 examples per run. We use $r = 3$ retries for examples and $m = 2$ retries for mending.

Like the previous experiment, this one is also based on the complete ASP theory, but we now selected parts of it by hand to generate three different initial theories. By their size, we call them Light, Medium and Heavy. Theory Light consists of only five rules that tell our system how to start and end the processing of a question, but nothing else. We added more rules to Light to generate Medium and even more rules to Heavy. For this experiment, we focused only on the most capable model, namely GPT-4.

Table 3 shows that GPT-4 still retains its ability to produce meaningful ASP rules even in this very challenging scenario. We can observe that small theories do not work well with bigger batches, as they do not provide enough background for the LLM to produce suitable rules. When the size of the initial theory increases, the LLM can handle bigger batches and produces rules of higher quality.

**Ablation Experiment.** We finally study the effect of the mending step in our method when the LLM suggests rules that are either syntactically or semantically not correct. As mentioned in the previous section, these checks can be turned off, which results in fewer LLM calls.

We only present the results for the previous three experiments on GQA in Tables 4–6; for CLEVR, they look similar. As one would expect, the results are worse most of the time, although not always, when mending is disabled. The rate of

improvement with mending ranges from a couple of percentage points to up to ca. 20% (row for "select").

**Discussion.** We turn back to our research questions from the beginning of this section.

**(R1)** Experiment 1 shows that LLMs are capable of completing an ASP program that has all rules for a single operation removed. This is the case when a dataset is extended with new examples that require reasoning operations that are not yet encoded.

**(R2)** Regarding the suitability of different LLMs for declarative knowledge distillation, we conclude that only grand-scale LLMs, with GPT-4 currently the market leader, are able to tackle this problem effectively. Arguably, the LLMs that we used have more knowledge about mainstream programming languages such as Python than logical programming languages. It will be interesting so see whether small, self-hosted language models will eventually catch up in the future.

**(R3)** When challenged with restoring increasingly large parts of an ASP theory, the current approach reaches its limits. Only the most powerful model we used is still able to produce sound ASP rules.

**(R4)** Our experiments on batch processing and the ablation study helped to illuminate the trade off between performance and reducing the number LLM calls. In conclusion, better performance can be bought by using more prompts, which can be expensive if a subscription-based LLM is used.

## 6   Conclusion

We have presented an approach for declarative knowledge distillation using LLMs to find rules that extend the reasoning component of a VQA system to extend its capabilities. This process uses examples from a dataset as guidance and relies entirely on prompting without the need to train or fine-tune the used language models. We have demonstrated this approach on the prominent CLEVR (Johnson et al. 2017) and GQA (Hudson and Manning 2019) datasets. The benefit of using logic-based methods in combination with foundation models is that we obtain systems that behave in an interpretable and verifiable way to ensure correct reasoning. Logical rules are intuitive, and they can helpful in VQA architectures to create advanced reasoning capabilities, including explainability. Our knowledge distillation method shows promise for automating the process of ASP modelling for VQA, a complex scenario that requires understanding of intricate representations.

For future work, we want to study further VQA datasets, but we also want to explore other tasks than VQA that potentially benefit from distilling rules from LLMs. As using API-based LLMs can be expensive, we want to look into balancing performance of the distillation approach and the number of LLM calls. Enhancing the LLM approach with concepts from ILP and a possible combination would also be interesting to explore.

## Appendix

We provide an evaluation of our logic-based VQA approach, which we call GS-VQA. We evaluated the GS-VQA pipeline

| Model | Category | Accuracy |
|---|---|---|
| BLIP-2 | end-to-end | 44.7% |
| CodeVQA | question-symbolic | 49.0% |
| FewVLM | end-to-end | 29.3% |
| GS-VQA (ours) | neurosymbolic | 39.5% |
| PnP-VQA | semi-symbolic | 42.3% |
| ViperGPT | question-symbolic | 48.1% |

Table 7: Comparison of GS-VQA's accuracy on the test-dev set of GQA with other state-of-the-art zero-shot approaches for VQA.

on the balanced test-dev set of GQA, which contains $12\,578$ questions. The evaluation uses the larger ViT-L/14 variant of OWL-ViT (Minderer et al. 2022) for object detection and the smaller ViT-B/32 variant of CLIP (Radford et al. 2021) for concept classification.

