Arman Afrasiyabi

Université Laval

Arman.afrasiyabi.1@ulaval.ca

ARTICLE

doi:10.1038/nature20101

Hybrid computing using a neural network with dynamic external memory

Alex Graves[1]*, Greg Wayne[1]*, Malcolm Reynolds[1], Tim Harley[1], Ivo Danihelka[1], Agnieszka Grabska-Barwińska[1], Sergio Gómez Colmenarejo[1], Edward Grefenstette[1], Tiago Ramalho[1], John Agapiou[1], Adrià Puigdomènech Badia[1], Karl Moritz Hermann[1], Yori Zwols[1], Georg Ostrovski[1], Adam Cain[1], Helen King[1], Christopher Summerfield[1], Phil Blunsom[1], Koray Kavukcuoglu[1] & Demis Hassabis[1]

Artificial neural networks are remarkably adept at sensory processing, sequence learning and reinforcement learning, but are limited in their ability to represent variables and data structures and to store data over long timescales, owing to the lack of an external memory. Here we introduce a machine learning model called a differentiable neural computer (DNC), which consists of a neural network that can read from and write to an external memory matrix, analogous to the random-access memory in a conventional computer. Like a conventional computer, it can use its memory to represent and manipulate complex data structures, but, like a neural network, it can learn to do so from data. When trained with supervised learning, we demonstrate that a DNC can successfully answer synthetic questions designed to emulate reasoning and inference problems in natural language. We show that it can learn tasks such as finding the shortest path between specified points and inferring the missing links in randomly generated graphs, and then generalize these tasks to specific graphs such as transport networks and family trees. When trained with reinforcement learning, a DNC can complete a moving blocks puzzle in which changing goals are specified by sequences of symbols. Taken together, our results demonstrate that DNCs have the capacity to solve complex, structured tasks that are inaccessible to neural networks without external read–write memory.

Modern computers separate computation and memory. Computation is performed by a processor, which can use an addressable memory to bring operands in and out of play. This confers two important benefits: the use of extensible storage to write new information and the ability to treat the contents of memory as variables. Variables are critical to algorithm generality: to perform the same procedure on one datum or another, an algorithm merely has to change the address it reads from. In contrast to computers, the computational and memory resources of artificial neural networks are mixed together in the network weights and neuron activity. This is a major liability: as the memory demands of a task increase, these networks cannot allocate new storage dynamically, nor easily learn algorithms that act independently of the values realized by the task variables.

Although recent breakthroughs demonstrate that neural networks are remarkably adept at sensory processing[1], sequence learning[2,3] and reinforcement learning[4], cognitive scientists and neuroscientists have argued that neural networks are limited in their ability to represent variables and data structures[5–9], and to store data over long timescales without interference[10,11]. We aim to combine the advantages of neural and computational processing by providing a neural network with read–write access to external memory. The access is narrowly focused, minimizing interference among memoranda and enabling long-term storage[12,13]. The whole system is differentiable, and can therefore be trained end-to-end with gradient descent, allowing the network to learn how to operate and organize the memory in a goal-directed manner.

System overview

A DNC is a neural network coupled to an external memory matrix. (The behaviour of the network is independent of the memory size as long as the memory is not filled to capacity, which is why we view the memory as 'external'.) If the memory can be thought of as the DNC's RAM, then the network, referred to as the 'controller', is a differentiable CPU whose operations are learned with gradient descent. The DNC architecture differs from recent neural memory frameworks[14,15] in that the memory can be selectively written to as well as read, allowing iterative modification of memory content. An earlier form of DNC, the neural Turing machine[16], had a similar structure, but more limited memory access methods (see Methods for further discussion).

Whereas conventional computers use unique addresses to access memory contents, a DNC uses differentiable attention mechanisms[2,16–18] to define distributions over the N rows, or 'locations', in the $N \times W$ memory matrix M. These distributions, which we call weightings, represent the degree to which each location is involved in a read or write operation. The read vector $\boldsymbol{r}$ returned by a read weighting $\boldsymbol{w}^r$ over memory M is a weighted sum over the memory locations: $\boldsymbol{r} = \sum_{i=1}^{N} M[i,\cdot]\boldsymbol{w}^r[i]$, where the '$\cdot$' denotes all $j = 1, \ldots, W$. Similarly, the write operation uses a write weighting $\boldsymbol{w}^w$ to first erase with an erase vector $\boldsymbol{e}$, then add a write vector $\boldsymbol{v}$: $M[i,j] \leftarrow M[i,j](1 - \boldsymbol{w}^w[i]\boldsymbol{e}[j]) + \boldsymbol{w}^w[i]\boldsymbol{v}[j]$. The functional units that determine and apply the weightings are called read and write heads. The operation of the heads is illustrated in Fig. 1 and summarized below; see Methods for a formal description.

Interaction between the heads and the memory

The heads use three distinct forms of differentiable attention. The first is content lookup[16,17,19–21], in which a key vector emitted by the controller is compared to the content of each location in memory according to a similarity measure (here, cosine similarity). The similarity scores determine a weighting that can be used by the read heads for associative recall[19] or by the write head to modify an existing vector in memory. Importantly, a key that only partially matches the content of a memory location can still be used to attend strongly to that location.

[1]Google DeepMind, 5 New Street Square, London EC4A 3TW, UK.
*These authors contributed equally to this work.

Title

Differentiable Neural Computer (DNC; 2016)

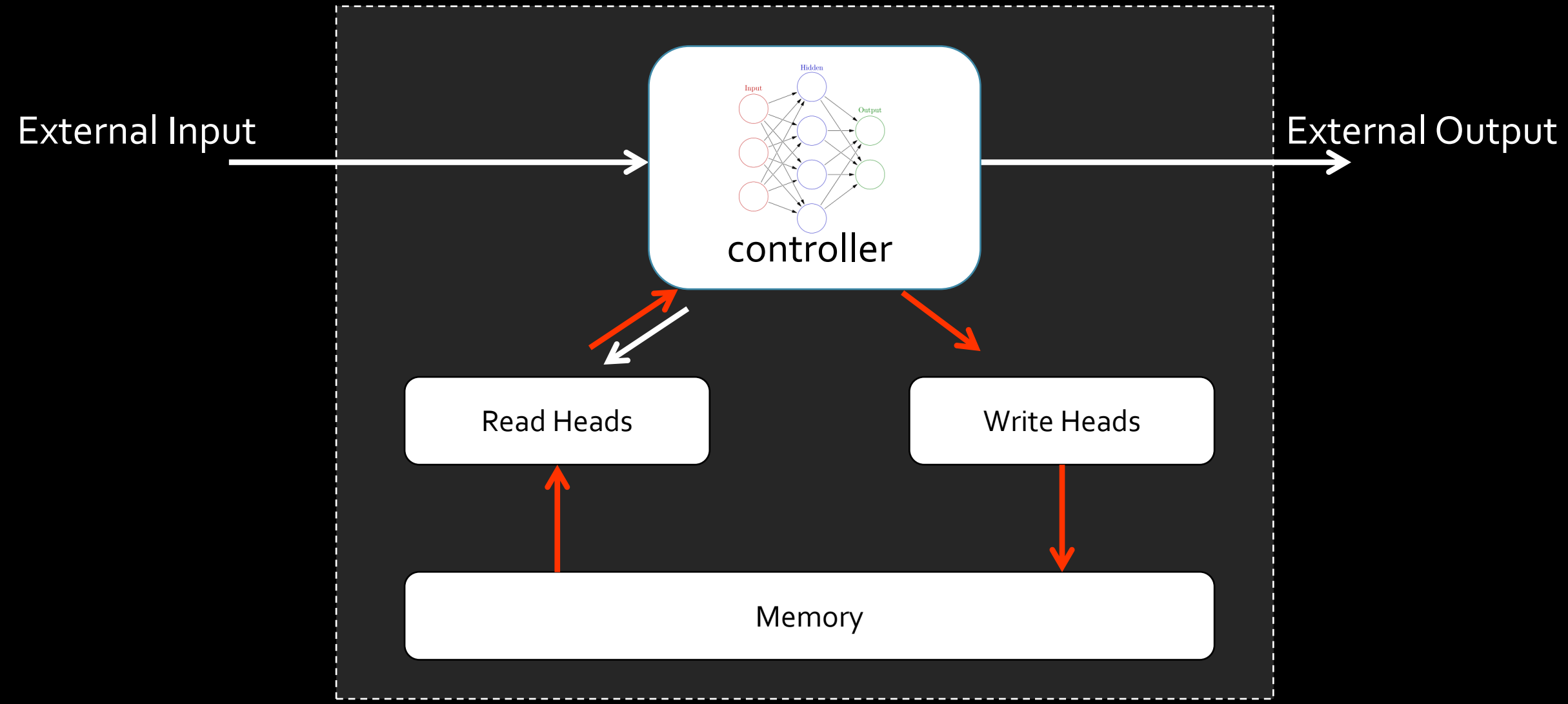
NTM 2.0

Neural Turing Machine (NTM; 2014)

NTM 1.0

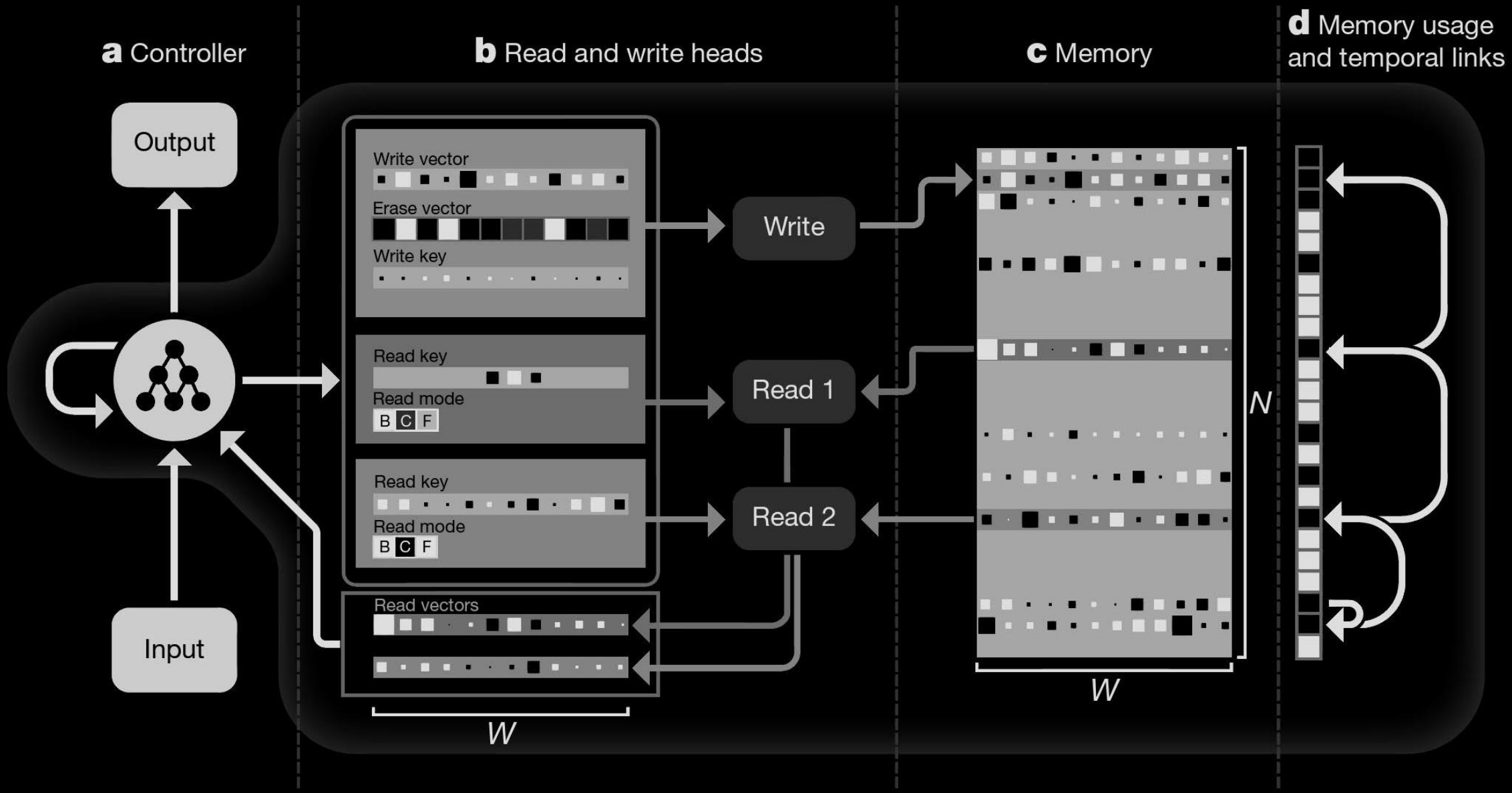
[1] Graves, Alex, et al. "Hybrid computing using a neural network with dynamic external memory." *Nature* 538.7626 (2016): 471-476.
[2] Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." *arXiv preprint arXiv:1410.5401* (2014).

