Controlling Attention To Solve Working Memory Tasks Using a Memory-Augmented Neural Network

T.S. Jayram[1], **Younes Bouhadjar**, **Tomasz Kornuta**[2], **Ryan L. McAvoy**, **Alexis Asseman**, and **Ahmet S. Ozcan**[3]

IBM Research AI, Almaden Research Center, San Jose, CA 95120

Abstract

We introduce a memory-augmented neural network, called Differentiable Working Memory (DWM), that captures some key aspects of attention in working memory. We tested DWM on a suite of psychology inspired tasks, where the model had to develop a strategy only by processing sequences of inputs and desired outputs. Thanks to novel attention control mechanisms called *bookmarks*, the model was able to rapidly learn a good strategy—generalizing to sequence lengths even two orders of magnitude larger than that used for training—allowing it to retain, ignore or forget information based on its relevance. The behavior of DWM is interpretable and allowed us to analyze its performance on different tasks. Surprisingly, as the training progressed, we observed that in some cases the model was able to discover more than one successful strategy, possibly involving sophisticated use of memory and attention.

Introduction

Keeping information in mind after it is no longer present in the environment is critical for all higher cognitive behaviors. *Working memory* (WM) is the term used for this ability, which is distinct from the storage of vast amount of information in long-term memory (Baddeley, 2003; Oberauer, 2009). The two main distinguishing characteristics of WM are the limited capacity (3-5 items) (Cowan, 2001) and temporary retention (secs-minutes). Hence, WM is not a storage per se, but a mental workspace utilized during planning, reasoning and solving problems. Most psychologists differentiate WM from "short-term" memory because it can involve the manipulation of information rather than being a passive storage (Cowan, 2017). Along the same lines, Engle, Tuholski, Laughlin, and Conway (1999) argued that WM is *all about the capacity for controlled, sustained attention in the face of interference or distraction.* Attention-control is a fundamental component of the WM system and probably the main limiting factor for capacity (Conway & Engle, 1994; Engle & Kane, 2004). Consequently, the inability to effectively parallel process two-attention demanding tasks limits our multitasking performance severely.

Over the past several decades psychologists have developed tests to measure the individual differences in WM capacity and better understand the underlying mechanisms. These tests have been carefully crafted to focus on the specific aspects of WM such as task-driven attention control, interference and capacity limits (Oberauer & Lin, 2017). The best known and successfully applied class of tasks for measuring WM capacity is the "complex span" paradigm. The challenge presented by complex span tasks is recalling the list of items, despite being distracted by the processing task. Studies show that individuals with high WM capacity are less likely to store irrelevant distractors (Vogel, McCollough, & Machizawa, 2005) and they are better at retaining task-relevant information (Maxcey-Richard & Hollingworth, 2013). Developing task-driven strategies for cognitive control are essential for the effective use of WM.

In the past there were several attempts to build computational models that mimic the operation of a human working memory (Henson, 1998; Farrell & Lewandowsky, 2002; Oberauer & Lewandowsky, 2011; Lemaire & Portrat, 2018). For example, Burgess and Hitch (1999, 2005) used a shallow neural network and put the emphasis on Hebbian-like learning rules that enabled the model to achieve similar behavior to the one achieved by human subjects. In those works the experimental paradigm was the serial recall task, which is limited in testing the complex processing and attention component of WM. One notable exception was (Oberauer, Lewandowsky, Farrell, Jarrold, & Greaves, 2012), where the authors focused on the complex span task.

The power of maintaining information over time has also been recognized by the AI community. Starting with the basic recurrent neural network architectures (Elman, 1990; Hopfield & Tank, 1986) followed by the introduction of gating mechanisms (Hochreiter & Schmidhuber, 1997), the research has recently moved onto more complex architectures with memories (Graves, Wayne, & Danihelka, 2014; Joulin & Mikolov, 2015; Weston, Chopra, & Bordes, 2015; Graves et al., 2016; Santoro, Bartunov, Botvinick, Wierstra, & Lillicrap, 2016; Gulcehre, Chandar, & Bengio, 2017). These models are typically applied to tasks (e.g. associative recall, bAbI QA (Weston, Bordes, Chopra, & Mikolov, 2015)) that require a complex mixture of long-term memory (episodic and semantic) and working memory. In the human brain, these kinds of memory systems are distinct: working memory is instantiated in multiple interconnected areas with the prefrontal cortex playing a major role (Constantinidis & Klingberg, 2016), whereas for episodic memory the hippocampus is the critical structure (Fortin, Agster, & Eichenbaum, 2002). Studying these mechanisms separately is necessary to disen-

[1] jayram@us.ibm.com. Primary contact author.