In Table 7, we present a comparison between our system and other models on GQA. We consider only zero-shot models and classify them into four sub-categories:

- **End-to-End**: End-to-end systems are those that rely solely on neural networks for computing the answer.
- **Neurosymbolic**: Neurosymbolic systems like ours are those that combine both neural networks for parsing data and symbolic execution to calculate the answers.
- **Question-Symbolic**: Such methods extract a symbolic representation from only the input question, usually in the form of some programmatic specification of the reasoning steps needed to arrive at the answer of the question.
- **Semi-Symbolic**: PnP-VQA (Tiong et al. 2022) extracts a symbolic representation of the image but does not perform its reasoning purely symbolically, hence we classify this method as semi-symbolic.

GS-VQA answers 39.5% of all questions of GQA's test-dev set correctly, with the current best zero-shot VQA model, CodeVQA (Subramanian et al. 2023), obtaining an accuracy of 49.0%, followed closely by ViperGPT (Surís, Menon, and Vondrick 2023). However, CodeVQA and ViperGPT translate input questions into Python code that may contain queries to another VQA model; thus the performance of the latter, which is PnP-VQA (Tiong et al. 2022) resp. BLIP-2 (Li et al. 2023), should be considered as their baseline.

While our model is slightly behind the baseline in terms of accuracy by a couple of percentage points, it uses ASP for deducing answers which comes with a range of advantages, where explainability and non-determinism to deal with ambiguous inputs are the most important ones (Eiter et al. 2022). More specifically, ASP allows for transparent execution, verifiability, and transferability, as well as the possibility of reasoning under different modalities, e.g., abduction on instances for computing contrastive explanations (Eiter et al. 2023). Furthermore, the components used for our system were run locally, in contrast to others such as ViperGPT which hinge on extensive external resources. Finally, since our architecture is modular, components can be easily replaced by better ones, and it can be expected that its performance improves when the underlying foundation models get better.

# References

Abraham, S. S.; Alirezaie, M.; and Raedt, L. D. 2024. CLEVR-POC: reasoning-intensive visual question answering in partially observable environments. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, 3297–3313. ELRA and ICCL.

Amizadeh, S.; Palangi, H.; Polozov, A.; Huang, Y.; and Koishida, K. 2020. Neuro-Symbolic Visual Reasoning: Disentangling "Visual" from "Reasoning". In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, 279–290. PMLR.

Anand, Y.; Nussbaum, Z.; Treat, A.; Miller, A.; Guo, R.; Schmidt, B.; Community, G.; Duderstadt, B.; and Mulyar, A. 2023. Gpt4all: An ecosystem of open source compressed language models. *CoRR* abs/2311.04931.

Antol, S.; Agrawal, A.; Lu, J.; Mitchell, M.; Batra, D.; Zitnick, C. L.; and Parikh, D. 2015. VQA: Visual Question Answering. In *2015 IEEE International Conference on Computer Vision(ICCV)*, 2425–2433. IEEE Computer Society.

Barra, S.; Bisogni, C.; Marsico, M. D.; and Ricciardi, S. 2021. Visual question answering: Which investigated applications? *Pattern Recognit. Lett.* 151:325–331.

Basu, K.; Shakerin, F.; and Gupta, G. 2020. AQuA: ASP-Based Visual Question Answering. In *22nd International Symposium on Practical Aspects of Declarative Languages (PADL 2020)*, volume 12007 of *LNCS*, 57–72. Springer.

Bauer, J. J.; Eiter, T.; Ruiz, N. H.; and Oetsch, J. 2023. Neuro-Symbolic Visual Graph Question Answering with LLMs for Language Parsing. In *Workshop on Trends and Applications of Answer Set Programming (TAASP 2023)*.

Brewka, G.; Eiter, T.; and Truszczynski, M. 2011. Answer set programming at a glance. *Commun. ACM* 54(12):92–103.