Architecture- Neural Turing Machine (NTM)

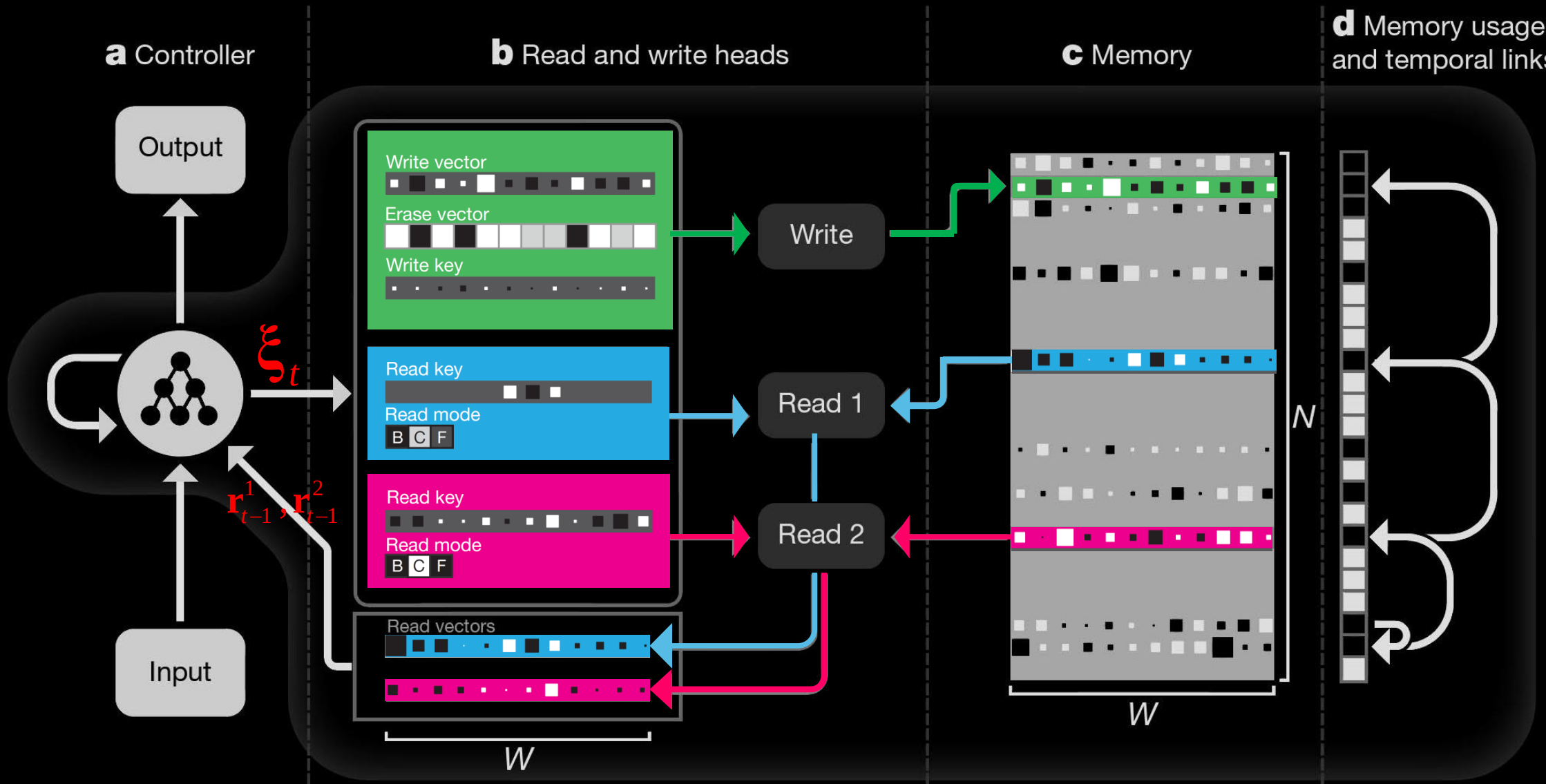


All parts are differentiable!

Architecture- Differentiable Neural Computing (DNC)



Architecture- Differentiable Neural Computing (DNC)



$$\xi_t = \left[\mathbf{k}_t^{\mathbf{r},1};\ldots;\mathbf{k}_t^{\mathbf{r},\mathrm{R}};\hat{\beta}_t^{\mathbf{r},1};\ldots;\hat{\beta}_t^{\mathbf{r},\mathrm{R}};\mathbf{k}_t^{w};\hat{\beta}_t^{w};\hat{\mathbf{e}}_t;\mathbf{v}_t;\hat{\mathbf{f}}_t^{1};\ldots;\hat{\mathbf{f}}_t^{R};\hat{\mathbf{g}}_t^{a};\hat{\mathbf{g}}_t^{w};\hat{\pi}_t^{1};\ldots;\hat{\pi}_t^{R}\right]$$

The Controller $\mathcal{N}$

The controller can be any neural network architecture (in this case a deep LSTM)

Input:

$$\mathrm{X}_t = [x_t;\ \mathbf{r}_{t-1}^1\ ;\ldots;\ \mathbf{r}_{t-1}^R]$$

Controller:

$$(\mathbf{v}_t, \boldsymbol{\xi}_t) = \mathcal{N}\left([\mathrm{X}_1;\ldots,\mathrm{X}_t];\theta\right)$$

Output:

$$y_t = v_t + \mathrm{W}_r\left[r_t^1,\ldots,r_t^R\right]$$

The controller $\mathcal{N}$ (LSTM)

In each layer, $l \in \{1, \ldots, \mathrm{L}\}$ we compute the following functions:

$$i_t^l = \sigma(\mathrm{W}_i^l[\mathrm{X}_t; \mathrm{h}_{t-1}^l; \mathrm{h}_t^{l-1}] + \mathrm{b}_i^l)$$

$$f_t^l = \sigma(\mathrm{W}_f^l[\mathrm{X}_t; \mathrm{h}_{t-1}^l; \mathrm{h}_t^{l-1}] + \mathrm{b}_f^l)$$

$$s_t^l = f_t^l s_{t-1}^l + i_t^l \tanh\left(\mathrm{W}_s^l[\mathrm{X}_t; \mathrm{h}_{t-1}^l; \mathrm{h}_t^{l-1}] + \mathrm{b}_s^l\right)$$

$$o_t^l = \sigma(\mathrm{W}_o^l[\mathrm{X}_t; \mathrm{h}_{t-1}^l; \mathrm{h}_t^{l-1}] + \mathrm{b}_o^l)$$

$$\mathrm{h}_t^l = o_t^l \tanh(s_t^l)$$

output vector, inference vector

$$v_t = \mathrm{W}_y\left[h_t^1; \ldots; h_t^L\right]$$

$$\xi_t = \mathrm{W}_\xi\left[h_t^1; \ldots; h_t^L\right]$$

Interface Vector

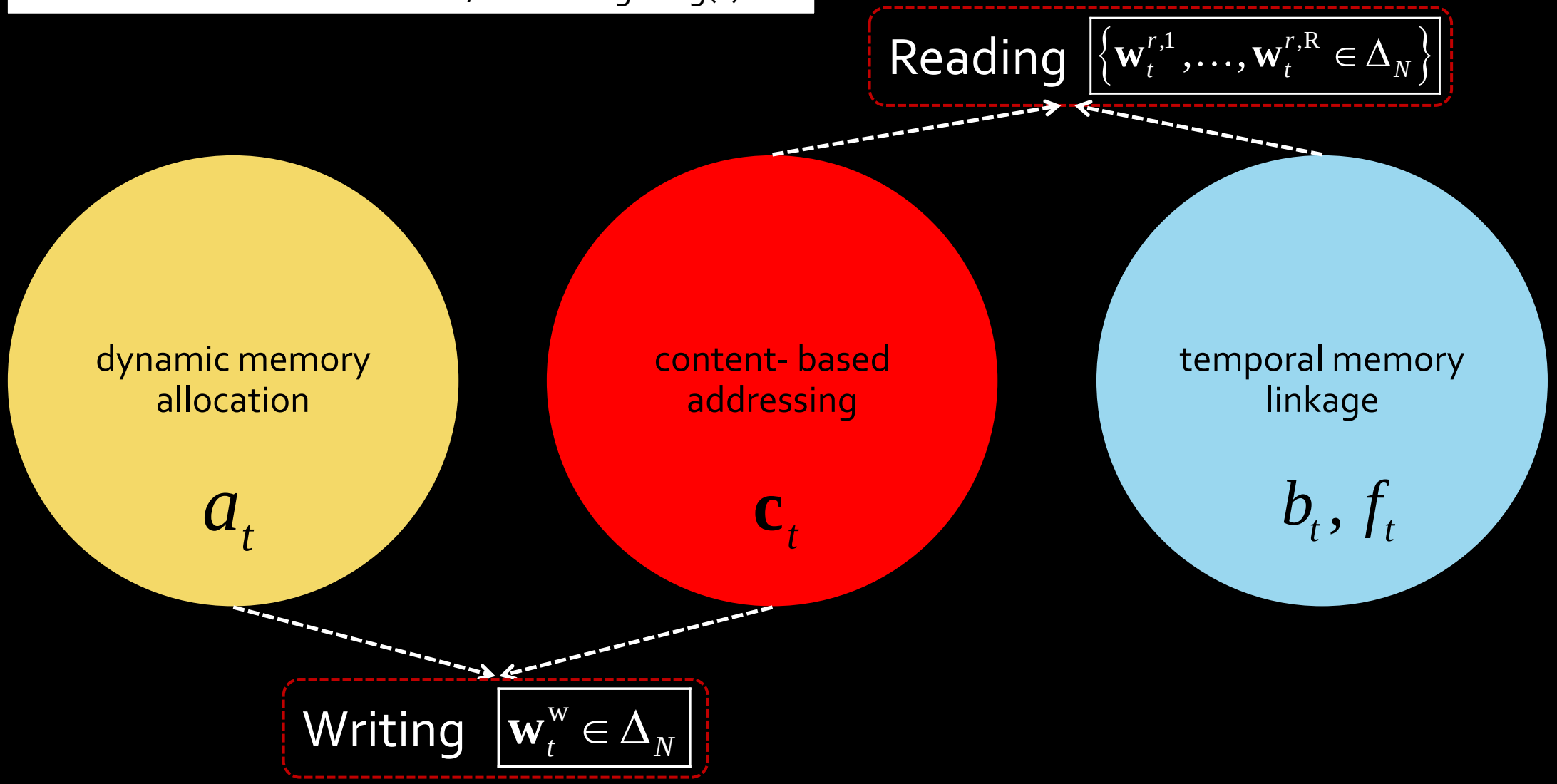
- The interface vector contains parameters
- Controller uses interface vector to connect to memory for Reading/Writing