[2] tkornut@us.ibm.com

[3] asozcan@us.ibm.com

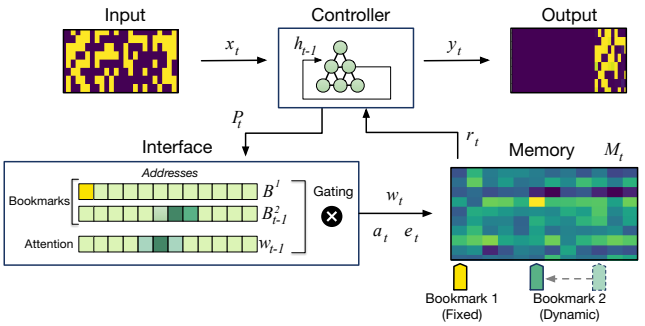


Figure 1: Illustration of the operation of the DWM Model

tangle the contributions of each memory system and develop a detailed understanding of human intelligence.

In this work, we take inspiration from biological and computational models of working memory to develop an *artificial* neural network model augmented with external memory, called *Differentiable Working Memory (DWM)*. We provide the DWM model a set of generic mechanisms to encode inputs, access the memory, and control its attention over the memory contents. Key to this design is a new attention control mechanism in memory, called *bookmarks*, which helps in dealing with interference. In contrast to previous works, we applied our model to a variety of psychometry-inspired tasks, each requiring the system to control its attention in a slightly different way. We show that the DWM is capable of solving these diverse tasks by looking only at input-output pairs using the provided attention mechanisms. The model is easy to train and accurately generalizes to sequences two orders of magnitude longer than the training data. We also describe the strategies that the DWM develops during training and demonstrate that the bookmarks can effectively deal with interference in complex tasks. Finally, we show that the DWM is also capable of learning multiple strategies during training and, moreover, develop better strategies in the presence of memory scarcity.

Differentiable Working Memory (DWM)

The operation of Differentiable Working Memory (DWM) model is presented in Fig. 1. As a memory-augmented neural network (MANN), the DWM has three main components: a *controller*, an *external memory* and an *interface* between the two (Zaremba, Mikolov, Joulin, & Fergus, 2016). The interface is composed of several *attention* mechanisms that the controller learns to use by generating appropriate parameters for accessing the external memory. The procedure is sketched in Algorithm 1. We describe the main steps, Lines 4–7, below in order of significance.

Attention control. The memory consists of N addresses, each storing a vector of real numbers of length L. Thus the memory contents are given by an $L \times N$ matrix of real numbers. The read and write operations share a single attention

Algorithm 1 Operation of Differentiable Working Memory

1: Initialize:
- the hidden state h_0 and memory array M_0
- the read/write attention vector w_0
- bookmarks $\{B_0^i : i = 1,2,\ldots,K\}$

2: **for** $t \in \{1,2,\ldots,T\}$ **do**
3: **Memory read**: $r_t \leftarrow M_{t-1} w_{t-1}$
4: **Controller**: $h_t, P_t \leftarrow \phi(x_t, r_t, h_{t-1})$
5: **Memory update**: $M_t \leftarrow \mathsf{update}(w_{t-1}, P_t, M_{t-1})$
6: **Attention control**: $w_t, \{B_t^i\} = \mathsf{attn}(w_{t-1}, \{B_{t-1}^i\}, P_t)$

mechanism. Further, we use *soft addressing*: let w denote a non-negative weight vector of dimension N whose components sum up to 1. Each component indicates the *relative strength* with which a value (i.e. a vector of dimension L) will be read/written at the corresponding address.

The behavioral studies indicate that people can access memories sequentially (Singh, Tiganj, & Howard, 2018). For that reason we have decided to add a mechanism based on circular convolution, similar to the one used in Neural Turing Machine (NTM) (Graves et al., 2014), enabling it to shift attention over memory:

$$w_t = \text{convolution}(w_t^g, s_t), \tag{1}$$

where w_t^g and w_t are the vectors of attention weights over cells in memory at time t before and after shifting, and s_t is a shift vector outputted by the controller. We also apply a weight sharpening step typically used after the shifting; as we observed, it seemed to be crucial for models using circular convolution to converge properly.

