Differentiable Adaptive Computation Time for Visual Reasoning

Cristóbal Eyzaguirre and Álvaro Soto
Pontificia Universidad Católica de Chile
ceyzaguirre4@uc.cl, asoto@ing.puc.cl

1. Supplementary Material

1.1. Implementation details

Unless otherwise stated, all models use the default parameters from the official implementation of MAC [1]. Notable differences between these and those in the original paper [3] include using different dropout rates for different sections of the network (92% on output of biLSTM, 82% on convolutional filters), initializing weights by sampling from the Xavier uniform distribution [2], setting the question as the first control, and using a LR scheduler. We use a gate bias of 1.0 for the gated MAC as this achieves the best performance in the original paper; however, unless stated otherwise, we do not bias our adaptive computation algorithms.

Experimental results in Section 4 are obtained by training from scratch the MAC network three times and then using the resulting weights to train multiple other models. Each of the pre-trained MACs are further trained to obtain: four variants of DACT, corresponding to different values of λ; and seven variants of ACT, six with different λ's and one without ponder penalty. Training the variants is done by reseting all schedulers and optimizers and then reinitializing a new MAC with its corresponding gating mechanism. We then overwrite the pertinent weights with the pre-trained ones and train for another 30 epochs saving the best model according to the monitored precision. This is done to reduce the time needed to train all models while still providing significant mean and variance values for relevant metrics (*i.e.* precision and steps). Figure 1 shows how the average number of steps used by DACT-MACs for CLEVR [5] adapts from the original 12 step algorithm learnt during the pre-training phase of the 12-step non-adaptive MACs, progressively reducing the total computation needed by iterating fewer times.

Few changes were needed in order to apply adaptive MACs to GQA [4]. Instead of using regions of the image (top-down attention), our knowledge-base is composed of region detections (bottom-up) [1]. We also use 1x1 convolution filters instead of 3x3 for the same reason. Additionally, we use a 4 step non-adaptive MAC during the pre-training phase as this is the recommended number of steps for the model [4] and also yielded the best results in our experiments. The resulting accuracies, along with the mean number of steps used to attain them, are shown in Table 1.

[1] https://github.com/stanfordnlp/mac-network

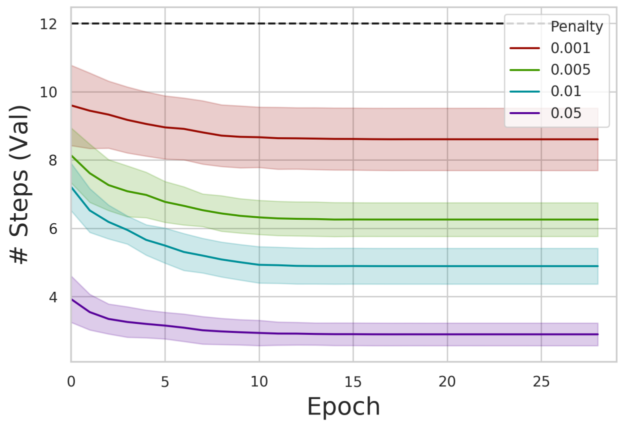

Figure 1. Curves illustrate change in the number of steps employed by DACT-MACs as training progresses. For reference, we include the case of standard non-adaptive MAC as a dotted line (12 steps). Note that in the first epoch the model uses more steps than the value to which it converges, following a behavior similar to the pre-trained MAC, while subsequent epochs reduce computation. Using higher ponder costs translates into less variability in the number of steps used, shown as translucent bands of the same color.

Each question in GQA has been assigned one of 105 *question types* according to the operations needed to answer the question. Therefore, as was the case with question families in CLEVR, we group questions by type with the underlying assumption being that all questions of a type have similar complexities. Figure 2 validates that, as in CLEVR, DACT is adapting computation, and the total amount of computation varies following the time penalty.

1.2. Question Families

For any given synthetic image in the CLEVR dataset, a series of queries are generated by chaining a sequence

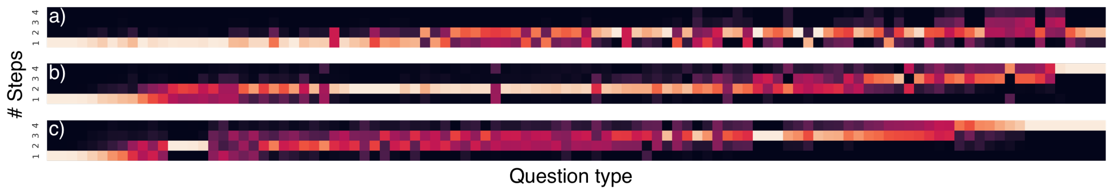

Figure 2. The figure shows the the distribution of the number of steps used by DACT for each one of the 105 different *question types* in the GQA dataset. In order from top (a) to bottom (c) we show how decreasing the *time penalty* (5×10^{-2}, 1×10^{-2}, 5×10^{-3} for a,b, c respectively) results in increased total computation.