Bruno, P.; Calimeri, F.; Marte, C.; and Manna, M. 2021. Combining deep learning and asp-based models for the semantic segmentation of medical images. In *Proceedings of the 5th International Joint Conference on Rules and Reasoning, RuleML+RR 2021*, volume 12851 of *Lecture Notes in Computer Science*, 95–110. Springer.

Calimeri, F.; Faber, W.; Gebser, M.; Ianni, G.; Kaminski, R.; Krennwallner, T.; Leone, N.; Maratea, M.; Ricca, F.; and Schaub, T. 2020. ASP-core-2 input language format. *Theory and Practice of Logic Programming* 20(2):294–309.

Cropper, A.; Dumancic, S.; Evans, R.; and Muggleton, S. H. 2022. Inductive logic programming at 30. *Mach. Learn.* 111(1):147–172.

Eiter, T.; Higuera, N.; Oetsch, J.; and Pritz, M. 2022. A neuro-symbolic ASP pipeline for visual question answering. *Theory and Practice of Logic Programming* 22(5):739–754.

Eiter, T.; Geibinger, T.; Higuera, N.; and Oetsch, J. 2023. A logic-based approach to contrastive explainability for neurosymbolic visual question answering. In *Proceedings of the 32nd International Joint Conference on Artificial Intelligence (IJCAI 2023)*, 3668–3676. ijcai.org.

Gelfond, M., and Lifschitz, V. 1988. The stable model semantics for logic programming. In *Proceedings of the 5th International Conference and Symposium on Logic Programming*, 1070–1080. MIT Press.

Goyal, Y.; Khot, T.; Summers-Stay, D.; Batra, D.; and Parikh, D. 2017. Making the V in VQA Matter: Elevating the Role of Image Understanding in Visual Question Answering. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6325–6334. IEEE Computer Society.

Gutmann, B., and Kersting, K. 2006. TildeCRF: Conditional random fields for logical sequences. In *Proceedings of the 17th European Conference on Machine Learning (ECML 2006)*, volume 4212 of *Lecture Notes in Computer Science*, 174–185. Springer.

Hadl, J. 2023. GS-VQA: Zero-shot neural-symbolic visual question answering with vision-language models. Master's thesis, Technische Universität Wien, Wien.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Hudson, D. A., and Manning, C. D. 2019. GQA: A New Dataset for Real-World Visual Reasoning and Compositional Question Answering. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 6700–6709. Computer Vision Foundation / IEEE.

Ishay, A.; Yang, Z.; and Lee, J. 2023. Leveraging large language models to generate answer set programs. In *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, 374–383.

Jiang, A. Q.; Sablayrolles, A.; Mensch, A.; Bamford, C.; Chaplot, D. S.; de las Casas, D.; Bressand, F.; Lengyel, G.; Lample, G.; Saulnier, L.; Lavaud, L. R.; Lachaux, M.-A.; Stock, P.; Scao, T. L.; Lavril, T.; Wang, T.; Lacroix, T.; and Sayed, W. E. 2023. Mistral 7b.

Johnson, J.; Hariharan, B.; van der Maaten, L.; Fei-Fei, L.; Zitnick, C. L.; and Girshick, R. B. 2017. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 1988–1997. IEEE Computer Society.

Johnston, P.; Nogueira, K.; and Swingler, K. 2023. NS-IL: neuro-symbolic visual question answering using incrementally learnt, independent probabilistic models for small sample sizes. *IEEE Access* 11:141406–141420.

Krishna, R.; Zhu, Y.; Groth, O.; Johnson, J.; Hata, K.; Kravitz, J.; Chen, S.; Kalantidis, Y.; Li, L.; Shamma, D. A.; Bernstein, M. S.; and Fei-Fei, L. 2017. Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations. *International Journal of Computer Vision* 123(1):32–73.

Law, M.; Russo, A.; Bertino, E.; Broda, K.; and Lobo, J. 2020. Fastlas: Scalable inductive logic programming incorporating domain-specific optimisation criteria. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence,*

*AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, 2877–2885. AAAI Press.

Law, M.; Russo, A.; and Broda, K. 2020. The ILASP system for inductive learning of answer set programs. *CoRR* abs/2005.00904.