The Embedded-Processes Framework (Cowan, 1988) assumed the presence of Focus of Attention (FOA). In this model the items in the FOA were interpreted as pointers to the representations stored in the long-term memory rather then being the actual representations themselves. Inspired by this concept, we introduced a new attention mechanism called *bookmarks* that store the system's attention at previous time steps. This is recorded in K bookmark vectors $\{B_t^i : i = 1,2,\ldots,K\}$ at time t. The first bookmark $B^1 := B_t^1$ has a time-independent *fixed* attention to a single address so that the model maintains a reference frame for memory. The remaining bookmarks are *dynamic*: at time t, the DWM must decide whether to remember its previous (read/write) attention w_{t-1} by recording it in a bookmark, as:

$$B_t^i = g_t^i w_{t-1} + (1 - g_t^i) B_{t-1}^i, \quad i = 2,3,\ldots K, \tag{2}$$

where the gating parameter g_t^i is emitted by the controller. As discussed later, we found in our experiments that even limiting to only two bookmarks (one *fixed* and one *dynamic*), the model could still solve all tasks.

The DWM must also decide before moving sequentially whether it wishes to return to a previous bookmark. For this

purpose we once again use a gating mechanism, this time in a slightly more sophisticated form:

$$w_t^g = \delta_t^0 w_{t-1} + \sum_{i=1}^{K} \delta_t^i B_{t-1}^i, \tag{3}$$

where $\delta_t^i, i = 1,2,\ldots K$ are gating parameters emitted by the controller. These gating parameters are scalars, normalized using a softmax function.

The DWM attention control incorporates the presented mechanisms by applying equations (3), (2), and (1) in order.

Memory read and update. We use the standard formula for soft attention, e.g., (Weston, Chopra, & Bordes, 2015), that computes the read vector r_t from memory M_{t-1}:

$$r_t = M_{t-1} w_{t-1} \tag{4}$$

For memory update, we decided to use the simple erase-add scheme derived from NTM (Graves et al., 2014):

$$M_t = M_{t-1} \circ (E - e_t \otimes w_t) + a_t \otimes w_t, \tag{5}$$

where E is a matrix of all ones, e_t and a_t are vectors of content to be erased and added to memory, respectively. The parameters e_t and a_t are emitted by the controller.

Controller. The controller's role is to process inputs so as to produce outputs as well as interface parameters. In DWM we use a single-layer recurrent neural network controller:

$$h_t = \sigma(W_h[x_t, h_{t-1}, r_t]), \tag{6}$$

where x_t denotes the current input and h_{t-1} and r_t are the hidden state and vector read from memory in the previous time step, respectively. To prevent the controller from acting as a separate working memory, the hidden state size is chosen to be smaller than that of a single input vector (in all of our experiments it was set to 5). The output logits, y_t and interface vector P_t are produced similarly as:

$$y_t = W_y[x_t, h_{t-1}, r_t] \tag{7}$$

$$P_t = W_P[x_t, h_{t-1}, r_t] \tag{8}$$

W_h, W_y and W_P are the only trainable parameters of our DWM model. The interface vector P_t contains all of the parameters that control reading, writing, and the attention mechanisms. Denoting the unprocessed parameters from the interface with a hat, the full list of parameters in P_t is as follows:

- The write vector $a_t \in \mathbb{R}^{N_M}$
- The erase vector $e_t = \sigma(\hat{e}_t) \in [0,1]^{N_M}$
- The shift vector $s_t = \text{softmax}(\text{softplus}(\hat{s})) \in [0,1]^3$
- The bookmark update gates $g_t^i = \sigma(\hat{g}_t^i) \in [0,1]^{K-1}$
- The attention update gate $\delta_t^i = \text{softmax}(\hat{\delta}_t^i) \in [0,1]^{K+1}$
- The sharpening parameter $\gamma = 1 + \text{softplus}(\hat{\gamma}) \in [1,\infty]$