$$\xi_t = \left[\mathbf{k}_t^{\mathbf{r},1};\ldots;\mathbf{k}_t^{\mathbf{r},\mathrm{R}};\hat{\beta}_t^{\mathbf{r},1};\ldots;\hat{\beta}_t^{\mathbf{r},\mathrm{R}};\mathbf{k}_t^{w};\hat{\beta}_t^{w};\hat{\mathbf{e}}_t;\mathbf{v}_t;\hat{\mathrm{f}}_t^{1};\ldots;\hat{\mathrm{f}}_t^{R};\hat{\mathrm{g}}_t^{a};\hat{\mathrm{g}}_t^{w};\hat{\pi}_t^{1};\ldots;\hat{\pi}_t^{R} \right]$$

location selection for reading/writing depends on weighting(s)

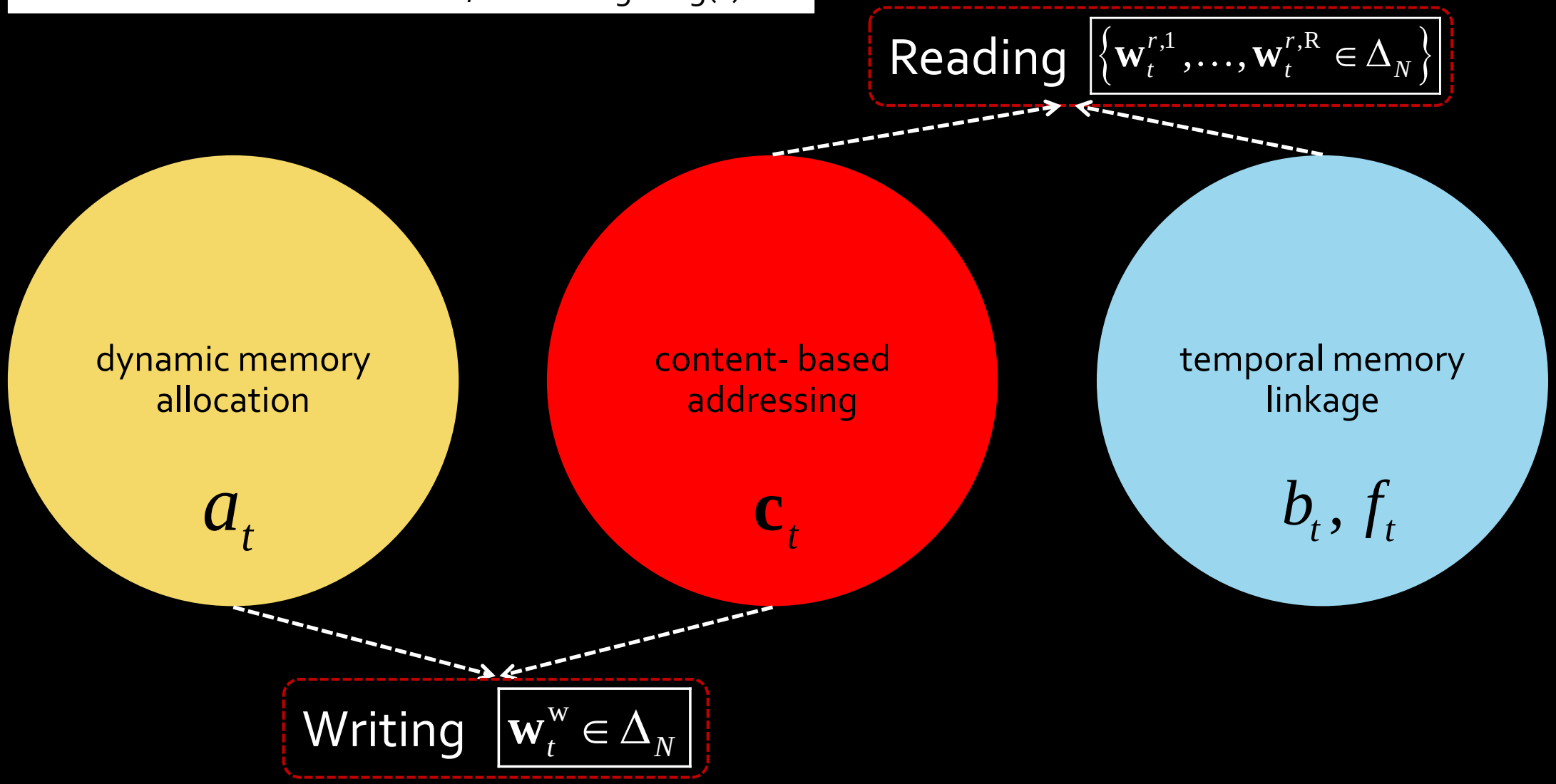
$$\Delta_N = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^N : \boldsymbol{\alpha}_i \in [0,1], \sum_{i=1}^{N} \boldsymbol{\alpha}_i \leq 1 \right\}$$

Memory | Attention mechanisms for Read/Write Weighting(s)



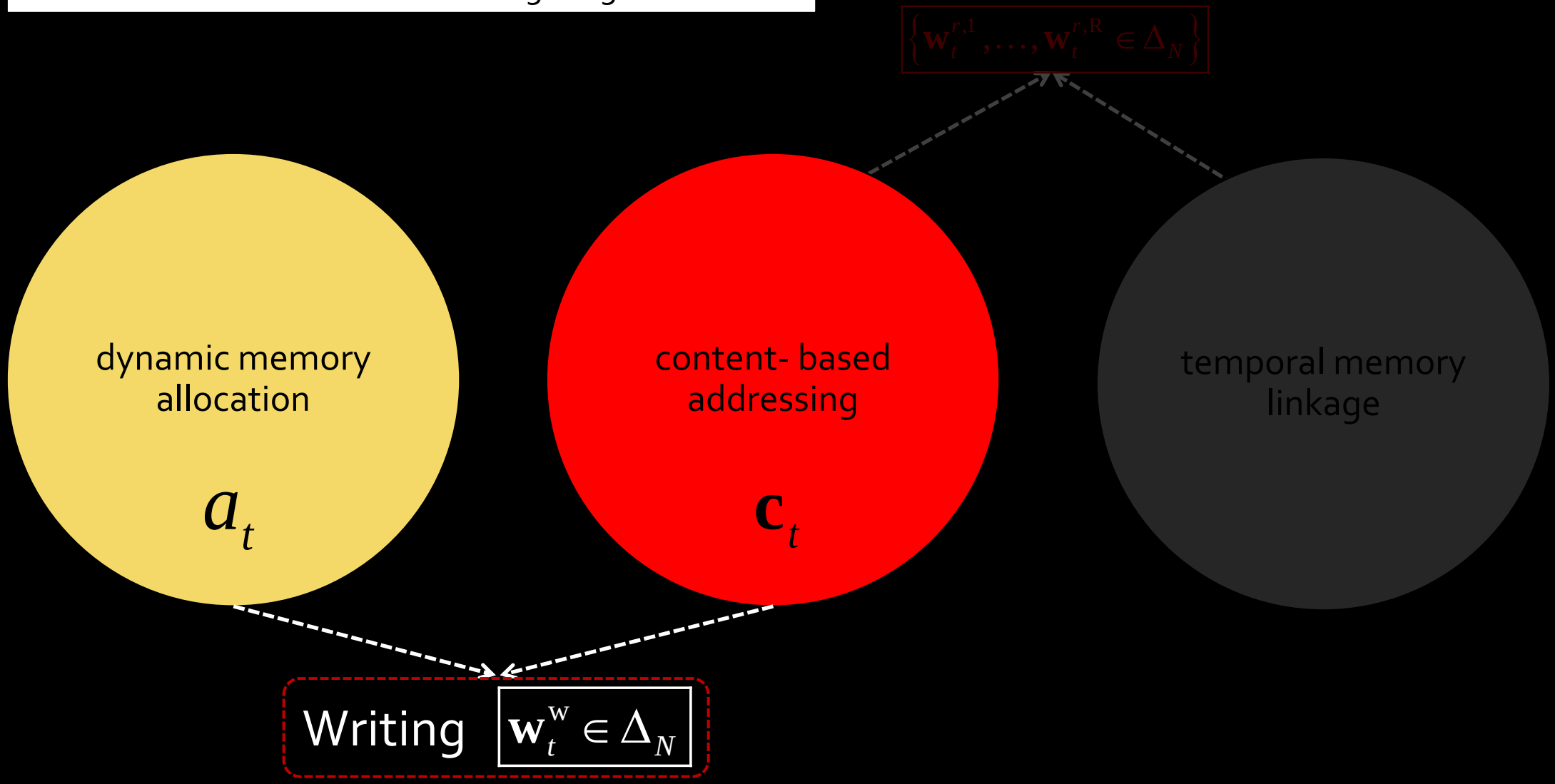
The controller interpolates among these mechanisms using scalar gates

Memory | Attention mechanisms for Read/Write Weighting(s)



