REPRODUCIBILITY CHALLENGE - THE NEURO-SYMBOLIC CONCEPT LEARNER

Thanakorn Panyapiang
tp2n19@soton.ac.uk
ID: 31446612

Vasin Srisupavanich
vs2n19@soton.ac.uk
ID: 31300162

Peng Chen
pc3n19@soton.ac.uk
ID: 31142354

1 INTRODUCTION

Mao et al. (2019) proposed Neuro-Symbolic Concept Learner, inspired by human learning process, to learn visual concepts and accurately answer questions on CLEVR dataset. Their model achieved remarkable accuracy in visual concept learning, question answering, and generalised well to images with more objects and more complex questions. As shown in Figure. 1, the model is composed of three parts: a visual module to extract features from the scene, a semantic parser to translate questions into programs, and a symbolic reasoning module to execute the program to obtain an answer and bridge the learning between a visual module and a semantic parser.

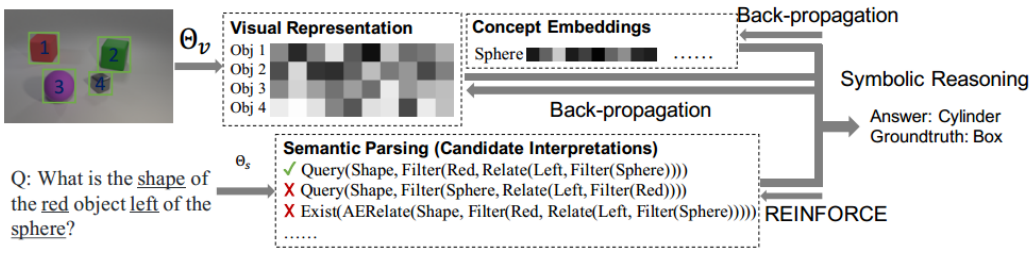

Figure 1: NSCL architecture [Figure from Mao et al. (2019)]

Visual Module. The visual module are responsible for two tasks: feature extraction and concept quantization. Given an input image, the model first extracted region-based and image-based features which contains the information of objects' properties and relations between objects respectively. Then the features are concatenated and mappped to embedding space to compare with concept embeddings (e.g. Red, Green) by measuring the cosine similarity.

Semantic Parser. The semantic parser aims to translating questions into a series of programs which are in a domain specific language (DSL). Each program represents an operation that has its own input and output type, such as that `Filter(Red)` means selecting all the red objects and it inputs the objects in the scene and concept 'Red' and return red objects.

Symbolic Program Executor. The program executor combines the objects' representations and programs sequences to derive the answer. The executor is actually a collection of functions of all the programs. To apply gradient-based optimisation, the outputs of the intermediate program are in a probability form. For example, if we have `Filter(Red)`, the executor will output the probability of all the objects in the scene being red.

The training strategy is also motivated by human learning. The author employed **curriculum learning** method to help optimisation. The curriculum learning is consisted of the following four steps.
Lesson 1: Only learning object-level visual concepts and trained on four kinds of simple questions
Lesson 2: Learning relational-level questions and added a new type of question for generation
Lesson 3: Learning more complex questions on full CLEVR with visual modules fixed
Lesson 4: Joint fine-tuning of all three modules

2 METHOD

For this reproducibility challenge, we have implemented the visual module, concept embedding, and program executor from scratch based on the original paper. However, we assumed that the semantic parser and the Mask R-CNN module, which is responsible for generating object proposals are pre-trained. Our implementation are available in `https://github.com/COMP6248-Reproducibility-Challenge/nscl_reproducability_challenge`

2.1 VISUAL MODULE

Feature Extraction. The input images which contains the objects' location produced by masked R-CNN are sent into a ResNet-34 to extract the features. Then we used RoI Align to locate the area of interest on the those features and extract region-based representation which contains the information of each object. To get the image-based representation, we passed the ResNet-34's output to a convolutional layer that has the same size as the features for linear combination of channels and then applied RoI Align with the box size equals to the down-sampled image. Finally, the region-based and image-based representation are concatenated to output a feature that represents an object.

Concept Embedding. For each object in a scene, the extracted features are linearly mapped to an attribute embedding space by an attribute operator. Each attribute (e.g. `Shape`) has its own space. Then the objects' embeddings and concept embeddings(e.g. `Red`) are compared using cosine distance to determine the object property.

2.2 REASONING MODULE

Program Executor. The executor resolves answer of a question's program sequence. Since our scope is limited to object-level attributes, we only implement four programs in the DSL: `Filter`, `Query`, `Count`, and `Exist`, aiming to answer four kinds of questions (i.e.query the attribute, query the existence, count the objects and query objects have same attribute or not). The result of the last program will be returned as a final answer.

2.3 TRAINING

Curriculum Learning. As mentioned in section 1, the model is trained using only simple questions in the early stage of curriculum learning. Hence, for each lesson, the dataset is filtered by the maximum length of the program sequence and the maximum number of objects in a scene based on the curriculum strategy. In addition, we also generated additional synthetic questions to help with the learning process.

Loss. Loss functions are selected dynamically according to the question type. In essence, we considered `query` and `exist` questions as classification problems so we choose cross-entropy, `count` questions are considered regression problems and measured using MSE.

3 EXPERIMENTS

We have chosen to replicate three experiments on the CLEVR dataset: classification-based concept evaluation, count-based concept evaluation, and data-efficiency evaluation. Note that the scope of our work only cover object attribute questions, therefore all the questions containing relational attributes are excluded. We trained the model using only the training data which satisfied lesson 1 strategy, and generated additional 20 questions for each scene using the scene annotation file.