Method	Ponder Cost	Steps	Accuracy
MAC+Gate	NA	2	77.51
MAC+Gate	NA	3	77.52
MAC+Gate	NA	4	77.52
MAC+Gate	NA	5	77.36
ACT	1×10^{-2}	1.99	77.17
ACT	1×10^{-3}	2.26	77.04
ACT	1×10^{-4}	2.31	77.21
ACT	0	2.15	77.20
DACT	5×10^{-2}	1.63	77.23
DACT	1×10^{-2}	2.77	77.26
DACT	5×10^{-3}	3.05	77.35
DACT	1×10^{-3}	3.69	77.31

Table 1. Our proposed method (DACT) achieved better accuracy than existing adaptive algorithms on the GQA *test-dev* set, while also adapting computation coherently to the values taken by the ponder cost hyper-parameter. However, the task did not benefit from increased computation, so all adaptive models incur in a small metric loss compared to non-adaptive variants.

of modular operations such as *count*, *filter*, *compare*, *etc*. These functional programs can then be expressed in natural language in multiple ways, for instance translating $count(filtercolor(red, scene()))$ into *"How many <C> <M> things are there?"*, a translation which is accomplished by instantiating the text templates specific to each program following [5]. As a result, questions with the same functional program can be clustered together into question families that share a similar complexity. Figure 3 includes a text template for each of the question families present in CLEVR, sorted by the average number of steps used for validation questions belonging to the specific family.

1.3. Proofs

In this section we prove that our method for building the final answer Y can be interpreted as attending the *intermediate outputs* y_n, with attention weights that follow a valid probability distribution. We include two proofs by induction to show that, for any n, the accumulated answer a_n can be expressed as a weighted sum of all *intermediate outputs* up to the nth step, and that these weights always add up to one.

Proposition. Every accumulated answer a_n can be expressed as a weighted sum of all *intermediate outputs* up to the nth step.

Proof. Assume α_i exists for each y_i such that every $a_{n-1} = y_{n-1}\alpha_{n-1} + \cdots + y_0\alpha_0$. This is trivial to prove for $n = 1$ as $p_0 = 1$ makes $a_1 = y_1 p_0 + a_0(1 - p_0)$ become $a_1 = y_1$.

$$\begin{aligned} a_n &= y_n p_{n-1} + a_{n-1}(1 - p_{n-1}) \\ &= y_n p_{n-1} + (\alpha_{n-1} y_{n-1} + \cdots + \alpha_0 y_0)(1 - p_{n-1}) \\ &= y_n p_{n-1} + \sum_{i=0}^{n-1} y_i(\alpha_i(1 - p_{n-1})) \end{aligned} \qquad \square$$

Proposition. Every accumulated answer a_n can be expressed as a weighted sum of all *intermediate outputs* up to the nth step, and the sum of the weights is equal to one.

Proof. The base case is again trivial to prove since $p_0 = 1$ when $n = 1$. Using the proof above we define β_i to be the weights used to express a_n as a weighed sum of y_i $\forall i \in [1, n]$.

$$\beta_i = \begin{cases} p_{n-1} & \text{if} \quad i = n \\ \alpha_i(1 - p_{n-1}) & \text{otherwise} \end{cases}$$

Assume α_i exists for each y_i such that every $a_{n-1} = \alpha_{n-1} y_{n-1} + \cdots + \alpha_0 y_0$ and $\sum_{i=0}^{n-1} \alpha_i = 1$.

$$\begin{aligned} \sum_{i=0}^{n} \beta_i &= p_{n-1} + \sum_{i=0}^{n-1} \alpha_i(1 - p_{n-1}) \\ &= p_{n-1} + \sum_{i=0}^{n-1} \alpha_i - p_{n-1} \sum_{i=0}^{n-1} \alpha_i \\ &= p_{n-1} + 1 - p_{n-1} \\ &= 1 \end{aligned} \qquad \square$$