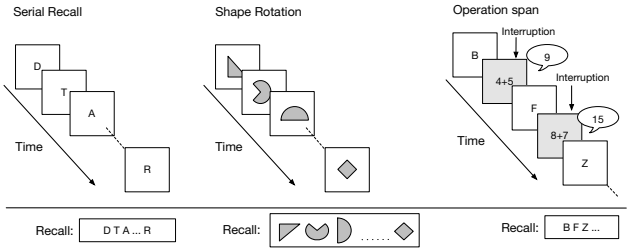


Figure 2: Exemplary tasks for testing the performance of human working memory

Psychometric tasks for working memory

Over last several decades cognitive psychologists have developed many psychometric tests (Conway et al., 2005) to measure the performance of human WM (See Fig. 2 for examples). These tasks are mainly sequential and typically divided into verbal and visuospatial domains. Given that diversity and various categorizations by different researchers, we built a taxonomy of tasks (Fig. 3) and carefully selected tasks that seem to be the most representative for a given category. First order categorization is based on the number and complexity of tasks. For simple tasks, the presence of data manipulation is the next level sub-category, with Serial Recall being a prime example of a task without manipulation. The tasks requiring manipulation we further categorized into spatial and temporal domains. The complex tasks involve multiple sequential inputs or sub-tasks but not necessarily imply "multi-tasking". We follow the framework of Clapp, Rubens, and Gazzaley (2009) to distinguish the sources of goal interference, i.e. Distraction (to-be-ignored) and Interruption (i.e. multi-tasking). For example, in Operation Span (Fig. 2c) the subjects had to attend and process the summation (Interruption) even though they did not need to recall the results afterwards, whereas in Reading Span (Daneman & Carpenter, 1980) subjects had to read sentences and recall the last word of each one. In addition to the classical psychometric tasks, we introduced several tasks testing the effectiveness of attention control in memory (Ignore, Forget and Scratch Pad). As a result, a suite of tasks presented in Table 1 emerged.

The input to every task is a sequence of items. As we wanted to be agnostic to audio/visual preprocessing, we have implemented those tasks using sequences of randomly generated *binary patterns* (vectors of bits) as items (instead of words/images). At a higher level, we view the input as a *concatenation* of various subsequences that represent different functional units of processing. For all simple tasks, there is only one type of subsequence, and the output will be reproduced from the memory with or without manipulation. The complex tasks may involve a secondary set of subsequences, optionally requiring immediate output as indicated in the Forget and Operation Span tasks.

Additionally, we use a constant-sized set of special items (called *command markers*) to both mark the beginning of a subsequence as well as indicate its functional type. It is im-

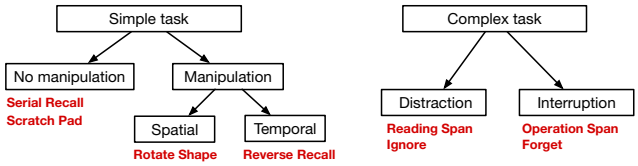


Figure 3: Taxonomy of working memory tasks

portant to note that the system does not know a priori what kind of operation is associated with a given type of marker and must learn that from data. We ignored markers in Table 1 to keep the description simple. Also, note that such markers are also commonly employed in the psychometric tests, e.g., see McNab and Klingberg (2008).

Experimental results

We evaluated the performance of DWM on the proposed tasks and compared it to two models: LSTM (Long Short-Term Memory) (Hochreiter & Schmidhuber, 1997), considered as a classical baseline for sequential problems, and DNC (Differentiable Neural Computer) (Graves et al., 2016) being one of the state-of-the-art MANN models. In our implementation we used the MI-Prometheus (Kornuta et al., 2018) framework built on top of PyTorch (Paszke et al., 2017). During training we used the Adam (Adaptive Momentum) optimizer (Kingma & Ba, 2014) and (average) binary cross-entropy as the loss function. We apply early stopping based on validation loss (10^{-4}). Additionally, we terminate training when the number of training episodes reach 100,000 where a single episode involves processing a batch of sequences. The size of batch was a hyper-parameter that was tuned along with training rate for each model using validation loss as the reference.

	Task		(I)input/(O)output sequences	
Simple	Serial Recall	I:	$x_1 x_2 \dots x_n$	$\_\_\dots\_$
		O:	$\_\_\dots\_$	$x_1 x_2 \dots x_n$
	Scratch Pad	I:	$\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_k$	$\_$
		O:	$\_\_\dots\_$	$\mathbf{x}_k$
	Reverse Recall	I:	$x_1 x_2 \dots x_n$	$\_\_\dots\_$
		O:	$\_\_\dots\_$	$x_n x_{n-1} \dots x_1$
	Rotate Shape	I:	$x_1 x_2 \dots x_n$	$\_\_\dots\_$
		O:	$\_\_\dots\_$	$x_1^{\circlearrowleft} x_2^{\circlearrowleft} \dots x_n^{\circlearrowleft}$
Complex	Reading Span	I:	$\mathbf{x}_1 \mathbf{x}_2 \dots \mathbf{x}_k$	$\_\_\dots\_$
		O:	$\_\_\dots\_$	$z_1 z_2 \dots z_k$
	Ignore	I:	$\mathbf{x}_1 \mathbf{y}_1 \dots \mathbf{x}_k \mathbf{y}_k$	$\_\dots\_$
		O:	$\_\_\dots\_\_$	$\mathbf{x}_1 \dots \mathbf{x}_k$
	Operation Span	I:	$\mathbf{x}_1 y_1 \_\dots \mathbf{x}_k y_k \_$	$\_\dots\_$
		O:	$\_\_ y_1^{\circlearrowleft} \dots \_\_ y_k^{\circlearrowleft}$	$\mathbf{x}_1 \dots \mathbf{x}_k$
	Forget	I:	$\mathbf{x}_1 \mathbf{y}_1 \_\dots \mathbf{x}_k \mathbf{y}_k \_$	$\_\dots\_$
		O:	$\_\_ \mathbf{y}_1 \dots \_\_ \mathbf{y}_k$	$\mathbf{x}_1 \dots \mathbf{x}_k$

Table 1: Working Memory Tasks. A bold letter denotes a subsequence of items. The | sign indicates delay between input and output of the primary subsequence(s). Above, $x_i^{\circlearrowleft}$ denotes the circular shift of x_i by half its bitlength. In the Reading Span task, z_i is the last item of $\mathbf{x}_i$.

