Differentiable Memory Allocation Mechanism For Neural Computing

Itamar Ben-Ari[1], *Alan Joseph Bekker*[1]

[1]Advanced Analytics, Intel, Israel
itamar.ben-ari@intel.com, alanbekker@gmail.com

Abstract

Memory neural networks were recently introduced in several variants. DNC by [1] which is one extensive implementation of this kind of models make use of several mechanisms designed to allow the iterative modification of external memory. Three types of attention forms were presented. The attention mechanism which is in-charge of the memory allocation, involves applying a sort operation on the memory blocks usage vector. Since the sort operation is not differentiable, a workaround was introduced to enable the back-propagation step. In our work we propose an alternative differentiable allocation mechanism in the form of a weighted soft-max on the usage vector of the memory blocks. We show that our method achieves 1.75x in training time on bAbI task, moreover we show that our proposed method positively impacts the convergence, stability and accuracy of the neural computer in a copy and repeat task.

1. Introduction

Since the analytical engine model was proposed in 1838 [2], any designed computer is composed of two crucial mechanisms. The CPU is in charge of the computational operations while the memory holds the data to be processed. Despite the recent advances in Deep Learning architectures and algorithms, only a few studies have shown to incorporate external memories to be used by a neural network.
Recurrent neural networks and their two most famous variants [3] and [4] differ from other deep learning models by being able to share information between different time steps in sequential data. The sharing of information is made possible by maintaining a memory which is referred to as the network state. This state is embedded inside the computational model. The work of [5] was the first to propose a system consisting of a neural network and an external memory resource. This work introduced a new neural network model called the Neural Turing Machine (NTM) which have shown to outperform standard LSTM models on simple tasks such as copying and sorting.
In Memory Networks [6], the authors presented a dynamic memory neural model showing outstanding results on cognitive tasks such as question and answering. This work proposed a memory architecture for storing a knowledge database to represent the statements presented to the model. The work of [7] described a similar architecture but unlike [6], it is trainable end-to-end, making it more generally applicable in realistic settings.
Undoubtedly, [1] extended all previous works in a sense that the read and write controller operations were equipped with richer mechanisms allowing iterative modification of external memory. In this work, the memory controller is designed to incorporate three types of attention forms. The first is the content lookup (a.k.a associative recall) mechanism in which part of the controller output is used as a memory key which is compared against the memory blocks content. This produces a similarity score which is used to select memory blocks with similar content, thus enabling information to be completed by the information stored in the memory key. The second mechanism uses a temporal link matrix to hold the consecutively written memory locations. This enables the controller to recover sequences in the order they were written to. The third attention form is the memory allocation mechanism which enables the controller to reallocate memory blocks not being used. The selection of blocks to be reallocated involves sorting of the memory blocks according to a usage measure. Although the proposed model is called a differential neural computer, the reallocation mechanism which involves sorting as part of the algorithm, is non differentiable, causing the entire model to be non-differentiable. The authors of the DNC ignore the non-differential part by passing the incoming gradients in the back-propagation step as is effectively treating the operation as having an identity gradient. In our work we propose a differentiable allocation function which replaces the sorting mechanism. We show that it positively impacts the convergence, stability and accuracy of the neural computer.
The rest of this paper is organized as follows. First we describe the original DNC method, then we present our new allocation mechanism and last we present our improved results over the benchmark in a copy and repeat task and over the bAbI dataset[8].

2. Method

The DNC architecture is composed of a control unit and a Memory Access Unit (MAU) (implemented as sub networks) which are the equivalent of CPU and RAM in a standard computer. The computation process at time step t starts by feeding input data to the controller which in turn outputs two vectors (ξ_t, υ_t). The vector ξ_t is forwarded to the MAU and controls the reading and writing of content to memory. The vector υ_t combined with the Memory Access Unit output r_t produces the DNC output:

$$y_t = \upsilon_t + W_r r_t \tag{1}$$

2.1. Memory Access Unit

The memory of a Differentiable Neural Computer is a matrix $M_{m\times n}$ where each row represents a memory block. The controller reads and writes to M through the MAU by forwarding it a control vector ξ_t at time step t. This vector contains a sub vector $v_t \in R^n$ representing new data to store in memory along with flags which are used to compute weight vectors $w_t^w, w_t^r \in [0,1]^m$, these weights are fuzzy memory block addresses the controller ask to read or write to. In addition, ξ_t contains an erase vector $e_t \in [0,1]^n$ which selects memory cells that should be overwritten. The read and write operations are described by the following equations:

The read operation is defined by a weighted average of the memory blocks:

$$r_t = M_t^\top w_t^r \tag{2}$$

The write operation is a weighted average of the memory matrix M_{t-1} and v_t

$$M_t = M_{t-1} \circ (1 - w_t^w e_t^\top) + w_t^w v_t^\top \tag{3}$$

$(1 - w_t^w e_t^\top)$ can be viewed as fuzzy memory addresses of cells the controller wants to keep. $w_t^w v_t^\top$ is a weighted replication of v_t which will write a scaled version of v_t to each row.