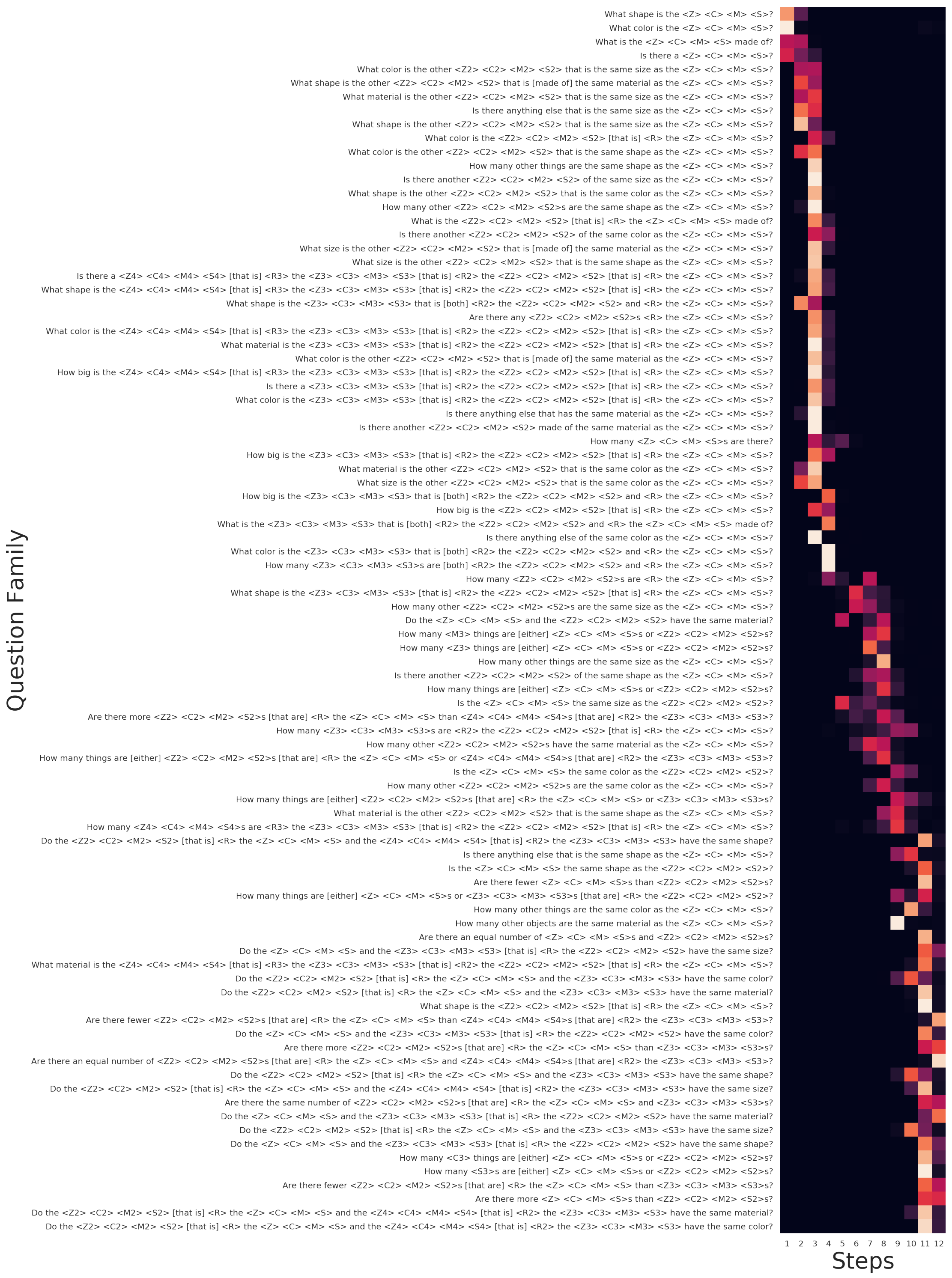

Figure 3. The image above shows the average number of steps used by DACT-MAC ($\lambda = 5 \times 10^{-3}$) for each of the question families present in CLEVR, along with one template for each family to typify the whole group. Following [5], each question is generated by replacing *<Z>, <C>, <M>, and <S>* with the size, color, material and/or shape of objects present in the image. Note that families with fewer supporting objects are more likely to be answered in less steps, and finding the number of objects that possess a pair of qualities (*[both]*) is regarded as generally easier than finding those that possess *[either]*.

References

[1] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018. 1

[2] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. 1

[3] D. Hudson and C.D. Manning. Compositional attention networks for machine reasoning. In *ICLR*, 2018. 1

[4] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 1

[5] Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. 1, 2, 3