Task	Validation Accuracy Seq. Size 100 [%]			Test Accuracy Seq. Size 1000 [%]		
	LSTM	DNC	DWM	LSTM	DNC	DWM
Serial	53.3*	100	100	50.2*	64.6	100
Scr. Pad	71.3*	100	100	70.0*	75.0	100
Reverse	53.0*	50.6*	100	50.4*	50.2*	99.8
Rot. Shape	52.2*	100	100	50.2*	60.9	100
Read. Span	50.9*	53.4*	100	50.4*	49.0*	91.9
Ignore	56.1*	69.3*	100	50.9*	50.0*	90.0
Op. Span	58.2*	79.2*	99.9	51.3*	53.6*	99.6
Forget	55.9*	69.4*	98.9	50.5*	49.9*	94.1

Table 2: Summary of experimental results. The first column is the average of validation accuracies achieved by the models for 10 training runs on each task. The second column is the average of test accuracies achieved by models that converged during training. For the majority of tasks, the DNC and LSTM models did not converge. In those cases (indicated with *) we report scores of the best (even though diverged) model.

As stated in the introduction, the main question we wanted to answer was whether a model can learn an algorithm to solve a task. In case of tasks presented in Table 1, this implies that the model should generalize over the sequence lengths. For that reason, our methodology assumed that we will use different lengths of sequences for training (up to 10), validation (exactly 100) and testing (exactly 1000). Although human WM does not have the capacity to handle 1000 items, our goal was to show that the model truly generalizes in that actually develops an effective memory strategy, i.e., it learns an algorithm to solve the task.

All models achieved perfect accuracies on training sequences. However, as presented in Table 2, LSTM and DNC struggled with generalization to longer sequences. Besides, the DWM models converged faster, requiring less than 5000 episodes in most cases (exemplary convergence plot is presented in Fig. 4). The convergence speed is associated with number of trainable parameters of those models (the DWM controller had 1066, the DNC had 4,792, whereas for LSTM baseline we used stacked LSTM with 3 layers and over 5 million trainable weights). Please note that *fair comparison* simply made no sense, as the LSTM and DNC models with less trainable parameters could not even learn to generalize over short (i.e. training) sequences. Aside of that, we hypothesize that the DNC had problems with convergence because of the

complexity of its attention and memory management mechanisms (the *Temporal Link Matrix*, in particular).

Analysis of strategies for solving tasks

The proposed tasks require the models to develop different strategies for solving them. For example, ignoring distractions without encoding them in the memory is arguably the best strategy to minimize memory consumption. On the other hand, for a complex task with an interruption (i.e. multitasking), the secondary task cannot be ignored and may require extensive memory usage. In this case, the best strategy might be to forget (e.g. erase or overwrite) the secondary information as soon as possible in order to maintain sufficient memory capacity for the main task.

During the training and testing of all of the tasks reported in Table 2 we provided sufficient memory size, so that the system could store all the encoded input items in the memory (if it has chosen to). However, limitation of the memory size can force the system to develop more memory efficient strategies, thus we decided to investigate that issue further.

Strategies for the Scratch Pad task The goal of the Scratch Pad task is to recall only the last input subsequence.

