REPRODUCIBILITY REPORT FOR: LEARNING TO COUNT OBJECTS IN NATURAL IMAGES FOR VISUAL QUESTION ANSWERING

Lingeshwaran Kanniappan, Subhasish Ghosh,
Wenhao Wang, Sai Nayan Dhoria
Department of Electronics and Computer Science
University of Southampton
{lk3u18, sg1g18, ww2g18, snd1n18}@soton.ac.uk

ABSTRACT

This report presents a reproduction of the ICLR accepted paper "Learning to Count Objects in Natural Images for Visual Question Answering". The main objective of the report is to reproduce the results of the original paper, highlight any deficiencies and present the code in a modular scalable format using the torchbearer library [1].

1 PAPER OVERVIEW

The authors in this paper discusses on the fact that the valuable loss of information is significantly due to duplicate objects and soft attention mechanism. As a result, a novel approach to modify the neural network to learn the presence of the duplicate objects is made using differentiable bounding boxes which can be made to learn the presence of duplicate objects in the given image. The objects in the image are interpreted as elements of a graph and using basic graph theory a differentiable counting method is implemented.

Briefly, the paper points out two fundamental issues due to which VQA architectures perform poorly in the counting task:

- **Soft-attention mechanism:** The weighted average of soft-attention to answer the question "is there a cat" is not suitable to answer the question "how many cats". Soft attention averages out the features corresponding to different instances of the objects and thus loses instance information.
- **Ground truth labelling:** Unlike standard counting task, for VQA, there is no ground truth labelling of the object to be counted.

To circumvent these issues, the paper proposes the following:

- Use raw attention maps (instead of processed counting features) and perform extra operations such as de-duplication etc. to output counting features more suitable for counting objects.
- Create a separate counting class to make the implementation plug-and-play.
- Ensures that the counting class is implemented in such a way that the original VQA model is not affected.

2 PAPER CONTRIBUTION

The counting class is the main contribution of this paper. Other than this, a baseline model (Kazemi & Elqursh (2017)), was modified to integrate this class. In this section we shall discuss the counting class, changes to the baseline model and the piecewise linear functions.

2.1 COUNTING CLASS

To deal with the problem of overlapping proposals, the scene is converted into a graph with the edges indicating the weights of the activation functions that are learnt by a differentiable neural network component. The object proposals remain the key to the improvement in accuracy and is

[1] Our implementation is available at https://github.com/COMP6248-Reproducability-Challenge/LearningToCountObjectsInVQA

discussed by Anderson et al. (2017). Steps involved in the Count module:

1. Calculate the attention weights (a) and generate the attention matrix A by performing the outer product between a and a^T.
2. Eliminate the inter-object edges and the intra-object edges.
3. Intra object edges are removed by computing the distance matrix D and attention matrix A using the equation $f_1(A) \odot f_2(D)$. The distance matrix is given by the formula
$$D_{ij} = \left(1 - IoU\left(b_i, b_j\right)\right) \tag{1}$$
4. The IoU matrix corresponds to the Intersection-over-Union and the b_i denotes the bounding box of the ith object proposal.
5. The Inter object edges are removed using the count from the object proposals that are duplicates of the same object. The edges are scaled down to the different values of the instances of the same object proposal. The effect of inter object edge de-duplication will reduce the number of multiple occurrences of the same object. This can be done using the Similarity index S_i given by:
$$s_i = \frac{1}{\sum_j \mathrm{sim}_{i,j}} \tag{2}$$

The final count is calculated using the Vertices (V) and the edges(E) of the subgraph using the relation $V = \sqrt{(E)}$. This can be further simplified to an outer product of the similarity index and this is done to avoid the bidirectional nature of the graph edges. The self-loops in the graph is removed while computing the value for the $\tilde{A}$. And using the scaling factor *s,* the formula for the final count is as follows:
$$C = \tilde{A} \odot ss^T + diag\left(s \odot f_1(a \odot a)\right) \tag{3}$$
This count is the final graph for the number of object proposals. The actual count can be obtained using the sum over all the values in the graph.

The count module finally evaluates the confidence in the prediction by defining two variables p_a and p_D which compute the average distance of the piecewise linear function $f_6(a)$ and $f_7(D)$ from 0.5. The final output *o* is given by $f_8(p_a + p_D)$. o

The piecewise linear functions are special functions that are monotonically increasing and lie between *0* and *1*. They are defined in such a way that 16 weights are learnt via backpropagation. This can be further implementer using 8 and 32 weights respectively.

