Visual Question Answering with DeepProbLog

Jorrit Willaert[1]
[1]Catholic University of Leuven, Leuven – Belgium
e-mail: jorrit.willaert@student.kuleuven.be

***Abstract* – TODO**

***Keywords* – Neuro Symbolic AI, Visual Question Answering, DeepProbLog, Problog, Convolutional Neural Networks**

I. INTRODUCTION

The Neuro Symbolic AI community is growing rapidly, indicating that people start to recognize the value of the field. The Neuro Symbolic AI field is interested in building a bridge between the robustness of probabilistic knowledge, with the well-known popularity and proven strengths of deep neural networks. DeepProbLog [1] offers this ability, by using both the strengths of neural networks (i.e. system 1, typical subconscious tasks such as visual recognition, the processing of languages, ...), along with the strengths of rule-based probabilistic systems (i.e. system 2, slow, sequential thinking such as the derivation of a proof).

This paper elaborates on an application that requires both systems to be used, namely Visual Question Answering. System 1 will be required in order to gain an understanding of the image under investigation, with in particular their shapes and colors. System 2, on the other hand, will use this extracted information for deriving certain properties of objects[1], or even for capturing the relations[2] between the objects.

II. ORGANIZATION OF THE PAPER

III. LITERATURE SURVEY

The application focuses on Visual Question Answering (VQA), for which huge datasets are present, along with very sophisticated methods. The best known dataset for VQA is CLEVR [2], which contains 100k images with one million questions. An example image is given in Figure 1, while example questions are:

- Are there an equal number of large things and metal spheres?
- What size is the cylinder that is left of the brown metal thing that is left of the big sphere?
- How many objects are either small cylinders or metal things?

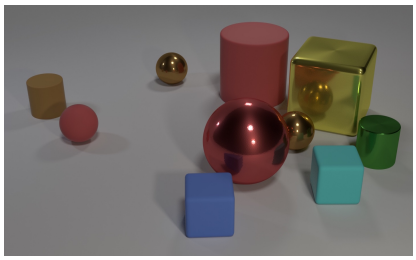

Fig. 1. A sample image from the CLEVR dataset [2]

Clearly, both system 1 and system 2 are actively used when answering these questions. One could wonder if neural networks alone could answer these questions without having an explicit system 2 encoding (i.e. the rule based knowledge base). Intuitively, it makes sense that if certain facts of the world are known[3], learning can proceed much more quickly[4]. Seen from an optimization viewpoint, errors made during prediction in this setup can be targeted exactly, which makes the optimization process also more targeted, and hence more efficient. Finally, this paper also provides evidence for these statements, since in Section Athe comparison between a VQA implementation with DeepProbLog is made with a purely CNN based approach.

This paper is based on the CLEVR dataset, but uses however a much more simplified version. In essence, it is almost like the Sort-Of-CLEVR dataset [3]. This Sort-Of-CLEVR dataset contains images such as in Figure 2, while asking questions such as:

- Non-relational questions: the shape, horizontal or vertical location of an object.
- Relational questions: shape of the closest/furthest object to the object under investigation, or the number of objects with the same shape.

[1]For example, finding the shape of the green object, or deriving if it is located on the left hand side of the image.

[2]Here, one could think of deriving if an object is located to the left of another object, or also for finding the number of circles in the image.

[3]They can be encoded, e.g. counting the number of spheres is simply a matter of detecting all the spheres in the image, after which a mathematical summation is a statement in the knowledge base.

[4]Not to say that learning might even be impossible if a lot of background knowledge is required.

Fig. 2. A sample image from the CLEVR dataset [3]

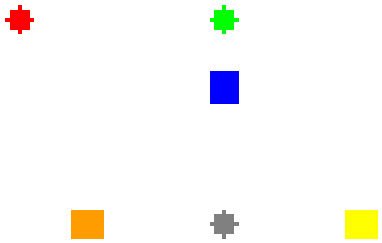

Fig. 3. A sample image from the dataset that has been used for this application

As explained earlier, both system 1 and system 2 are required for these types of VQA's.

Finally, since this application uses DeepProbLog, quite some time was spent in digesting the DeepProbLog paper [1], along with understanding the examples provided in the code repository [4].

IV. APPROACH

The implementation process involved three main parts:

1. Generation of data.
2. Creation of the logical part with DeepProbLog statements.
3. Linking the data and controlling the training process in pure Python code.