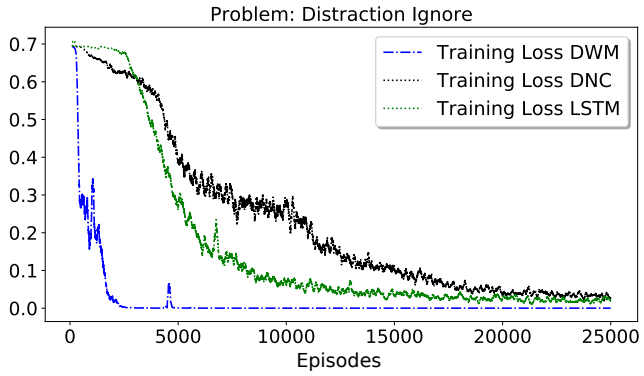


Figure 4: Convergence of the best models on Ignore task

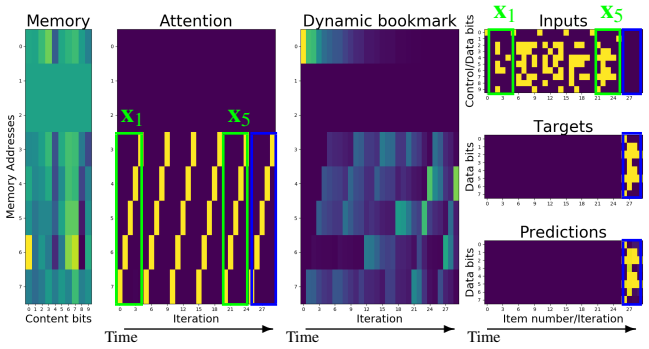


Figure 5: Overwrite strategy developed by DWM for Scratch Pad (episode 969). Memory plot contains a snapshot of the memory content from the last iteration, whereas the other ones present concatenation of states from consecutive iterations (evolution in time)

Given the DWM mechanisms, we expect two possible strategies for the model to learn in order to solve this task.

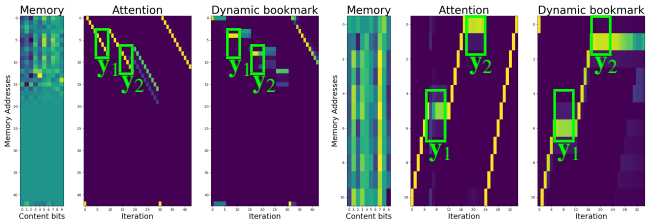
The "Expand" strategy exploits the fact that memory can be used in a similar way to a circular buffer, storing each consecutive subsequences one after the other in the memory. In this case the model should write each subsequence, then place the *dynamic bookmark* at the start of given subsequence, and then update the bookmark position to the beginning of the next subsequence. Finally, when the model receives a command marker indicating it needs to recall, it should recall the attention associated with that *dynamic bookmark* and then retrieve consecutive items one by one by shifting.

The "Overwrite" strategy for the Scratch Pad relies on the fact that when a new subsequence appears, the elements from the previous one can be discarded. The model could exploit this by learning to recall attention stored in the *fixed bookmark* every time it processes a command marker denoting the next subsequence, which will result in overwriting the previous subsequences until the system is told to recall. This strategy may be interpreted as memory saving, as the system reuses the same addresses and overwrites them repeatedly.

To our (initial) surprise, the model *always* developed the Overwrite strategy, irrespective of the memory size (i.e. as long as the memory size was sufficient to fit all the encoded items of a single subsequence). An exemplary run of an early training episode is presented in Fig. 5. Note that memory addresses 1 and 2 remain unchanged and the model stores consecutive items of subsequences $\mathbf{x}_1$ to $\mathbf{x}_5$ in the same addresses 3-7. After analyzing several runs, we hypothesize that overwriting was simpler to learn for this task because: a) both for storing and recalling the command markers, the model had to learn exactly the same behavior: recalling the attention stored in the *fixed bookmark*, b) for every other input item it had to shift by one address location with the circular convolution. As a result, it could converge rapidly by disregarding the control (update, recalling) of the *dynamic bookmark* (in the later training episodes the *dynamic bookmark* was typically "following" the current attention, despite it wasn't recalled at all).

Strategies for the Ignore task The main goal of the Ignore task is to test the retention capabilities of the system in the presence of distractors. For this task the input consists of two types of subsequences $\mathbf{x}$ and $\mathbf{y}$, where the system is supposed to ignore all $\mathbf{y}_i$ and at the end recall $\mathbf{x}_i$ one by one in the order of their appearance. The task can be solved with two strategies which we call "Overwrite" and "Skip".

The "Overwrite" strategy involves overwriting of the distractors, similarly to the "Overwrite" from Scratch Pad task. It assumes that model will store the consecutive items in memory and use the bookmark for moving its attention to the first address containing $\mathbf{y}$ to be overwritten. The difference is, however, that in here the model must learn to use the *dynamic bookmark* for that purpose. Our experiments with sufficient memory have shown that the system can learn this strategy. Exemplary plot from one of the final training episodes (Fig. 6a) shows that the *dynamic bookmark* re-



(a) Overwrite strategy (episode 17000)

(b) Skip strategy (limited memory, episode 5614)

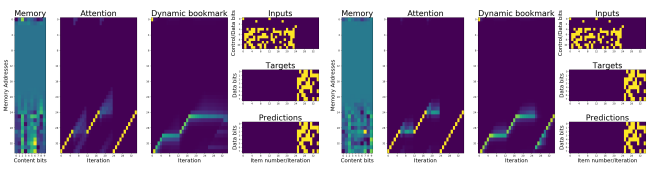
Figure 6: Strategies developed by DWM for solving the Ignore task (two different training runs)

tains its attention while processing items from $\mathbf{y}_1$ and $\mathbf{y}_2$. As soon as the command marker indicating $\mathbf{x}$ appears, the model jumps back its attention to the *dynamic bookmark* and starts to overwrite memory content. Finally, when the recall marker appears, it recalls the attention stored in the *fixed bookmark*.