2.2 Baseline Architecture Modifications

The model is developed from the work of (Kazemi & Elqursh, 2017). Their model was able to outperform most of the basic VQA models at that time. The authors have worked on modifying the baseline by utilizing the counting component. The modification includes the fusion mechanism for the features of the images x and the features from the question y. This is done using the concatenation operation augmented with the linear projection and a ReLU layer. The new fusion operation is defined as follows:
$$x \diamond y = ReLU\left(W_x x + W_y y\right) - \left(W_x x - W_y y\right)^2 \tag{4}$$
The LSTM used for the encoding of the question is replaced with a GRU (Cho et al., 2014) with the same hidden size with dynamic per-example unrolling instead of a constant word length question. The model utilizes the effect of batch normalization and dropout at the same time. This was not experimented from our side as the computational cost in executing the VQA v2 dataset was exhaustive.

2.3 Piecewise Linear Functions

The piecewise linear functions are used for the differentiable learning of the weights required for the learning of the model. They are structured in a way that it is split into d equal size intervals in [0,1]. Each function contains a line segment that is connected to the neighboring line segments at the boundaries of the intervals. The line segments form the overall architecture of the activation function. For each activation function f_k, there are d weights $w_{k1}, w_{k2}, \dots, w_{kd}$ where the weight w_{ki} is the gradient in the interval $\left[\frac{i-1}{d}, \frac{i}{d}\right)$. The author has used an arbitrary value of 16 for d. We have implemented the toy task using the values of 8 and 32. All the weights are assumed to be non-negative and thus will flow through the absolute mechanism in all cases. The function is written as follows:

$$f_k(x) = \sum_{i=1}^{d} max(0,1 - |dx - i|) \frac{\sum_{j=1}^{i}|w_{kj}|}{\sum_{m=1}^{d}|w_{km}|} \quad (5)$$

The division of the weights by the value $\sum_m^d |w_{km}|$ will results in the normalization and this approach of using normalized cumulative sums is like the sub-gradient approach followed by Jaderberg et al (2015). To make sampling from indices applying the function $f_k(x)$. All weights are initialized to 1 and this makes the function linear initially.

3 Re-Implementation Approach

For the re-implementation, we partially adapted/re-implemented the models and classes from the original paper source code. However, we used Torchbearer model training library (Harris et al., 2018) to train and test. The motivation was to enforce evaluating the counter module using a standardized training approach. Further, we also implemented metrics collection and plotting mechanisms independently.

With this approach, we treated the counter module as a black box and re-implemented all external instrumentation such that no external biases other than what is claimed in the paper could influence the results. However, due to computational resource limitations, we were only able to reproduce the results from the Toy dataset. The reproducibility of the VQA dataset was already previously extrapolated (Sodhani & Pahuja, 2018).

4 Results

The results for the tasks reproduced included the accuracy plots for the baseline and the Counter Module as shown in Fig. 2. The accuracy plot using the counter module was reproduced as generated by the author, however the performance baseline model on the data did not seem to be similar. This could be because the original paper did not fix any random seed for exact reproducibility. We did not investigate this further. The additional experiment executed from our side was setting the number of weights in the activation functions $d = 16$ & 32. This can be seen in Fig. 3 respectively. This shows similar performance as hypothesized in the original paper. The activation plots for the "easy" and "hard" parameters are seen in Fig 4. The plots generated similar results.

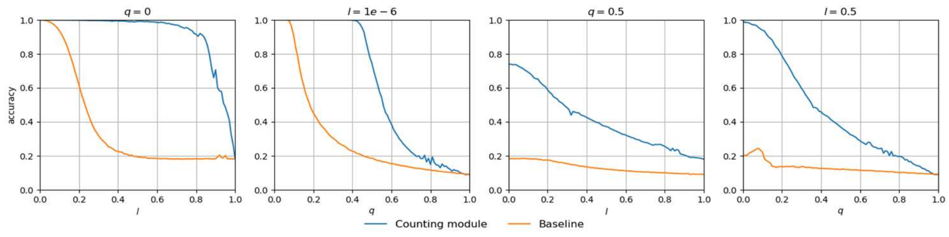

Figure 2. Baseline and Counter Module Accuracy Plot

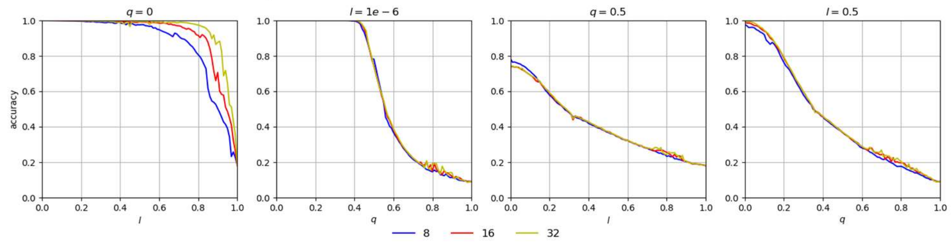

Figure 3. Accuracy Plots for activation functions with weights 8, 16 and 32 for the Counter Module

5 Observations

- The training time required for each iteration was around 4.5 mins. The resolution was set to 100. Thus, for all 'l' and 'q' values for 'easy' and 'hard' task, the training time was approximately 30hrs. Then for the counter and baseline models, the training time doubled to more than 60hrs sequentially.