2.2. Memory Addressing

The read and write address weights w_t^r, w_t^w are computed using a combination of three methods. **Associative recall**: where a search key is compared to the content of each memory block and selects the address of the most similar block. **Sequential retrieval**: where the memory blocks are retrieved in the (reverse) order they were written. **Free blocks list**: where the memory blocks are selected from a list of unused blocks. In the Associative recall method The control vector ξ_t contains read and write keys ${k_t}^r, {k_t}^w \in R^n$. These keys are compared to the content of each memory block using a cosine similarity function.

$$D(u, v) = \frac{u \cdot v}{\|u\|\|v\|} \tag{4}$$

The control vector also contains read and write key strengths $\beta_t^r, \beta_t^w \in [1, \infty]$. A key and strength are combined to produce a content-similarity softmax weight.

$$C_t(k)[i] = \frac{exp\{D(k, M_{t-1}[i,\cdot])\beta\}}{\Sigma_j exp\{D(k, M_{t-1}[j,\cdot])\beta\}} \tag{5}$$

where i is the block index. The most similar block is (fuzzy) selected. β controls the sharpness of the weight distribution. A large β means we are effectively reading or writing to a single block.

The sequential retrieval mechanism is less relevant to our work and therefore we skip its explanation and refer the reader to the original paper [1] In the Free Blocks List mechanism The controller can pick the next available space for writing. In order to do so, the MAU first computes how free or unused each memory block is and store it in an allocation weight vector $a_t \in [0, 1]^m$. The vector a_t selects the most available block. The controller sends the MAU an allocation gate $g_t^a \in [0, 1]$ to select which memory addressing mechanism to use for writing.

$$\hat{w}_t^w = g_t^a a_t + (1 - g_t^a) C_t(k^w) \tag{6}$$

where the final write address weight is defined by:

$$w_t^w = \hat{w}_t^w g_t^w \tag{7}$$

and $g_t^w \in [0, 1]$ is a memory protection gate sent by the controller.

A block is not in use if the controller just read its content and asked to free it by sending the MAU a free gate $f_t \in [0, 1]$

The memory usage vector $u_t \in [0, 1]^m$ is defined by the recurrence relation:

$$u_t = (1 - f_t w_{t-1}^r) \circ (u_{t-1} + w_{t-1}^w - u_{t-1} \circ w_{t-1}^w) \tag{8}$$

The original allocation vector a_t is a sharper version of the "non-usage" vector $(1 - u_t)$. It is computed by computing a free block list $\phi \in \mathbb{Z}^m$ which is defined by sorting the indices of the memory blocks in ascending order of usage. The allocation vector a_t is defined by the following equation:

$$a_t[\phi_t[j]] = (1 - u_t[\phi_t[j]]) \prod_{i=1}^{j-1} u_t[\phi_t[i]] \tag{9}$$

As stated in the introduction the proposed sorting method is non differentiable, the authors of DNC suggested to neglect this issue by assuming the gradient of the cost function in respect to this layer is the identity function. They assumed the other network parameters will compensate for this error in back-propagating the gradients through this non-differentiable layer. Instead we propose to replace the free block list mechanism previously executed by the sorting operation by a weighted soft-max on the non-usage vector $(1 - u_t)$ for the computation of a_t.

$$a_t[i] = \frac{exp\{(1 - u_t[i])\beta_a\}}{\Sigma_j exp\{(1 - u_t[j])\beta_a\}} \tag{10}$$

where i is the block index and $\beta_a \in [1, \infty]$ is a new learned parameter which will be part of the control output vector ξ_t. The proposed weighted soft-max serves as an alternative weight sharpening operation similar to the one used in the content similarity mechanism.

From here until the end of the paper we dub to the original allocation mechanism as Non-differentiable Memory Allocation (NMA) and to our proposed mechanism we dub as Differentiable Memory Allocation (DMA).