The "Skip" strategy involves ignoring elements within the $\mathbf{y}$ subsequences, i.e. *skipping* writing them to memory. Our experiments with limited memory have shown that the model could also learn this strategy. Exemplary plots from the final episode from one of the training runs are presented in Fig. 6b. Note that in this case the model has learned to keep its attention focused on a single address for all items of $\mathbf{y}_i$ and shift attention only for items belonging to $\mathbf{x}_i$.

That behavior of the model that mastered the "Skip" strategy seems to be more difficult from the operational point of view. In the "Overwrite" strategy the system develops a reactive behavior, i.e. it always performs convolutional shift except for the rare cases when it hits the command marker – at that point it has to retrieve attention from one or the other bookmark. In the "Skip" strategy the command markers for $\mathbf{x}$ and $\mathbf{y}$ activate one of two distinct operation modes that will be executed for the whole subsequence until hitting the next marker, i.e. for $\mathbf{x}$ attention is supposed to move to the next address, whereas for $\mathbf{y}$ it is supposed to stay at the same position. The only way to perform this is that the controller must learn how to *carry the information about the current operation mode* from one iteration to another in its hidden state, which is more difficult to learn.

We performed several experiments to support that hypothesis. In Fig. 7 we present two episodes from one of the training runs when the operation of the system seems to be evolving from one strategy to the other. At the early stages of the training (Fig. 7a) we can observe that the attention shift with the circular convolution is active for both types of input subsequences, whereas the dynamic bookmark already learned how to *follow* attention for $\mathbf{x}$ and *freeze* for $\mathbf{y}$. As learning to shift attention is crucial for learning both storing and recall, model has to master that first. However, once achieved, it seems to switch to different operation mode. Obviously, learning two modes is simpler for *dynamic bookmark*, as it possesses simpler gating mechanism and cannot shift its attention. As the training progresses (Fig. 7b) the model finally learns to freeze its attention when processing $\mathbf{y}$ subsequences.



(a) episode 1757: utilization of Overwrite strategy

(b) episode 17685: utilization of Skip strategy

Figure 7: Evolution of the strategy developed by DWM, when learning the Ignore task (during a single training run, intentionally used the same verification sequence in both cases)

Conclusion

We have demonstrated that DWM has the appropriate attention mechanisms to tackle psychology-inspired tasks. When compared to existing models such as DNC, LSTM it appeared to manage generalization to much longer sequences. Besides, after careful step-by-step analysis we discovered that the model is able to develop more than one strategy to control attention and use its memory resources for a given task. While some strategies are harder to learn, DWM can develop them by first finding *any* working strategy and then gradually modifying it towards a different one as learning progresses. Why the model seems to prefer some strategies is intriguing and worth further investigation. Another direction is to incorporate this mechanism into a larger system in order to solve tasks that require both working and long-term memory.

References

Baddeley, A. (2003). Working memory: Looking back and looking forward. *Nature Reviews Neuroscience*, *4*(10), 829–839.

Burgess, N., & Hitch, G. (1999). Memory for serial order: a network model of the phonological loop and its timing. *Psychological review*, *106*(3), 551.

Burgess, N., & Hitch, G. (2005). Computational models of working memory: putting long-term memory into context. *Trends in cognitive sciences*, *9*(11), 535–541.

Clapp, W., Rubens, M., & Gazzaley, A. (2009). Mechanisms of working memory disruption by external interference. *Cerebral Cortex*, *20*(4), 859–872.

Constantinidis, C., & Klingberg, T. (2016, may). The neuroscience of working memory capacity and training. *Nature Reviews Neuroscience*, *17*, 438.

Conway, A., & Engle, R. (1994). Working memory and retrieval: A resource-dependent inhibition model. *Journal of Experimental Psychology: General*, *123*(4), 354.

Conway, A., Kane, M., Bunting, M., Hambrick, D., Wilhelm, O., & Engle, R. (2005). Working memory span tasks: A methodological review and user's guide. *Psychonomic bulletin & review*, *12*(5), 769–786.

Cowan, N. (1988). Evolving conceptions of memory storage, selective attention, and their mutual constraints within the human information-processing system. *Psychological bulletin*, *104*(2), 163.

