THE COMP6248 REPRODUCIBILITY CHALLENGE: INTERPRETABLE COUNTING FOR VISUAL QUESTION ANSWERING

Qi Zhao
Student ID: 30571626
Email:qz1y18@soton.ac.uk

Xin Zhang
Student ID: 29896665
Email:xz5m18@soton.ac.uk

ABSTRACT

This work is a part of COMP6248 Reproducibility Challenge. Our goal is replicating the experiments described in the conference submission by Trott et al. (2018): *Interpretable Counting for Visual Question Anwersing.* This paper propose a new method to count in visual question answering, which increases accuracy. In addition, authors built a subset named How-Many-QA to filter questions which are not suitable to the method. In the details of reproduction, we have some places that are different from the original paper, including decrease the size of dataset, the structure of language model and the types of counting models. The code of our work is available at `https://github.com/COMP6248-Reproducability-Challenge/Interpretable-Counting-for-Visual-Question-Answering`

1 INTRODUCTION

Visual question answering (VQA) is an important benchmark for testing context-specific reasoning of complex images. Although substantial progress has been made in this area, the improvement in counting-based problems is minimal. The authors proposed a effective counting model which can process counting question more accurate. The detailed information will be introduced as follows.

2 DATASETS

The author of original paper defined a subset named HowMany-QA, which is based on VQA 2.0 and Visual Genome dataset. The subset aims to filter QA pairs that the answer is a number. About the subset, there are two filter rules. One is that it requires the question need to contain phrases similar as How many, number of or count of, while phrases such as number of the or the age of . The other rule is that the value of answer should be between 0 to 20. Some examples in HowMany-QA are shown in Figure 1. HowMany-QA is available at `https://einstein.ai/research/interpretable-counting-for-visual-question-answering/HowMany-QA.zip`.

3 MODEL

3.1 OBJECT DETECTION

In this section, I mainly used the method of Anderson et al. (2017) and Teney et al. (2016) to detect the objects. They model represents the current technical level in VQA, which regard deducing object as the input of question-answer section. This section used Faster R-CNN architecture. When gave a picture, the Faster R-CNN is able to output a serious of areas that include objects in the pictures. During the Faster R-CNN working, there are four main steps. Firstly, input the target image into CNN to get feature maps. Then, inputting the convolution feature to the RPN to obtain the feature information of the candidate frame. Thirdly, as for the features extracted in the candidate box, using the classifier to determine whether it belongs to a specific class. At last, for a candidate box

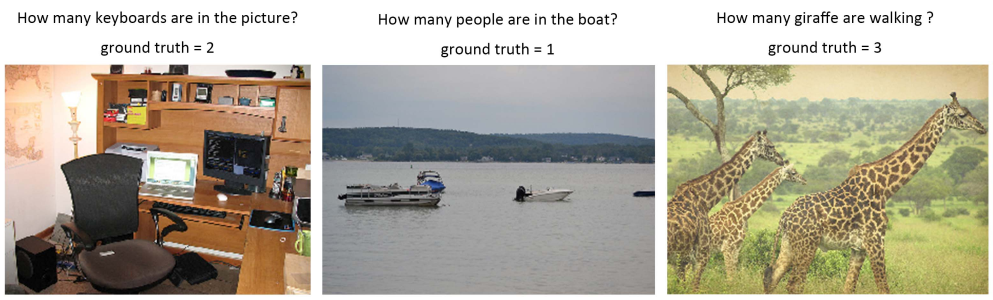
How many keyboards are in the picture?
ground truth = 2
How many people are in the boat?
ground truth = 1
How many giraffe are walking ?
ground truth = 3

Figure 1: Examples of HowMany-QA

belonging to a category, using a regression to further adjust its position. The output of Faster R-CNN are a set of bounding boxes$\{b_1, \ldots, b_n\}$, and a set of object encodings$\{e_1, \ldots, e_n\}$ which means the relationship between boxes and target objects. In addition, the model is trained based on the already trained model to save costs. About this section, There is a great help for me in the repository `https://github.com/hengyuan-hu/bottom-up-attention-vqa`, which is a reproducibilty of Anderson et al. (2017) and Teney et al. (2016).