A. Generation of data

As mentioned in Section III, the data used in this application is based on the Sort Of CLEVR dataset, with one extra simplification. Given that the logical part will have to decide whether an object is located on the left side of an image, the neural network will have to convey positional information to the logical part. Hence, each discrete position will have to be encoded by a possible outcome of the neural network. Therefore, objects may only be located at certain places in a grid. In this application, a grid of 2x2, 4x4 and 6x6 has been used.

The data generator that was used for the creation of the Sort Of CLEVR dataset has been modified in order to place objects in the mentioned grid positions [3]. An example of a generated image is given in Figure 3.

Each specified color will have an object located somewhere in the grid, of which the shape can be a square or a circle.

These images are accompanied with three non-binary questions and one binary question. For each question, a random object is chosen. The possibilities are:

1. Non-binary - What is the shape of this object [5]?
2. Non-binary - Is this object located on the left side of the image?
3. Non-binary - Is this object located on the bottom side of the image?
4. Binary - How many objects have the same shape of this object?

These questions are encoded in a one-hot encoding, after which they are stored in a CSV file, along with the expected answers. A training set, a validation set, and a test set are generated.

B. Controlling the training process

The overall training process is controlled via the Python interfaces of DeepProbLog, along with general PyTorch implementations of the CNN's. First of all, CNN's are defined with PyTorch. A relatively simple network is used, where the input is given as a square RGB image of 100 pixels wide, which is transformed by the CNN into 72 output features for the 6x6 grid. Each color that is present in the image has its accompanied CNN network, hence the 72 output features encode the possible positions of the object with that color, along with their shape, which can be either square or circular ($6 \cdot 6 \cdot 2 = 72$).

The final thing (besides the logical rule encodings) required before commencing the training process, are the data loaders. The most challenging part here is the transformation from the generated data to specific query mappings and their outcome. One of the questions are chosen, and a correct mapping between a predicate in the logical rule encoding file and the answer is set up.

C. Logical rule encodings

Once the CNN belonging to a specific color has determined the position and the shape of that object, logical rules can deduce whether this object is located on the left hand side of

[5] I.e. the shape will be either a square or a circle.

the image, on the bottom side, and how many objects have the same shape. The logical rule program has been listed in Appendix A.

V. EXPERIMENTS

A. COMPARISONS WITH PURE SYSTEM 1 APPROACHES

The network based on pure neural predicates is able to recognize the questions quickly, however, seems to experience difficulties when having to decide for the correct answer. This can clearly be seen in the confusion matrix (Figure 4).

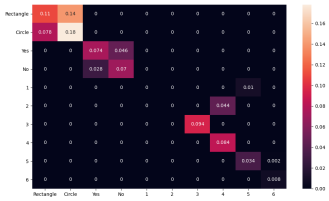

Fig. 4. The confusion matrix of the 6x6 dataset on a purely Neural Network based model

It is clear that the network trained with DeepProbLog is able to learn way faster then the purely NN based approach, if measured in the number of iterations [6]. However, the differences in training time are much less significant, due to the high computational cost associated with the arithmetic parts of the circuit (i.e. the semirings).

VI. CONCLUSIONS

VII. APPENDIX

A. Logical rule encodings

REFERENCES

[1] R. Manhaeve, A. Kimmig, S. Dumančić, T. Demeester, L. De Raedt, "Deepproblog: Neural probabilistic logic programming", *in Advances in Neural Information Processing Systems*, vol. 2018-Decem, pp. 3749–3759, jul 2018, doi:10.48550/arxiv.1907.08194, URL: `https://arxiv.org/abs/1907.08194v2`, `1805.10872`.

[2] J. Johnson, L. Fei-Fei, B. Hariharan, C. L. Zitnick, L. Van Der Maaten, R. Girshick, "CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning", *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1988–1997, 2017, doi: 10.1109/CVPR.2017.215, `1612.06890`.

[3] K. Heecheol, "kimhc6028/relational-networks: Pytorch implementation of "A simple neural network module for relational reasoning" (Relational Networks)", URL: `https://github.com/kimhc6028/relational-networks`.

[4] R. Manhaeve, "ML-KULeuven/deepproblog: DeepProbLog is an extension of ProbLog that integrates Probabilistic Logic Programming with deep learning by introducing the neural predicate.", URL: `https://github.com/ML-KULeuven/deepproblog`.

[6] By the number of iterations, the number of training steps on a batch is meant.