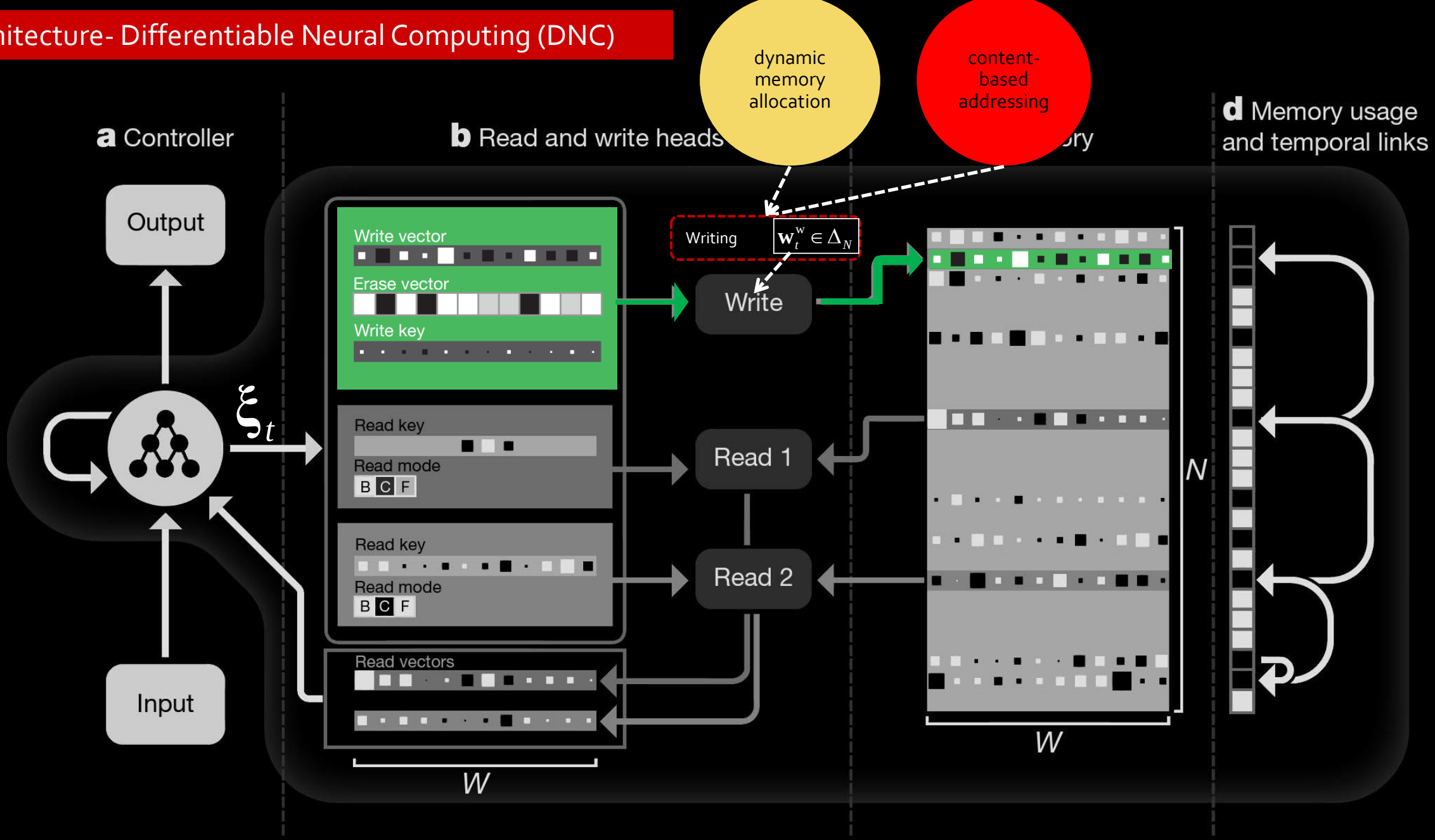
$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathrm{g}}_t^a; \hat{\mathrm{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},\mathrm{R}}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},\mathrm{R}}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

Memory | Attention mechanisms for write weighting



$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{r,1}; \ldots; \mathbf{k}_t^{r,R}; \hat{\beta}_t^{r,1}; \ldots; \hat{\beta}_t^{r,R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

Architecture- Differentiable Neural Computing (DNC)



Memory: Writing

$$E = ones(N \times W)$$

$$\Delta_N = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^N : \boldsymbol{\alpha}_i \in [0,1], \sum_{i=1}^{N} \boldsymbol{\alpha}_i \leq 1 \right\}$$

In writing, location selection depends on single weighting

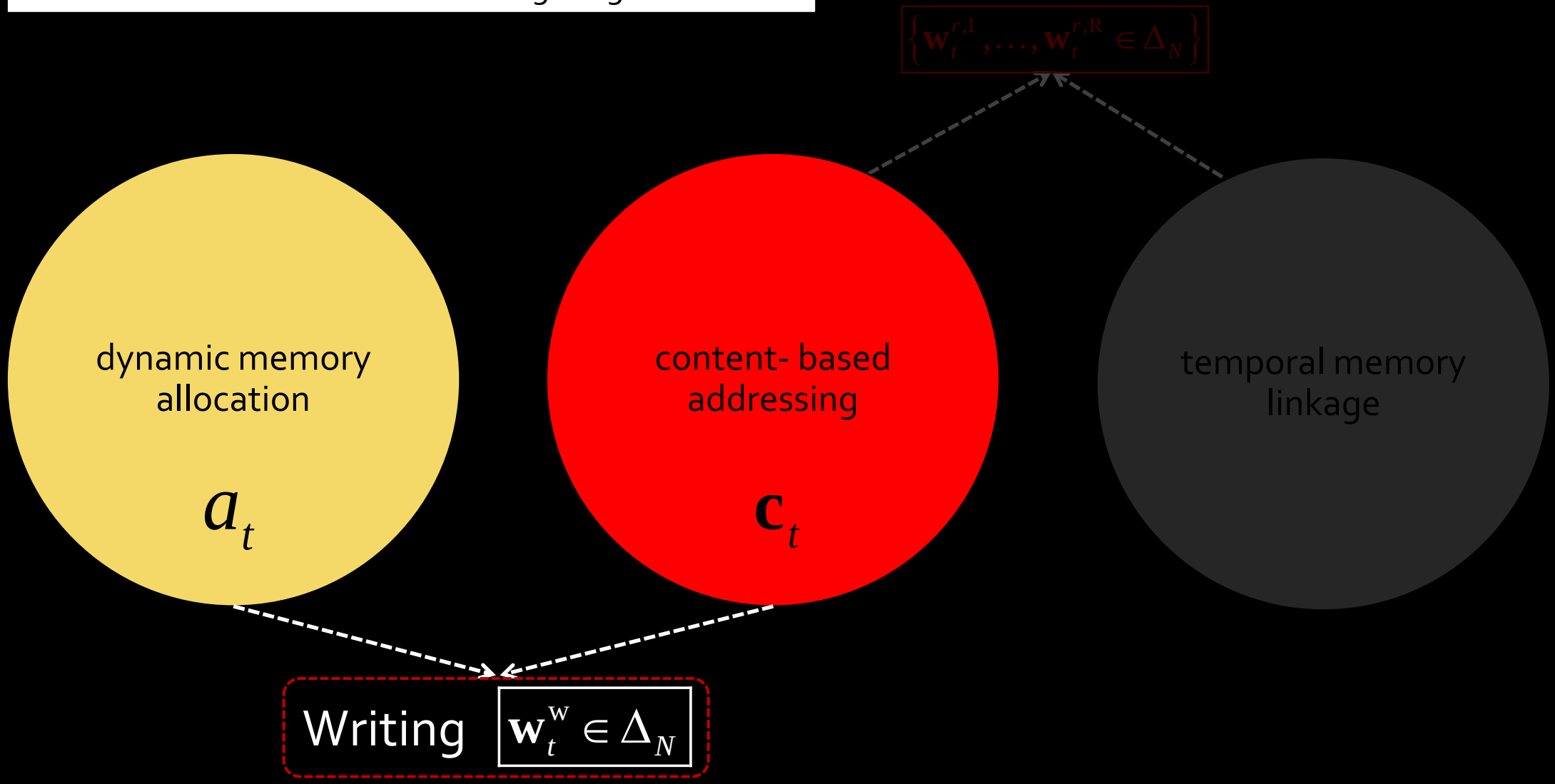
$$\mathbf{w}_t^w \in \Delta_N$$

Now, we can write the write vector $\mathbf{v}_t \in \mathbb{R}^W$ (after erase)

$$M_t = M_{t-1} \circ \left(E - \mathbf{w}_t^w \mathbf{e}_t^T \right) + \mathbf{w}_t^w \mathbf{v}_t^T$$

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},R}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

Memory | Attention mechanisms for Write Weighting



$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{r,1}; \ldots; \mathbf{k}_t^{r,R}; \hat{\beta}_t^{r,1}; \ldots; \hat{\beta}_t^{r,R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

content-based addressing

content lookup operations on the memory $M \in \mathbb{R}^{N \times M}$

$$C(M, \mathbf{k}, \beta)[i] = \frac{\exp\{D(\mathbf{k}, M[\mathrm{i},.])\beta\}}{\sum_j \exp\{D(\mathbf{k}, M[\mathrm{j},.])\beta\}}$$

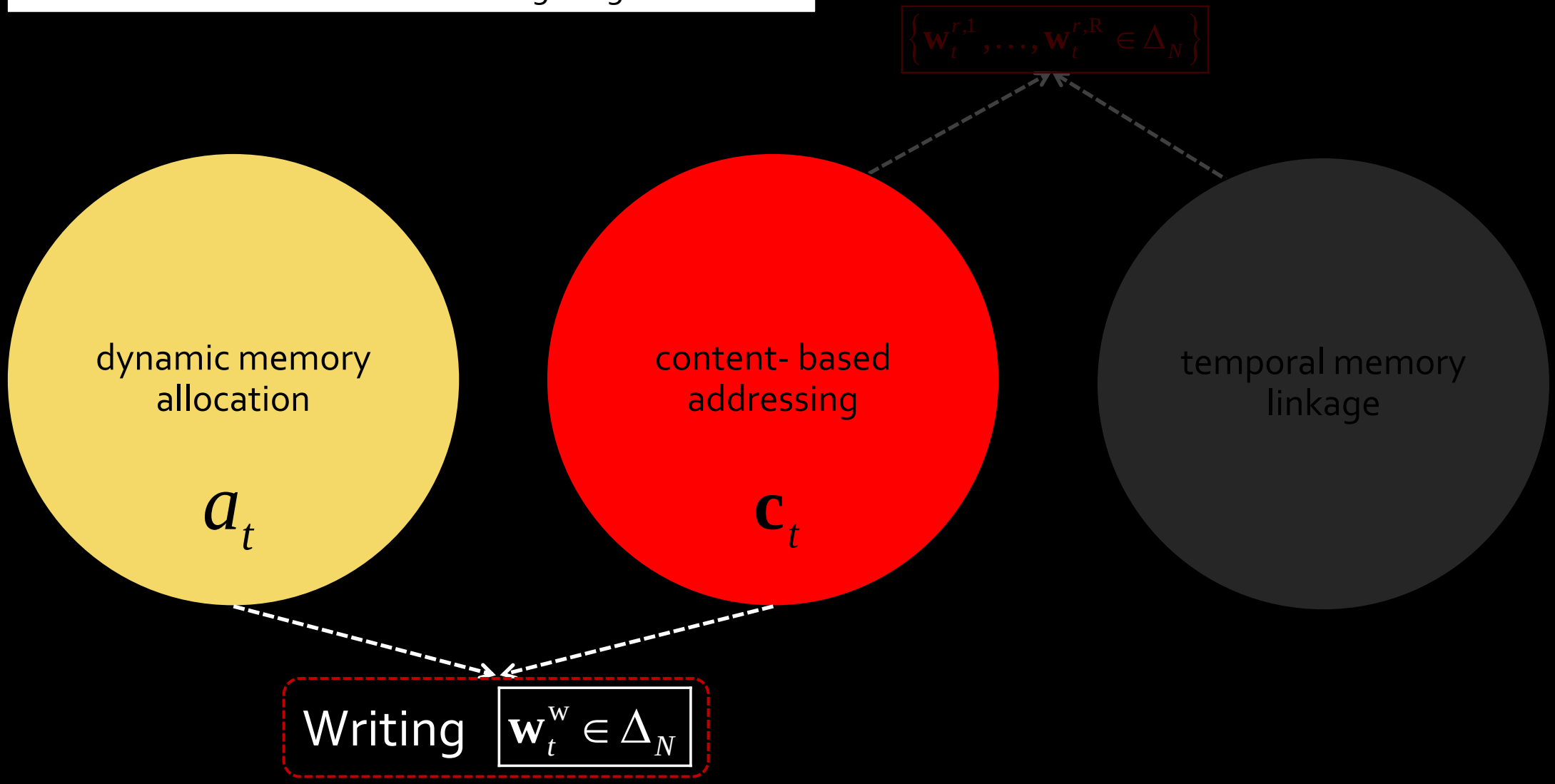
key strength (scalar)

$$D(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}.\mathbf{v}}{|\mathbf{u}||\mathbf{v}|}$$

content weighting $\mathbf{c}_t^{\mathrm{W}} = C\left(M_{t-1}, \mathbf{k}_t^{\mathrm{W}}, \beta_t^{\mathrm{W}}\right)$