3.2 Language Process

In this section, through a scoring function, questions will be encoded and compared with every target objects, to get a score vector which means the relevance between objects and questions. The details are shown in the equations below:

$$h^t = \mathbf{GRU}(x^t, h^{t-1}) \tag{1}$$

$$q = h^T \tag{2}$$

$$s_i = f^S([q, v_i]) \tag{3}$$

In the equation(1), x_t means the word embedding of question token at position t. q is the final hidden state of a GRU after processing the question an dcompute a score vector for every target objects. s_i means the score of relevance between objects and questions. And about the function $f^S : \mathbb{R}^m \to \mathbb{R}^n$, it is a 2-layer Leaky ReLu based network.

In this section, there are two difference between our work and authors' work. One is that we used GRU in equation (1), while authors used LSTM. The other difference is that we used a 2-layer Leaky ReLu based network while the authors used a layer of Gated Tanh Unit.

3.3 Counting

In this section, the authors implement 3 models and compare their performance to prove the model they created is the best one. In order to simplify the paper, we only use 2 models to compare. They are Interpretable RL Counte(IRLC) and SoftCount. The details of them will be introduced as follows.

Interpretable RL Counte(IRLC). This model is proposed by the authors, where is the most interesting place in the paper. It is a new way to count. It uses an iterative sequential decision process for counting the target objects. There are two mainly novel point to count subjects. The first one is that at every time step, the model will make discrete choices about the objects need to count. The second one is that after predicting answers, the model will return bounding boxes for the target objects.

About this section, firstly, we need to represent the probability of chosing a given action and how every actions will affect choices. In order to do so, we put the scores into a logits vector through equation (4), which means how likely every object will be chosed to be counted.

$$u = Ws + b \tag{4}$$

Then we will compute a matrix of interaction terms p to update the logits u. About the matrix, p_{ij} means the level how target objects will influence u_j. About the matrix, we used equation (5) to get it.

$$p_{ij} = f^p(Wq, {v_i}^T v_j, b_i, b_j, IoU_{i,j}, O_{i,j}, O_{j,i}) \tag{5}$$

In the equation, f^p is a 2-layer MLP with ReLU activations. Wq is the compressed representation of questions. ${v_i}^T v_j$ is the dot product of the normalized object vectors. b_i and b_j is the object coordinates $.IoU_{i,j}, O_{i,j}, O_{j,i}$ are basic overlap statistics.

In every steps t of the counting squence, we just select the action whcih has the highest value, and use the equation (7) to update u.

$$a^t = argmax_i[s^t, r] \tag{6}$$

$$k^{t+1} = k^t + p(a^t, :) \tag{7}$$

In the equations above, r represents the logit value of terminal action, which is a scalar. The action a^t is expressed as the index of the selected object.$p(a^t, :)$ means the row of p indexed by a^t.

During the process, every object should only be counted by once. We define the count C as the time step when the final action was chosed t: $a^t = N + 1$

When training the model, the optimizer we used is Adam and set learning rate as 0.0005 which is followed the authors. In addition, we decay the learning rate by 0.99999 every iteration. In addition, I set early stop which is based on the development set accuracy.

SoftCount. As a baseline method, we train the model directly from the outputs s of scoring function. For each object, we make its score vector s_i to a scalar value and use sigmoid function to give the object a count value between 0 to 1. The total count is the sum of these values. When training the model, we use Huber loss associated with the absolute difference e between the predicted count C and the ground truth count C^{GT}.

$$C = \sum_i sigmoid(Ws_i) \tag{8}$$

$$L_i = \begin{cases} 0.5e^2 & if \quad e \leq 1 \\ e - 0.5 & otherwise \end{cases} \tag{9}$$

When evaluating the model, we round the estimated count C to the nearest integer and limit the output to the maximum ground truth count.

4 Result and Analysis

Considering the limitation of memory of Colab, I decrease the amount of dataset to guarantee the operation of the program. Because of this action, it is conceivable that the accuracy of models will decrease. The Table 1 shows the training result of IRLC and SoftCount.

