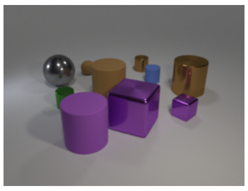
Visual Question Answering Using Differentiable Forth Interpreter

I. Introduction

In deep learning, visual question answering (VQA) involves answering questions about visual data. It requires recognizing objects in the provided image and reasoning to answer questions about the visual scenery, as illustrated in Figure 1.

In previous works on neuro-symbolic methods, Jacob A [1] used a module network with a question parser to manually specify the layout of the structure of the modules for an adaptive network, and they used an attention mechanism as the main component of each module. Jacob A [2] went further by learning the parser jointly with the module structure. These models use hand-engineered modules. However, Justin J [3] used a universal module architecture and an LSTM as a program generator to generate module structures for more complex questions. Instead of using the image as an input to the module network, Kexin Y [4] generated a structural scene representation from the image and executed the generated program on the scene, which offered interpretability.

In this work, we used the $\partial 4$ [5] to learn to choose what operation to execute and how to execute it, for questions involving operations, such as $>, <,$ and $=$, using CNN and RNN feature vectors, and using the CLEVR dataset [6]. The $\partial 4$ interpreter makes it possible to write a program sketch with slots that can be learned end-to-end in a differentiable way.


Q: Are there more small purple things on the left side of the large ball than big brown matte objects?
Figure 1. CLEVR dataset sample

II. Model Description

Our model connects the $\partial 4$ interpreter to a CNN+LSTM baseline, and learns them end-to-end.

The CNN+LSTM model uses CNN to extract the feature vector of the image, and the LSTM reads the text of the question and produces another feature vector. Features are first extracted using Resnet101 trained on ImageNet. Features are then transitioned to the CNN, consisting of a convolutional layer with 400 1x1 kernels each, ReLU activation and 4 max-pooling layers. The dimension of the LSTM is 256, preceded by word embedding with dimension of 300. The resulting feature vectors are used to initialize the heap of $\partial 4$.

Listing 1 shows the sketch used to solve the VQA problem. First, it copied the values of the question and feature vectors from the heap to the return stack. Second, it included three slots; the first two slots were used to choose the number of objects in the image that the question mentioned. The third slot was used to choose the operation (>, <, and =), these slots were learned using the vectors in the return stack. Finally, we emptied out the return stack.

The first implementation steps included rendering CLEVR images and generating questions. The initial CNN features were extracted using Resne101, and then the questions were tokenized.

```
\ choose first number of objects
{ observe R0 R-1 -> choose 1 2 3 4 5 6 7 8 9 10 }
\ choose second number of objects
{ observe R0 R-1 -> choose 1 2 3 4 5 6 7 8 9 10 }
\ choose the operation
{ observe R0 R-1 -> choose  = > < }
```

Listing 1. VQA sketch main lines

The CLEVR dataset was divided into 300 samples for training and 100 samples for the development set using 34 images. The batch size was set to 50, and the learning rate was 0.02. The CNN output vector length was set to 400, and the LSTM was set to 256. The width of the stack was set to 300, and the stack size was set to 5.

III. Results and Discussion

Our epoch number was set to 10. Our results showed the same accuracy for each epoch, which was 11%. This was the result of the first run of the program without any further optimization. However, we do not suspect that there is a problem with the $\partial 4$, nor with our implementation. In other words, to judge the $\partial 4$ precisely, we needed to optimize our model parameters first.

IV. References

[1] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, “Neural module networks,” *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.*, vol. 2016-Decem, pp. 39–48, 2016, doi: 10.1109/CVPR.2016.12.

[2] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein, “Learning to compose neural networks for question answering,” *2016 Conf. North Am. Chapter Assoc. Comput. Linguist. Hum. Lang. Technol. NAACL HLT 2016 - Proc. Conf.*, pp. 1545–1554, 2016, doi: 10.18653/v1/n16-1181.

[3] J. Johnson *et al.*, “Inferring and Executing Programs for Visual Reasoning,” *Proc. IEEE Int. Conf. Comput. Vis.*, vol. 2017-Octob, pp. 3008–3017, 2017, doi: 10.1109/ICCV.2017.325.

[4] K. Yi, A. Torralba, J. Wu, P. Kohli, C. Gan, and J. B. Tenenbaum, “Neural-symbolic VQA: Disentangling reasoning from vision and language understanding,” *Adv. Neural Inf. Process. Syst.*, vol. 2018-Decem, no. NeurIPS, pp. 1031–1042, 2018.

[5] M. Bošnjak, T. Rocktäschel, J. Naradowsky, and S. Riedel, “Programming with a differentiable forth interpreter,” *34th Int. Conf. Mach. Learn. ICML 2017*, vol. 2, pp. 842–859, 2017.

[6] J. Johnson, L. Fei-Fei, B. Hariharan, C. L. Zitnick, L. Van Der Maaten, and R. Girshick, “CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning,” *Proc. - 30th IEEE Conf. Comput. Vis. Pattern Recognition, CVPR 2017*, vol. 2017-Janua, pp. 1988–1997, 2017, doi: 10.1109/CVPR.2017.215.