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathrm{g}}_t^a; \hat{\mathrm{g}}_t^w; \mathbf{k}_t^{\mathrm{r},1}; \ldots; \mathbf{k}_t^{\mathrm{r},R}; \hat{\beta}_t^{\mathrm{r},1}; \ldots; \hat{\beta}_t^{\mathrm{r},R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R\right]$$

Memory | Attention mechanisms for Write Weighting



$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{r,1}; \ldots; \mathbf{k}_t^{r,R}; \hat{\beta}_t^{r,1}; \ldots; \hat{\beta}_t^{r,R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

dynamic memory allocation

dynamic memory allocation

- The controller should have a mechanism to **free** and **allocate** memory-when needed
- The controller emits free gates $\hat{f}_t^1;\ldots;\hat{f}_t^R$ (one per read head)
- can the most recently read locations be freed ?

Aim of dynamic memory allocation:

finding allocation weightings a_t (history based)

(used to provide new locations for writing)

$$\xi_t = \left[\mathbf{v}_t;\hat{\mathbf{e}}_t;\hat{f}_t^1;\ldots;\hat{f}_t^R;\mathbf{k}_t^w;\hat{\beta}_t^w;\hat{\mathbf{g}}_t^a;\hat{\mathbf{g}}_t^w;\mathbf{k}_t^{\mathbf{r},1};\ldots;\mathbf{k}_t^{\mathbf{r},R};\hat{\beta}_t^{\mathbf{r},1};\ldots;\hat{\beta}_t^{\mathbf{r},R};\hat{\pi}_t^1;\ldots;\hat{\pi}_t^R\right]$$

dynamic memory allocation

dynamic memory allocation a_t

- free-gates determines if the most recently read locations can be freed
  if high free gate, so DCN can forget those memories and write new information to those locations

I) free-gates are used to define memory retention vector

II) **Usage vector**, then, is defined as follow

differentiable free list

$$\mathbf{u}_t = \left(\mathbf{u}_{t-1} + \mathbf{w}^{w}_{t-1} - \mathbf{u}_{t-1} \circ \mathbf{w}^{w}_{t-1}\right) \circ \underbrace{\prod_{i=1}^{R}\left(\mathbf{1} - f_t^{\,i}\mathbf{w}^{r,i}_{t-1}\right)}_{\psi_t}$$

If for a memory cell $\psi_t[\mathrm{i}] \approx 0$, then it have been recently read

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^{1}; \ldots; \hat{f}_t^{R}; \mathbf{k}^{w}_t; \hat{\beta}^{w}_t; \hat{\mathbf{g}}^{a}_t; \hat{\mathbf{g}}^{w}_t; \mathbf{k}^{\mathbf{r},1}_t; \ldots; \mathbf{k}^{\mathbf{r},R}_t; \hat{\beta}^{\mathbf{r},1}_t; \ldots; \hat{\beta}^{\mathbf{r},R}_t; \hat{\pi}^{1}_t; \ldots; \hat{\pi}^{R}_t\right]$$

dynamic memory allocation

dynamic memory allocation a_t

- free-gates determines if the most recently read locations can be freed
  if high free gate, so DCN can forget those memories and write new information to those locations

I) free-gates are used to define memory retention vector

II) **Usage vector**, then, is defined as follow

differentiable free list

$$\mathbf{u}_t = \left(\mathbf{u}_{t-1} + \mathbf{w}^{w}_{t-1} - \mathbf{u}_{t-1} \circ \mathbf{w}^{w}_{t-1}\right) \circ \prod_{i=1}^{R} \left(1 - f_t^i \mathbf{w}^{r,i}_{t-1}\right)$$

if DNC has not recently write to

ψ_t

if DNC has recently read from

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{r,1}; \ldots; \mathbf{k}_t^{r,R}; \hat{\beta}_t^{r,1}; \ldots; \hat{\beta}_t^{r,R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R\right]$$

dynamic memory allocation

dynamic memory allocation a_t

- Memory usage vector tracks the usage of each memory location

$$\mathbf{u}_t = \left(\mathbf{u}_{t-1} + \mathbf{w}_{t-1}^{w} - \mathbf{u}_{t-1} \circ \mathbf{w}_{t-1}^{w}\right) \circ \underbrace{\prod_{i=1}^{R}\left(\mathbf{1} - f_t^i \mathbf{w}_{t-1}^{r,i}\right)}_{\psi_t}$$

$$\boxed{\mathbf{u}_t \in [0, 1]^N}$$

$$\phi_t = SortIndicesAscending\left(\mathbf{u}_t\right)$$

$\phi_t[1]$ is the "least used" index of $\mathbf{u}_t$

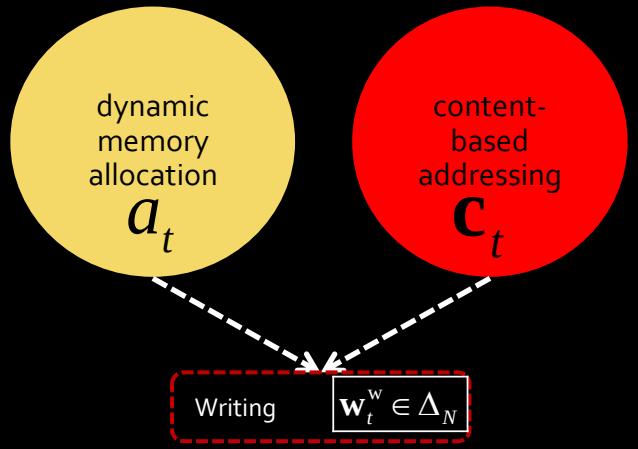
- Allocation weightings: used to provide new locations for the writing

$$a_t\left[\phi_t[\mathrm{j}]\right] = \left(1 - \left[\phi_t[\mathrm{j}]\right]\right) \prod_{\mathrm{j}=1}^{\mathrm{j-1}} \mathbf{u}_t[\mathrm{i}]$$

If usages are 1, then $a_t = 0$ and the controller can not longer allocate memory

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{r,1}; \ldots; \mathbf{k}_t^{r,R}; \hat{\beta}_t^{r,1}; \ldots; \hat{\beta}_t^{r,R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R\right]$$

Writing on Memory and Weightings



$$M_t = M_{t-1} \circ \left(E - \mathbf{w}_t^w \mathbf{e}_t^T \right) + \mathbf{w}_t^w \mathbf{v}_t^T$$

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},R}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

Writing on Memory and Weightings

allocate weighting $a_t[\phi_t[j]] = (1 - [\phi_t[j]]) \prod_{i=1}^{j-1} \mathbf{u}_t[i]$

content weighting $\mathbf{c}_t^{W} = C\left(M_{t-1}, \mathbf{k}_t^{W}, \beta_t^{W}\right)$

$$M_t = M_{t-1} \circ \left(E - \mathbf{w}_t^{w} \mathbf{e}_t^{T}\right) + \mathbf{w}_t^{w} \mathbf{v}_t^{T}$$

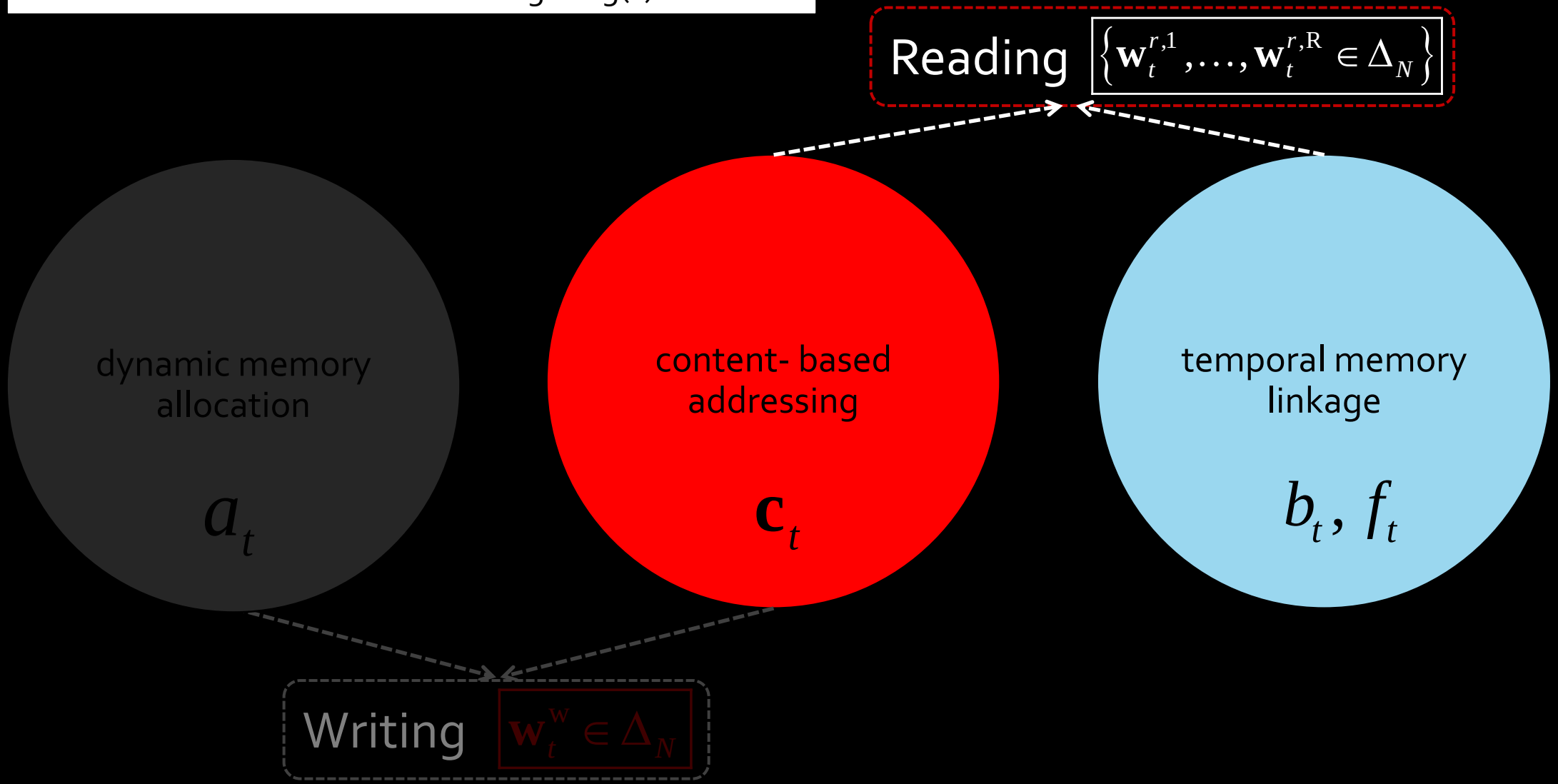
$$\mathbf{w}_t^{w} = g_t^{w}\left[g_t^{a} a_t + (1 - g_t^{a})\,\mathbf{c}_t^{W}\right]$$