3.1 IMPLEMENTATION DETAILS

Our model are implemented using PyTorch. For the experiments, we used the bounding box annotations pre-generated from Mask R-CNN module provided from the official implementation. The specific details on training the model are shown in Table 1.

Table 1: Training details and hyperparameters

Property	Values	Remark
Batch size	64	-
Epochs	10	15 in final step of lesson 3
Learning Rate	0.01	-
Optimiser	Adam	-
Curriculum Strategy	8-25; 4-10	max program depth; max number of objects in a scene

3.2 RESULTS

Classification-based concept evaluation. The objective of this experiment is to evaluate whether the model learns generic concept representation. The point is that accurate and generic representation of object properties would then be useful for various applications, including image captioning and retrieval. In this experiment, we evaluated the classification accuracy for all attributes (colour, material, shape, size) using the validation set. In the original paper, they reported 99% accuracy for all object properties, however we were able to achieve around 99% accuracy for all attributes except the shape attribute which is around 81.5% accuracy. Looking further, we found that our model tends to misclassify cylinder object to be a cube. We suspected that we might have missed some scene augmentation processes not reported in the original paper.

Count-based concept evaluation. Next, we performed the experiment to evaluate the accuracy of the count concept, which has been known to be difficult in deep learning due to the discrete nature of the problem. For this experiment, we evaluated our model on the validation set by generating 5 synthetic counting questions for each image. The questions generated are of the following form: "How many `Cube` objects are there?". As shown in table 2, the accuracy for all attributes except shape are quite close to what is reported in the original paper. We believed that we are facing the same issue as stated in the previous experiment.

Table 2: Accuracy of count concept (%)

	Mean	Colour	Material	Shape	Size
Original paper	98.7	99.0	98.7	98.1	99.1
Our implementation	92.7	98.4	97.4	70.1	99.1

Data-efficiency evaluation To verify the data efficiency, we trained the model using 10% of training images and compare against the model trained with the whole dataset. Despite the correctness is substantially lower than the original work, the accuracy difference between the model trained with 7k images and 70k images is minor. The former has an accuracy of 74.6% while the latter got 75.9% of questions correct on object-attribute questions in the validation set which is the similar pattern as the original.

3.3 REPRODUCIBILITY COST

All of the models used in this section are trained on Google Colab. The server has 2-core Intel(R) Xeon(R) CPU @ 2.30GHz, Tesla K80 GPU accelerator, 13GB RAM and 64 GB HDD. Training models takes 12 hours and 4 hours for the model using all images and 10% of images respectively.

4 ANALYSIS AND DISCUSSION OF FINDINGS

4.1 DIFFICULTY IN EVALUATING THE RESULT DUE TO SYNTHETIC DATA

The original paper claims that by using 5K images from the CLEVR dataset NS-CL is able to outperform other state-of-the-art models. However, the statement is difficult to evaluate as the questions, unlike the images, are synthetically generated by the authors themselves. The questions play a key role in the training because the model learns different concepts from different questions. For example, answering the question "What is the color of the cube?" makes the model learn about `Cube` and

all color concepts, while the model is required to learn only `Rubber` to answer the question "How many rubber objects are there?".

Since the detail regarding the question generation is not provided, we need to create our own approach. However, we are unable to reproduce the same accuracy. Without the clear specification of how the questions are generated, it is difficult to identify whether the issue comes from the model structure, the implementation, or the question generation method.

4.2 Importance of curriculum learning

We found that to make the model training converges, we needed to setup proper curriculum learning strategies. This curriculum learning process is required since the model only use the supervisory signal from answer's correctness to learn. However, the strategies are not clearly stated in the paper. For example, the number of epoch to train and how to increase the number of objects in each lesson are not mentioned. In addition, we found that using only the official CLEVR training data is not enough, because when we filter the dataset based on lesson 1 strategy, the data size became very small. These issues has posed a lot of difficulty when we were trying to replicate the experiments, and the difference in learning strategies could be another factor which affected the overall accuracy.

4.3 Augmentations of visual features

As stated in section 3.2, we suspected that one of the reasons that we were unable to reproduce the result could be related to the implementation of the visual module. Some details which are required to develop the visual module were missing from the paper. For instance, in the paper, they did not explicitly state how the concatenation of object features and image features are done. In addition, we found that there were additional processing steps of visual features in the official implementation. These includes the use of 1x1 convolutional layer with the same input and output channels, as well as, the reduction of feature's values in the region where objects are overlapped. Because the official implementation is based on author's personal library, there were some parts which were not very clear. Therefore, we were not able to reimplement all the additional processing steps of visual features, which could have impacted the accuracy.

4.4 Ambiguity in loss measurements

The original paper proposes KL divergence as the loss function of the visual module. Considering that questions in CLEVR dataset are either classification or regression problems, this choice is uncommon and the authors did not provide the reason to justify their decision. Moreover, we found that the model learns better using simple functions namely Cross-Entropy and MSE.

5 Conclusion

The data efficiency experiments show the similar pattern as the original work. However, we are unable to achieve the accuracy claimed by the paper on attribute classification and count-based questions.

We hypothesize that the discrepancies in the results come from two main causes: implementation of the visual module and question generation strategy. Regarding the visual module, although the paper gives sufficient explanations about the overall architecture, details such as the structure of the attribute operators and concept embeddings are not mentioned. Implementation difference of these components could impact model performance. Also, the fact that the training process involves using synthetic questions could be another key factor in the issue as we have pointed out the importance of questions toward the learning of the model in Section 4.1.

References

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.