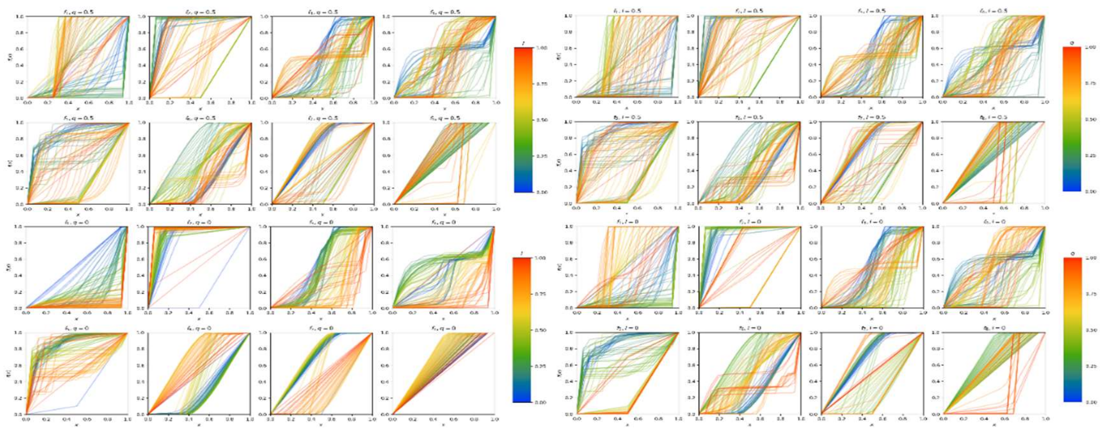
Figure 4. Shape of Activation Functions as q and l are varied on the toy dataset. Each line shows the shape of the activation function with the weight set to 16 and the value associated to its color.

- The paper did not explicitly state if a validation dataset was used. We tried using a validation set and observed that the result was similar. However, this was further doubling the training time. Due to computational resource limitations the results presented here do not include a validation dataset.
- We observed that most of the computation was occurring on the CPU, however upon investigation, we found that this was due to the ToyTask generating the dataset.
- We compared the GPU performance of the Counter module with the Baseline and found that it consumed only about 2% of GPU resources for a Nvidia 1050 ti GPU.
- All hyper-parameters and equations required to reproduce the results were available in the original paper.
- The author simplified their experiments by considering two extremes of bounding box overlap, they either fully overlap or do not overlap. It is not clear if this can be extended to real scenario.

6 Conclusion

The main intention behind this paper was to verify and validate the experimental results of the original paper. The paper explicitly provides the necessary details and is unequivocal to a great extent. Although we could not exactly reproduce the results due to limitation we have in terms of computational resources, we were able to replicate the work using toy dataset, the results of which proved to be consistent and experiential. Our evaluation proves that the model is largely efficient and gives notable gains in performance related to count-based questions when compared to the baseline model.

7 Acknowledgements

We would like to show our gratitude to Prof. Jonathon Hare, Prof. Adam Prugel Bennett and Yan Zhang for their valuable and constructive suggestions throughout numerous consultations and for allowing us to reproduce their work. Special mention to Ethan Harris for the usage of torchbearer in our work.

References

KyungHyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014. URL `http://arxiv.org/abs/1409.1259`.

Ethan Harris, Matthew Painter, and Jonathon Hare. Torchbearer: A model fitting library for pytorch. *arXiv preprint arXiv:1809.03363*, 2018.

Allan Jabri, Armand Joulin, and Laurens van der Maaten. Revisiting visual question answering baselines. *CoRR*, abs/1606.08390, 2016. URL `http://arxiv.org/abs/1606.08390`.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. *CoRR*, abs/1506.02025, 2015. URL `http://arxiv.org/abs/1506.02025`.

Vahid Kazemi and Ali Elqursh. Show, ask, attend, and answer: A strong baseline for visual question answering. *CoRR*, abs/1704.03162, 2017. URL `http://arxiv.org/abs/1704.03162`.

Shagun Sodhani and Vardaan Pahuja. Reproducibility report for" learning to count objects in natural images for visual question answering". *CoRR*, abs/1805.08174, 2018. URL `http://arxiv. org/abs/1805.08174`.