3. Experiments

3.1. Copy and repeat task

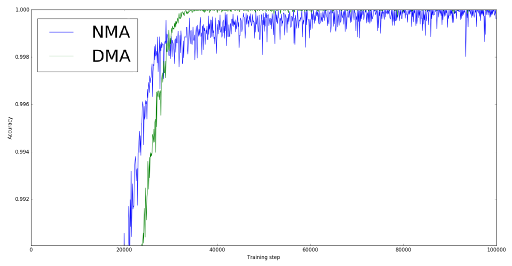

Figure 1: The accuracy in the copy and repeat task as a function of the training step. It can be clearly seen the accuracy achieved by our mechanism (DMA) in green outperforms the original one (DMA) with blue color in terms of convergence time, stability and the final obtained score.

In the "copy and repeat" task [14]. a random binary matrix with a random number of columns is given to the network along with the number of times it should be repeated (tiled along the second dimension) in the output. The task configuration parameters were taken from [14] and are described below. The DNC memory size was set to 16x16 and the controller, which was implemented as an LSTM layer, was given a hidden state size of 64 with 1 write head and 4 read heads.

For each sample, the size of the input matrix and the number of repetitions were randomly selected from 4x{1,2} and {1,2} respectively. We used the RMSProp solver with batch size of 16, learning rate $\tau = 10^{-4}$ and $\epsilon = 10^{-10}$ and run the optimization for 100k iterations. We used gradient clipping with

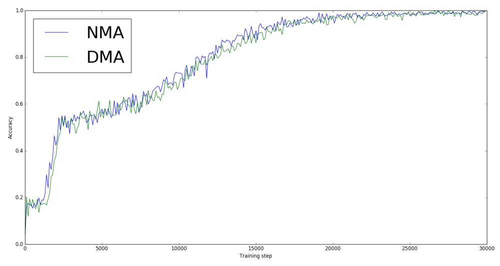

Figure 2: The accuracy in the bAbI task as a function of the training steps. It can be clearly seen that the accuracy achieved by our mechanism (NMA) in green was the same as the original one (DMA) although the training time was reduced by 1.75x.

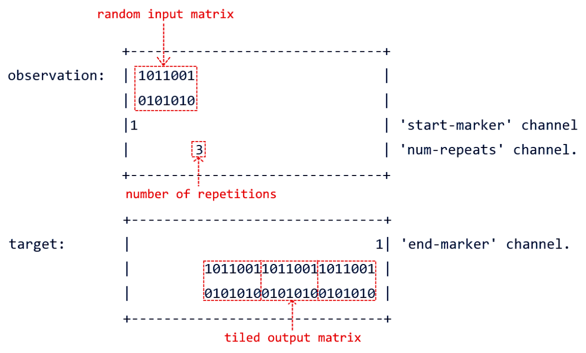

Figure 3: The observations and targets are binary matrices. For readability purposes we replaced zero paddings with blanks. The observation random input matrix is comprised of i.i.d uniform-random binary values. The target tiled output matrix is comprised of the input matrix repeated for some number of times. The number of columns of the input matrix and the number of times it is repeated in the target are both discrete random variables distributed according to uniform distributions whose parameters are configured at construction time.

norm 50 and clipped the controller and DNC output values to 20.

We calculated the average cross entropy loss and accuracy every 100 iterations and plotted them in Figure 2. It can be clearly seen that our proposed DMA method converges faster achieving a lower cross entropy loss and higher accuracy than the NMA. In addition the DMA loss and accuracy curves seem to be stable than the NMA ones, we attribute this to the fact that the original version of the model was non-differentiable and was apparently prejudicing the optimization process.

3.2. bAbI task

The Facebook bAbI dataset [8] is a synthetic dataset for testing a models ability to retrieve facts and reason over them. Each task tests a different skill that a question answering model ought to have, such as coreference resolution, deduction, and induction. The task configuration parameters were taken from [1] and are described below.