$g_t^{W} \in [0,1]$ is the write gate, if write gate is zero, then nothing is written

$g_t^{a} \in [0,1]$ the allocation gate governing the interpolation

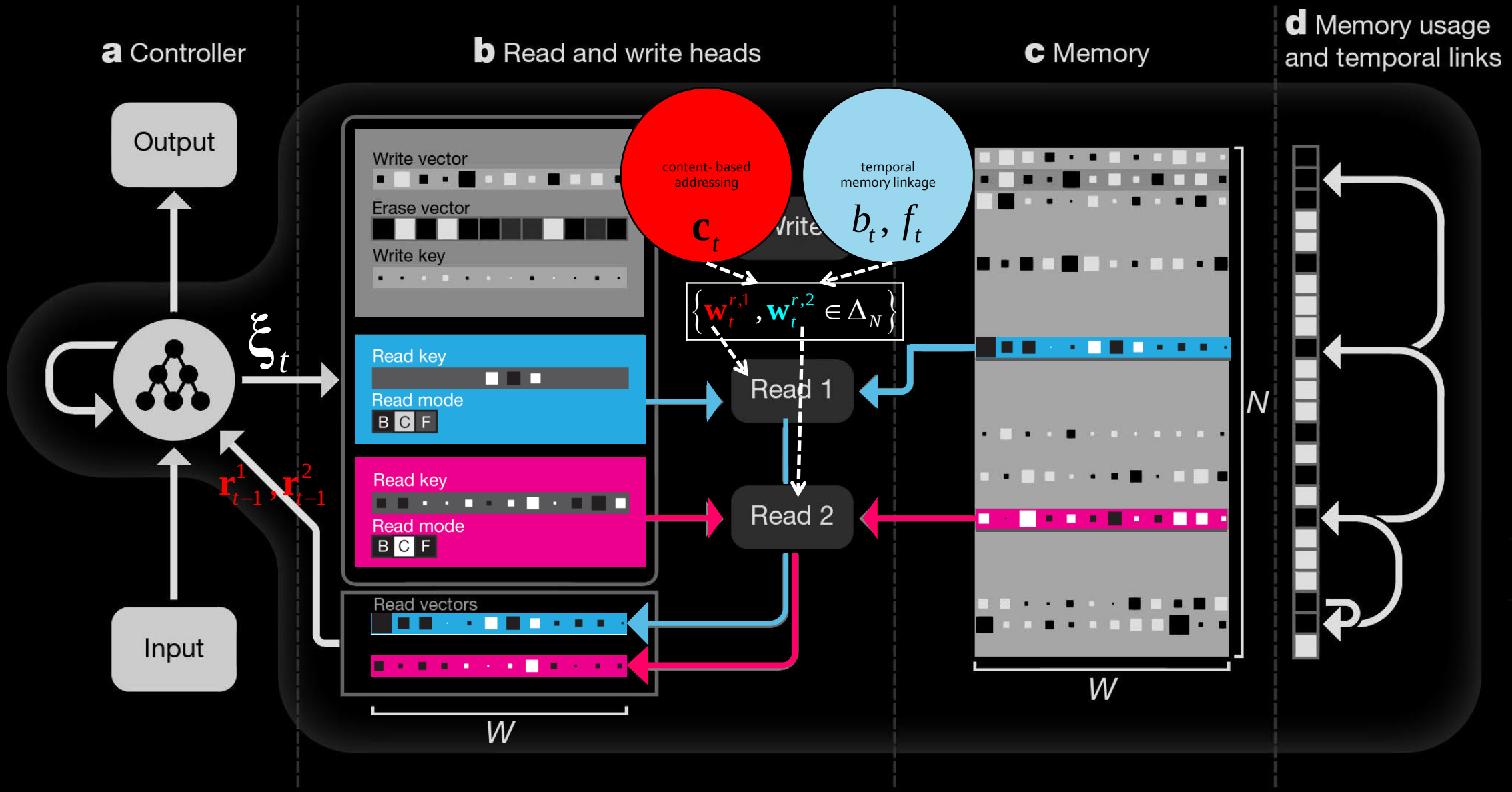
$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^{1}; \ldots; \hat{f}_t^{R}; \mathbf{k}_t^{w}; \hat{\beta}_t^{w}; \hat{\mathbf{g}}_t^{a}; \hat{\mathbf{g}}_t^{w}; \mathbf{k}_t^{r,1}; \ldots; \mathbf{k}_t^{r,R}; \hat{\beta}_t^{r,1}; \ldots; \hat{\beta}_t^{r,R}; \hat{\pi}_t^{1}; \ldots; \hat{\pi}_t^{R}\right]$$

Memory | Attention mechanisms for Read Weighting(s)



$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},R}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

Architecture- Differentiable Neural Computing (DNC)



Memory: Reading

$$\Delta_N = \left\{ \boldsymbol{\alpha} \in \mathbb{R}^N : \boldsymbol{\alpha}_i \in [0,1], \sum_{i=1}^{N} \boldsymbol{\alpha}_i \leq 1 \right\}$$

In reading, location select depends on weightings

$$\left\{ \mathbf{w}_t^{r,1}, \ldots, \mathbf{w}_t^{r,\mathrm{R}} \in \Delta_N \right\}$$

Now, we can define the read vectors $\left\{ \mathbf{r}_t^1, \ldots, \mathbf{r}_t^R \right\}$

$$\mathbf{r}_t^i = M_t^T \mathbf{w}_t^{r,\mathrm{i}}$$

content-based addressing

content lookup operations on the memory $M \in \mathbb{R}^{N \times M}$ use the following function

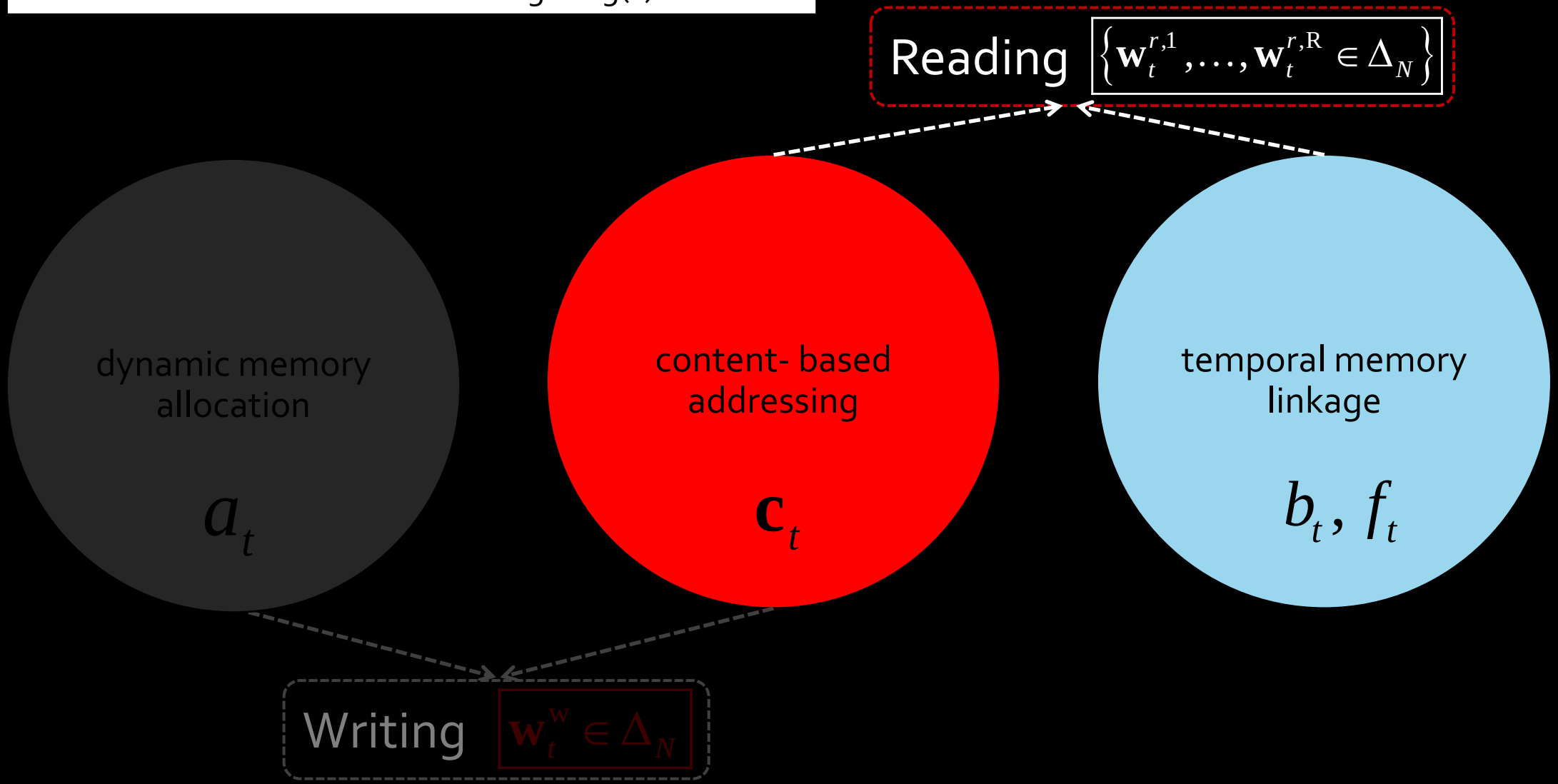
$$C(M, \mathbf{k}, \beta)[i] = \frac{\exp\{D(\mathbf{k}, M[\mathrm{i},.])\beta\}}{\sum_j \exp\{D(\mathbf{k}, M[\mathrm{j},.])\beta\}}$$

$$D(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u}.\mathbf{v}}{|\mathbf{u}||\mathbf{v}|}$$

content weighting $\mathbf{c}_t^{r,i} = C\left(M_{t-1}, \mathbf{k}_t^{r,i}, \beta_t^{r,i}\right)$

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathrm{g}}_t^a; \hat{\mathrm{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},\mathrm{R}}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},\mathrm{R}}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R\right]$$

Memory | Attention mechanisms for Read Weighting(s)



$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathbf{g}}_t^a; \hat{\mathbf{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},R}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},R}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