Cowan, N. (2001). The magical number 4 in short-term memory: A reconsideration of mental storage capacity. *Behavioral and Brain Sciences*, *24*(1), 87–114.

Cowan, N. (2017). The many faces of working memory and short-term storage. *Psychonomic Bulletin and Review*, *24*(4), 1158–1170.

Daneman, M., & Carpenter, P. (1980). Individual differences in working memory and reading. *Journal of verbal learning and verbal behavior*, *19*(4), 450–466.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, *14*(2), 179–211.

Engle, R., & Kane, M. (2004). Executive attention, working memory capacity, and a two-factor theory of cognitive control. *Psychology of learning and motivation*, *44*, 145–200.

Engle, R., Tuholski, S., Laughlin, J., & Conway, A. (1999). Working memory, short-term memory, and general fluid intelligence: a latent-variable approach. *Journal of experimental psychology: General*, *128*(3), 309.

Farrell, S., & Lewandowsky, S. (2002). An endogenous distributed model of ordering in serial recall. *Psychonomic bulletin & review*, *9*(1), 59–79.

Fortin, N. J., Agster, K. L., & Eichenbaum, H. B. (2002, mar). Critical role of the hippocampus in memory for sequences of events. *Nature Neuroscience*, *5*, 458.

Graves, A., Wayne, G., & Danihelka, I. (2014). Neural turing machines. *arXiv preprint arXiv:1410.5401*.

Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., ... others (2016). Hybrid computing using a neural network with dynamic external memory. *Nature*, *538*(7626), 471.

Gulcehre, C., Chandar, S., & Bengio, Y. (2017). Memory augmented neural networks with wormhole connections. *arXiv preprint arXiv:1701.08718*.

Henson, R. (1998). Short-term memory for serial order: The start-end model. *Cognitive psychology*, *36*(2), 73–137.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Hopfield, J. J., & Tank, D. W. (1986). Computing with neural circuits: A model. *Science*, *233*(4764), 625–633.

Joulin, A., & Mikolov, T. (2015). Inferring algorithmic patterns with stack-augmented recurrent nets. In *Advances in neural information processing systems* (pp. 190–198).

Kingma, D., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kornuta, T., Marois, V., McAvoy, R. L., Bouhadjar, Y., Asseman, A., Albouy, V., ... Ozcan, A. S. (2018). Accelerating machine learning research with mi-prometheus. In *NeurIPS 2018 MLOSS Workshop.*

Lemaire, B., & Portrat, S. (2018). A computational model of working memory integrating time-based decay and interference. *Frontiers in psychology*, *9*, 416.

Maxcey-Richard, A. M., & Hollingworth, A. (2013). The strategic retention of task-relevant objects in visual working memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, *39*(3), 760.

McNab, F., & Klingberg, T. (2008). Prefrontal cortex and basal ganglia control access to working memory. *Nature Neuroscience*, *11*(1), 103–107.

Oberauer, K. (2009). *Chapter 2: Design for a working memory* (1st ed.). Elsevier.

Oberauer, K., & Lewandowsky, S. (2011). Modeling working memory: A computational implementation of the time-based resource-sharing theory. *Psychonomic Bulletin & Review*, *18*(1), 10–45.

Oberauer, K., Lewandowsky, S., Farrell, S., Jarrold, C., & Greaves, M. (2012). Modeling working memory: An interference model of complex span. *Psychonomic bulletin & review*, *19*(5), 779–819.

Oberauer, K., & Lin, H.-y. (2017). An interference model of visual working memory. *Psychological Review*, *124*(1), 1–39. doi: 10.1037/rev0000044

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., ... Lerer, A. (2017). Automatic differentiation in pytorch. In *NIPS Autodiff Workshop.*

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., & Lillicrap, T. (2016). Meta-learning with memory-augmented neural networks. In *International conference on machine learning* (pp. 1842–1850).

Singh, I., Tiganj, Z., & Howard, M. (2018). Is working memory stored along a logarithmic timeline? converging evidence from neuroscience, behavior and models. *Neurobiology of learning and memory*.

Vogel, E. K., McCollough, A. W., & Machizawa, M. G. (2005). Neural measures reveal individual differences in controlling access to working memory. *Nature*, *438*(7067), 500.

Weston, J., Bordes, A., Chopra, S., & Mikolov, T. (2015). Towards ai-complete question answering: A set of prerequisite toy tasks. *CoRR*, *abs/1502.05698*.

Weston, J., Chopra, S., & Bordes, A. (2015). Memory networks. In *International conference on learning representations (ICLR).*

Zaremba, W., Mikolov, T., Joulin, A., & Fergus, R. (2016). Learning simple algorithms from examples. In *International conference on machine learning* (pp. 421–429).