The DNC memory size was set to 64x64 and the controller,

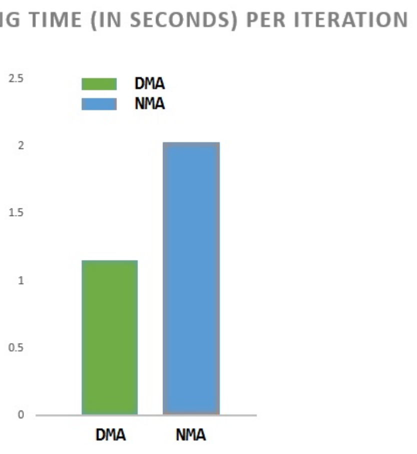

Figure 4: Training time per one iteration, it can be seen that our method (DMA) in greed took 1.15 sec while the regular DNC in blue (NMA) took 2.02 sec, meaning we are 1.75x faster

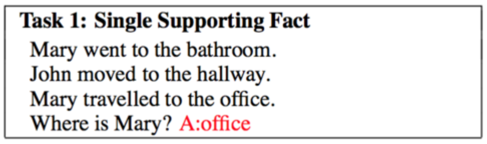
Task 1: Single Supporting Fact
Mary went to the bathroom.
John moved to the hallway.
Mary travelled to the office.
Where is Mary? A:office

Figure 5: Task number 1 in bAbI dataset. The task consist with a set of supporting facts and a deductive conclusion the network is supposed to respond.

which was implemented as an LSTM layer, was given a hidden state size of 64 with 1 write head and 4 read heads. We used the RMSProp solver with batch size of 1, learning rate $\tau = 10^{-3}$ and $\epsilon = 10^{-10}$ and run the optimization for 50k iterations. We used gradient clipping with norm 50 and clipped the controller and DNC output values to 20.
We calculated the average cross entropy loss and accuracy every 100 iterations and plotted them in Figure 2.

Since we replaced the sorting mechanism with the weighted soft-max module we gain a significant speed-up. For each sample, DNC sorts the network memory in the feed-forward step. In our DMA version we only compute the soft-max which requires significantly less computation instructions. As can be seen in 3.1 in our proposed method each iteration takes 1.15 seconds while it required 2.02 seconds to the NMA model, meaning we achieved a speed-up of 1.75x in the training and inference time.

4. Conclusions

In this work we proposed a differentiable allocation mechanism which replaced the original non-differentiable sorting function in the DNC. We have shown that our method achieves 1.75x in training time on bAbI task and positively impacts the convergence, stability and accuracy of the neural computer in the copy repeat task.

5. References

[1] A. Graves, G. Wayne, M. Reynolds, T. Harley, I. Danihelka, A. Grabska-Barwińska, S. G. Colmenarejo, E. Grefenstette, T. Ramalho, J. Agapiou *et al.*, "Hybrid computing using a neural network with dynamic external memory," *Nature*, vol. 538, no. 7626, pp. 471–476, 2016.

[2] A. G. Bromley, "Charles babbage's analytical engine, 1838," *Annals of the History of Computing*, vol. 4, no. 3, pp. 196–217, 1982.

[3] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[4] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," 1999.

[5] A. Graves, G. Wayne, and I. Danihelka, "Neural turing machines," *arXiv preprint arXiv:1410.5401*, 2014.

[6] J. Weston, S. Chopra, and A. Bordes, "Memory networks," *arXiv preprint arXiv:1410.3916*, 2014.

[7] S. Sukhbaatar, J. Weston, R. Fergus *et al.*, "End-to-end memory networks," in *Advances in neural information processing systems*, 2015, pp. 2440–2448.

[8] Facebook-Research, *bAbI Facebook Data-set*, 2017. [Online]. Available: https://research.fb.com/downloads/babi/

[9] A. Kumar, O. Irsoy, P. Ondruska, M. Iyyer, J. Bradbury, I. Gulrajani, V. Zhong, R. Paulus, and R. Socher, "Ask me anything: Dynamic memory networks for natural language processing," in *International Conference on Machine Learning*, 2016, pp. 1378–1387.

[10] A. Santoro, S. Bartunov, M. Botvinick, D. Wierstra, and T. Lillicrap, "Meta-learning with memory-augmented neural networks," in *International conference on machine learning*, 2016, pp. 1842–1850.

[11] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems*, 2015, pp. 1693–1701.

[12] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 2692–2700.

[13] J. Weston, A. Bordes, S. Chopra, A. M. Rush, B. van Merriënboer, A. Joulin, and T. Mikolov, "Towards ai-complete question answering: A set of prerequisite toy tasks," *arXiv preprint arXiv:1502.05698*, 2015.

[14] J. Ramapuram, *Repeat Copy Task*, 2017. [Online]. Available: https://github.com/deepmind/dnc/blob/master/repeat_copy.py