Temporal Memory Linkage

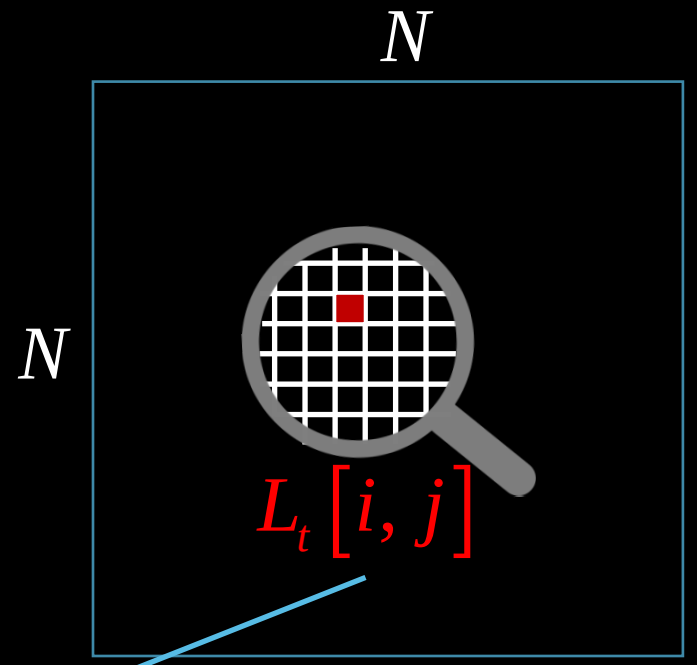
temporal memory linkage

The order of stored/written information is important for reading

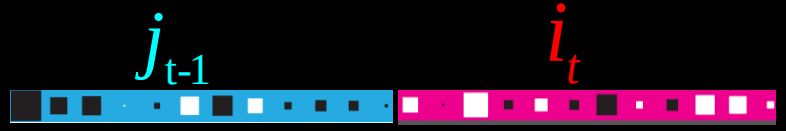
e.g. retrieving instructions

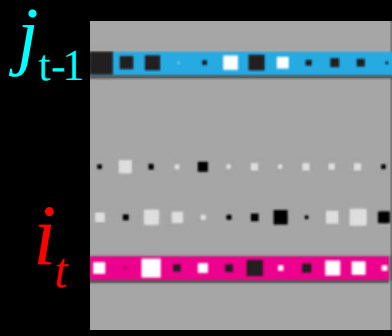
temporal memory linkage

$$L_t \in [0,1]^{N \times N}$$



location i was written to **just after** location j in the pervious time step.





Temporal Memory Linkage

defining $L_t \in [0,1]^{N \times N}$

temporal memory linkage

To define L_t

we need a precedence weighting $\mathbf{p}_t$

$$\mathbf{p}_t = \underbrace{\left(1 - \sum_i \mathbf{w}_t^w[i]\right)\mathbf{p}_{t-1}}_{\text{remove the old linkages}} + \underbrace{\mathbf{w}_t^w}_{\text{adds new linages}}$$

$\mathbf{p}_t[i]$ represents the degree to which location i was the last one written to.

DNC updates $\mathbf{p}_t$ based on how much writing occurred at the current time-step (by $\mathbf{w}_t^w$)

$$\text{if} \quad \mathbf{W}_t^W \approx 0 \quad \Rightarrow \quad \mathbf{p}_t \approx \mathbf{p}_{t-1} \text{ (no writing)}$$

Temporal Memory Linkage

updating $L_t \in [0,1]^{N \times N}$

temporal memory linkage

precedence weighting $\mathbf{p}_t$ is used to update a temporal link matrix

$$L_t[\mathrm{i},\mathrm{j}] = \left(1 - \mathbf{w}_t^w[\mathrm{i}] - \mathbf{w}_t^w[\mathrm{j}]\right) L_{t-1}[\mathrm{i},\mathrm{j}] + \mathbf{w}_t^w[\mathrm{i}]\, \mathbf{p}_{t-1}[\mathrm{j}]$$

tracks the order of writings on the memory locations

the write weight of location i at current time

precedence weigh of location j at pervious

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathrm{g}}_t^a; \hat{\mathrm{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},\mathrm{R}}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},\mathrm{R}}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R\right]$$

temporal memory linkage

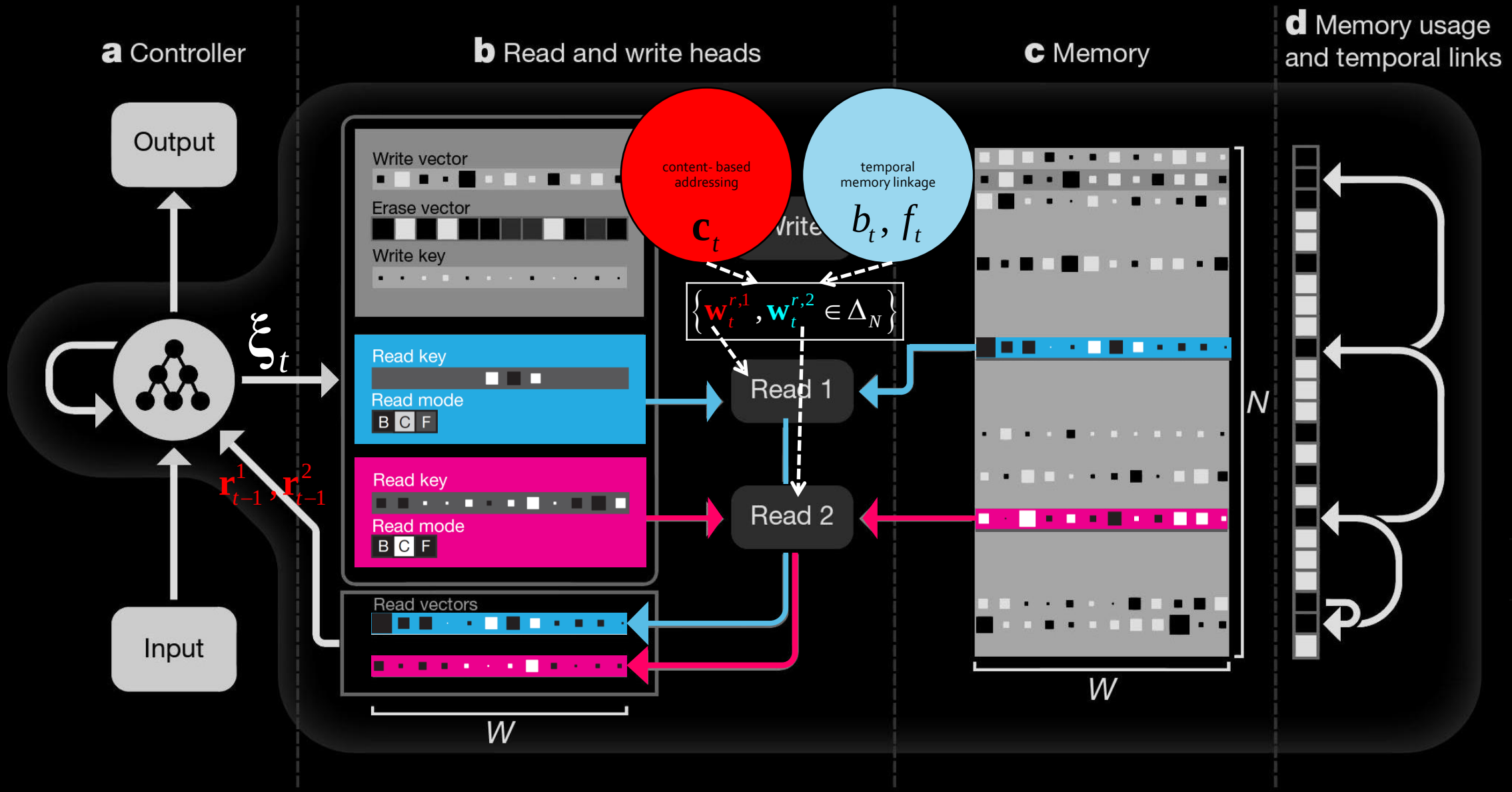
$$L_t[\mathrm{i},\mathrm{j}] = \left(1 - \mathbf{w}_t^w[\mathrm{i}] - \mathbf{w}_t^w[\mathrm{j}]\right) L_{t-1}[\mathrm{i},\mathrm{j}] + \mathbf{w}_t^w[\mathrm{i}]\, \mathbf{p}_{\mathrm{t\text{-}1}}[\mathrm{j}]$$

$L_t[\mathrm{i},\mathrm{j}]$ allowing the controller to iterate forward or backward in time

$$f_t^i = L_t \mathbf{w}_{\mathrm{t\text{-}1}}^{\mathrm{r,i}} \qquad b_t^i = L_t^T \mathbf{w}_{\mathrm{t\text{-}1}}^{\mathrm{r,i}}$$

$$\xi_t = \left[\mathbf{v}_t; \hat{\mathbf{e}}_t; \hat{f}_t^1; \ldots; \hat{f}_t^R; \mathbf{k}_t^w; \hat{\beta}_t^w; \hat{\mathrm{g}}_t^a; \hat{\mathrm{g}}_t^w; \mathbf{k}_t^{\mathbf{r},1}; \ldots; \mathbf{k}_t^{\mathbf{r},\mathrm{R}}; \hat{\beta}_t^{\mathbf{r},1}; \ldots; \hat{\beta}_t^{\mathbf{r},\mathrm{R}}; \hat{\pi}_t^1; \ldots; \hat{\pi}_t^R \right]$$

Architecture- Differentiable Neural Computing (DNC)



Finding read gates

$$\left\{\mathbf{w}_t^{r,1},\ldots,\mathbf{w}_t^{r,R}\in\Delta_N\right\}$$

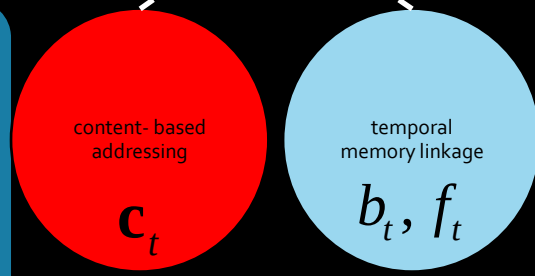
content based addressing

$$\mathbf{c}_t^{r,i}=C\left(M_{t-1},\mathbf{k}_t^{r,i},\beta_t^{r,i}\right)$$

temporal memory linkage

$$L_t[\mathrm{i},\mathrm{j}]=\left(1-\mathbf{w}_t^{w}[\mathrm{i}]-\mathbf{w}_t^{w}[\mathrm{j}]\right)L_{t-1}[\mathrm{i},\mathrm{j}]+\mathbf{w}_t^{w}[\mathrm{i}]\,\mathbf{p}_{t-1}[\mathrm{j}]$$

$$f_t^i=L_t\mathbf{w}_{t-1}^{r,i} \qquad b_t^i=L_t^T\mathbf{w}_{t-1}^{r,i}$$



Reading weight (using three way gates π_t^i)

$$\mathbf{W}_t^{r,i}=\pi_t^i[1]\,\mathbf{b}_t^i+\pi_t^i[2]\,\mathbf{c}_t^{r,i}+\pi_t^i[3]\,f_t^i$$

$$\xi_t=\left[\mathbf{v}_t;\hat{\mathbf{e}}_t;\hat{f}_t^1;\ldots;\hat{f}_t^R;\mathbf{k}_t^w;\hat{\beta}_t^w;\hat{\mathbf{g}}_t^a;\hat{\mathbf{g}}_t^w;\mathbf{k}_t^{\mathbf{r},1};\ldots;\mathbf{k}_t^{\mathbf{r},\mathrm{R}};\hat{\beta}_t^{\mathbf{r},1};\ldots;\hat{\beta}_t^{\mathbf{r},\mathrm{R}};\hat{\pi}_t^1;\ldots;\hat{\pi}_t^R\right]$$