Li, J.; Li, D.; Savarese, S.; and Hoi, S. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models.

Liang, W.; Niu, F.; Reganti, A.; Thattai, G.; and Tur, G. 2020. LRTA: A Transparent Neural-Symbolic Reasoning Framework with Modular Supervision for Visual Question Answering.

Lifschitz, V. 2019. *Answer Set Programming*. Springer.

Lin, Z.; Zhang, D.; Tao, Q.; Shi, D.; Haffari, G.; Wu, Q.; He, M.; and Ge, Z. 2023. Medical visual question answering: A survey. *Artif. Intell. Medicine* 143:102611.

Mao, J.; Gan, C.; Kohli, P.; Tenenbaum, J. B.; and Wu, J. 2019. 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)*. OpenReview.net.

Minderer, M.; Gritsenko, A. A.; Stone, A.; Neumann, M.; Weissenborn, D.; Dosovitskiy, A.; Mahendran, A.; Arnab, A.; Dehghani, M.; Shen, Z.; Wang, X.; Zhai, X.; Kipf, T.; and Houlsby, N. 2022. Simple Open-Vocabulary Object Detection. In *Computer Vision - ECCV 2022 - 17th European Conference*, volume 13670 of *Lecture Notes in Computer Science*, 728–755. Springer.

Muggleton, S. H., and Raedt, L. D. 1994. Inductive logic programming: Theory and methods. *J. Log. Program.* 19/20:629–679.

Muggleton, S. H. 1991. Inductive logic programming. *New Gener. Comput.* 8(4):295–318.

OpenAI. 2023. GPT-4 technical report.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; Krueger, G.; and Sutskever, I. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*, volume 139 of *Proceedings of Machine Learning Research*, 8748–8763. PMLR.

Raedt, L. D., and Kersting, K. 2017. Statistical relational learning. In *Encyclopedia of Machine Learning and Data Mining*. Springer. 1177–1187.

Rajasekharan, A.; Zeng, Y.; Padalkar, P.; and Gupta, G. 2023. Reliable natural language understanding with large language models and answer set programming. In *Proceedings 39th International Conference on Logic Programming, ICLP 2023, Imperial College London, UK, 9th July 2023 - 15th July 2023*, volume 385 of *EPTCS*, 274–287.

Subramanian, S.; Narasimhan, M.; Khangaonkar, K.; Yang, K.; Nagrani, A.; Schmid, C.; Zeng, A.; Darrell, T.; and Klein, D. 2023. Modular Visual Question Answering via Code Generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 747–761. Association for Computational Linguistics.

Surís, D.; Menon, S.; and Vondrick, C. 2023. ViperGPT: Visual Inference via Python Execution for Reasoning. *CoRR* abs/2303.08128.

Tiong, A. M. H.; Li, J.; Li, B.; Savarese, S.; and Hoi, S. C. H. 2022. Plug-and-Play VQA: Zero-shot VQA by Conjoining Large Pretrained Models with Zero Training. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, 951–967. Association for Computational Linguistics.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L. u.; and Polosukhin, I. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Wang, Z.; Yu, J.; Yu, A. W.; Dai, Z.; Tsvetkov, Y.; and Cao, Y. 2022. SimVLM: Simple Visual Language Model Pretraining with Weak Supervision.

Yi, K.; Wu, J.; Gan, C.; Torralba, A.; Kohli, P.; and Tenenbaum, J. 2018. Neural-Symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, 1039–1050.

Zhao, W. X.; Zhou, K.; Li, J.; Tang, T.; Wang, X.; Hou, Y.; Min, Y.; Zhang, B.; Zhang, J.; Dong, Z.; Du, Y.; Yang, C.; Chen, Y.; Chen, Z.; Jiang, J.; Ren, R.; Li, Y.; Tang, X.; Liu, Z.; Liu, P.; Nie, J.; and Wen, J. 2023. A survey of large language models. *CoRR* abs/2303.18223.

Zhu, Z.; Xue, Y.; Chen, X.; Zhou, D.; Tang, J.; Schuurmans, D.; and Dai, H. 2023. Large language models can learn rules. *CoRR* abs/2310.07064.