Model	**Accuracy**	**RMSE**
IRLC	0.333	3.194
SoftCount	0.346	3.245

Table 1: Sample table title

Through Table 1 we can see the accuracy of both models are lower than authors'. In my opinion, there are two main reasons. Firstly, we only used the data in VQA 2.0 but not used data in VG. In this situation, authors proved that the accuracy that VQA2.0 and VG both are used will be higher than the accuracy that only VQA 2.0 is used. The second reason is that because of the limitation of memory, I decrease that size of trainingset to guarantee the operation of the program, which may lead to the accuracy decreasing. In Tabel 1,the RMSE of IRLC is better than that of SoftCount, which is as same as authors' result. But the Accuracy of IRLC is worse than that of SoftCount, which is unexpected.

The process of IRLC answering questions are shown in Figure 2.The Blue Box means pending object, the red box means the confirm object which will be counted. More pictures are shown in my coursework code.

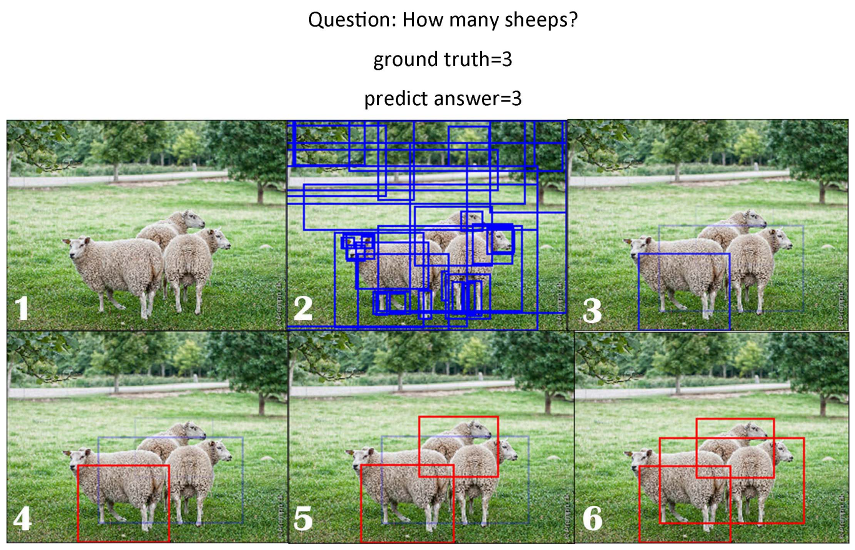

Figure 2: Examples of HowMany-QA

5 Discussion

As I mentioned above, our work reproduce most of the original paper. We implement the process of IRLC answering a question and show it above. But we failed to prove that the accuray of IRLC is higher than accuracy of SoftCount. There are mainly four difference between our work and authors' work. First, about dataset, we only used VQA 2.0 rather than VQA 2.0 + VG. In addition, in the section of language process, we used GRU to replace LSTM and used a 2-layer LeakyReLu based network to replace a layer of Gated Tanh Unit to generate score which means the relevance between question and objects. At last, we greatly decrease most of the training data to solve the limitation of computer's memory. If it is possible, we will try to cancel some simplify details to reproduce the method in future.

6 Conclusion

The method is feasible to implement. Although our work do not get the same result as authors, I think if we use a more powerful computer to use the a same size training set, we can get a same result.

Acknowledgments

Github repository `https://github.com/sanyam5/irlc-vqa-counting` and `https://github.com/hengyuan-hu/bottom-up-attention-vqa` greatly helped our work. Their codes are useful to reproduce the paper.

References

Peter. Anderson, Xiaodong. He, Damien. Buehler, Chris. Teney, Mark. Johnson, Mark. Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and VQA. arXiv, 2017.

Damien. Teney, Lingqiao. Liu, and Anton van den Henge. Graph-structured representations for visual question answering. arXiv, 2016.

Alexander. Trott, Caiming. Xiong, and Richard Socher. Interpretable counting for visual question answering. ICLR, 2018.