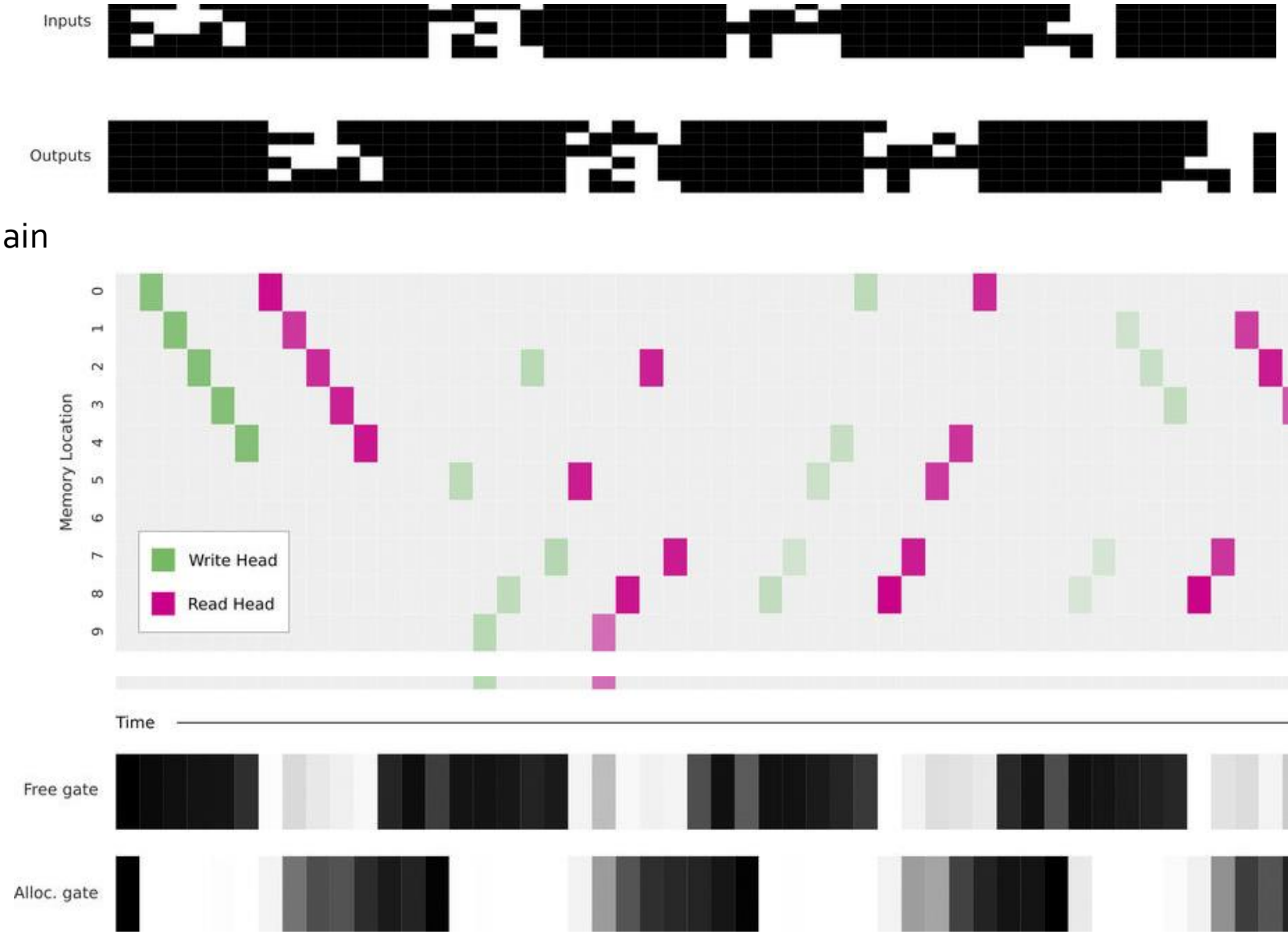
Dynamic Memory Allocation - Test

- present sequences to the network

- controller's aim is to recreate of the input
  If could recreate,
  then that input is not needed again

- many input sequences:
  - but the size of memory is 10;
  - it must erase and write

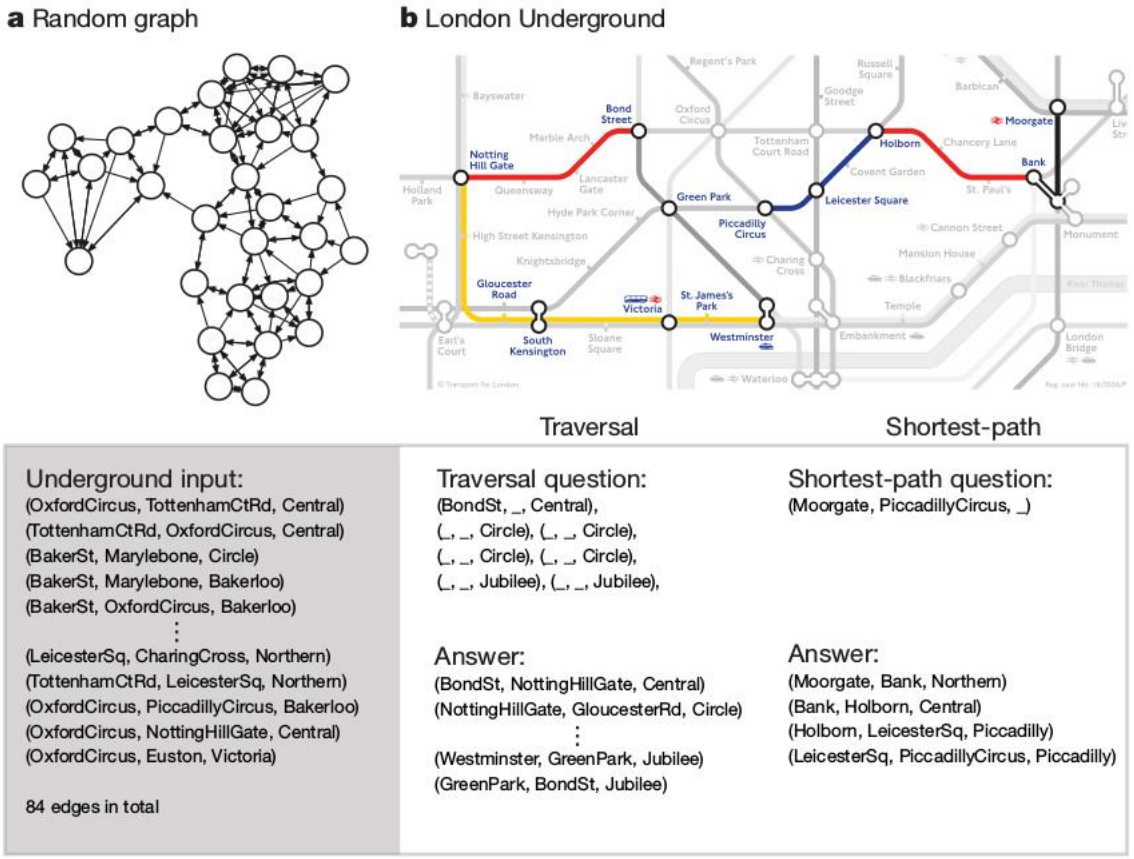
Goal: to test whether the memory allocation system would be used and free and re-use the locations



DNC is tested on answer questions from about the graphs

I) London Underground graph

- DNC asked single navigation questions

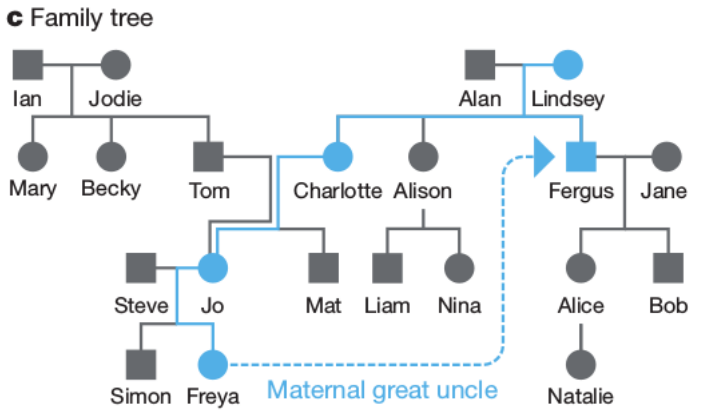


The Goal was not solve the problems in the graphs,

The goals was to show either DNC can create and navigate throughout the graph.

DNC is tested on answer questions from about the graphs

II) Family graph: the relation between people has been asked which has not been shown in the training time.



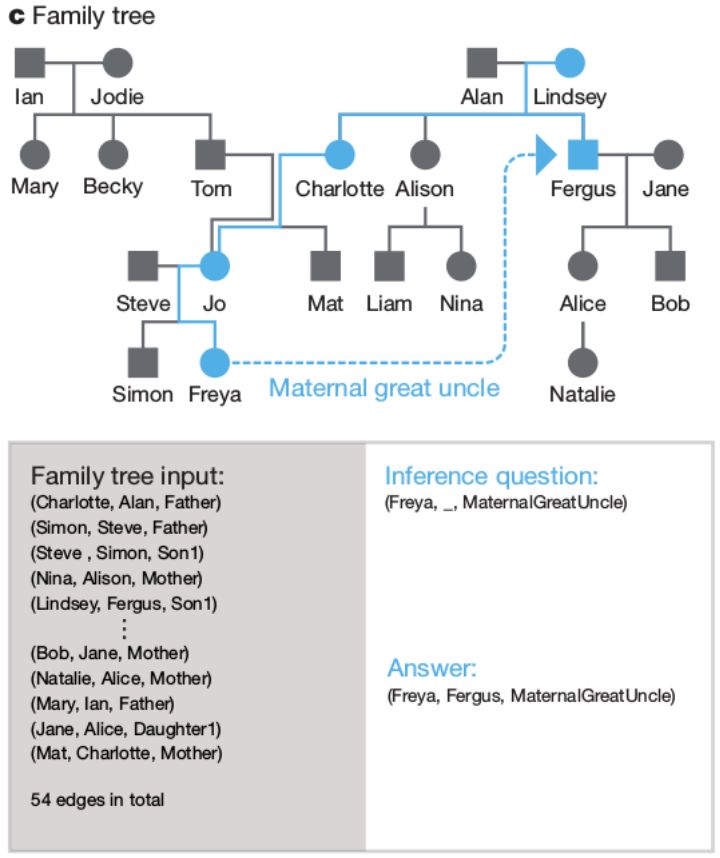
The goal was not solve the problems in the graphs,
The goals was to show either DNC can create and navigate throughout the graph.

DNC- Test on Graphs

DNC is tested on answer questions from about the graphs

II) family graph which the relation between people has been asked which has not been shown in the training time.

https://www.youtube.com/watch?v=B9U8sI7TcMY



The goal was not solve the problems in the graphs,
The goals was to show either DNC can create and navigate throughout the graph.

Thank You!