Hybrid computing using a neural network with dynamic external memory

By Fangyuan Yu, Shuai Lu, Lukang Sun

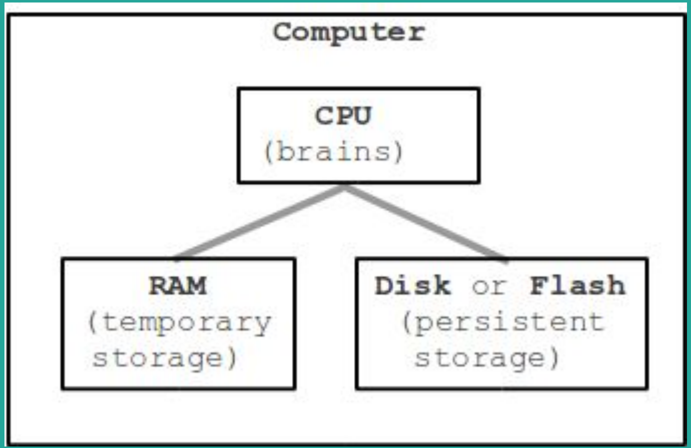
Part I:
Introduction

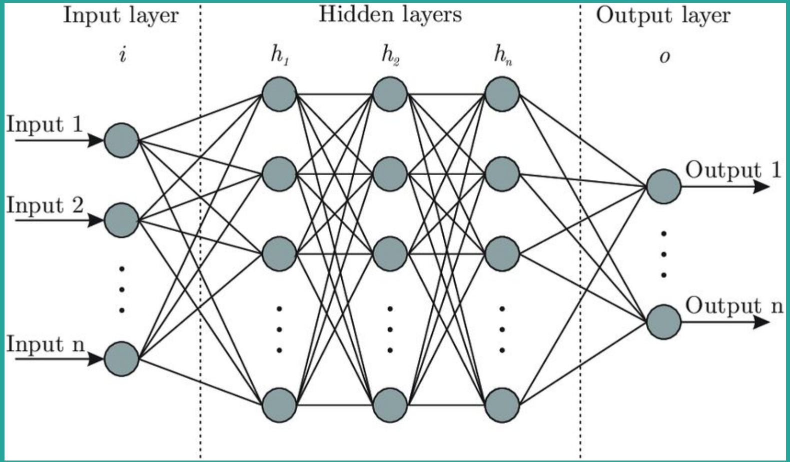
Why one needs memory?

Assume if you had no memory, who are you? Whole is your father/mother?.....

Modern computer: separate memory / computation

Artificial Neural Network: no separation, all in network weights and neuron activity





Types of Neural Network which keeps (explicit) internal memory:

1. RNN - internal state - vanishing gradient
2. LSTM - forget gate - ‘memory highways’
3. GRU - simplified LSTM, less parameter
4. …..

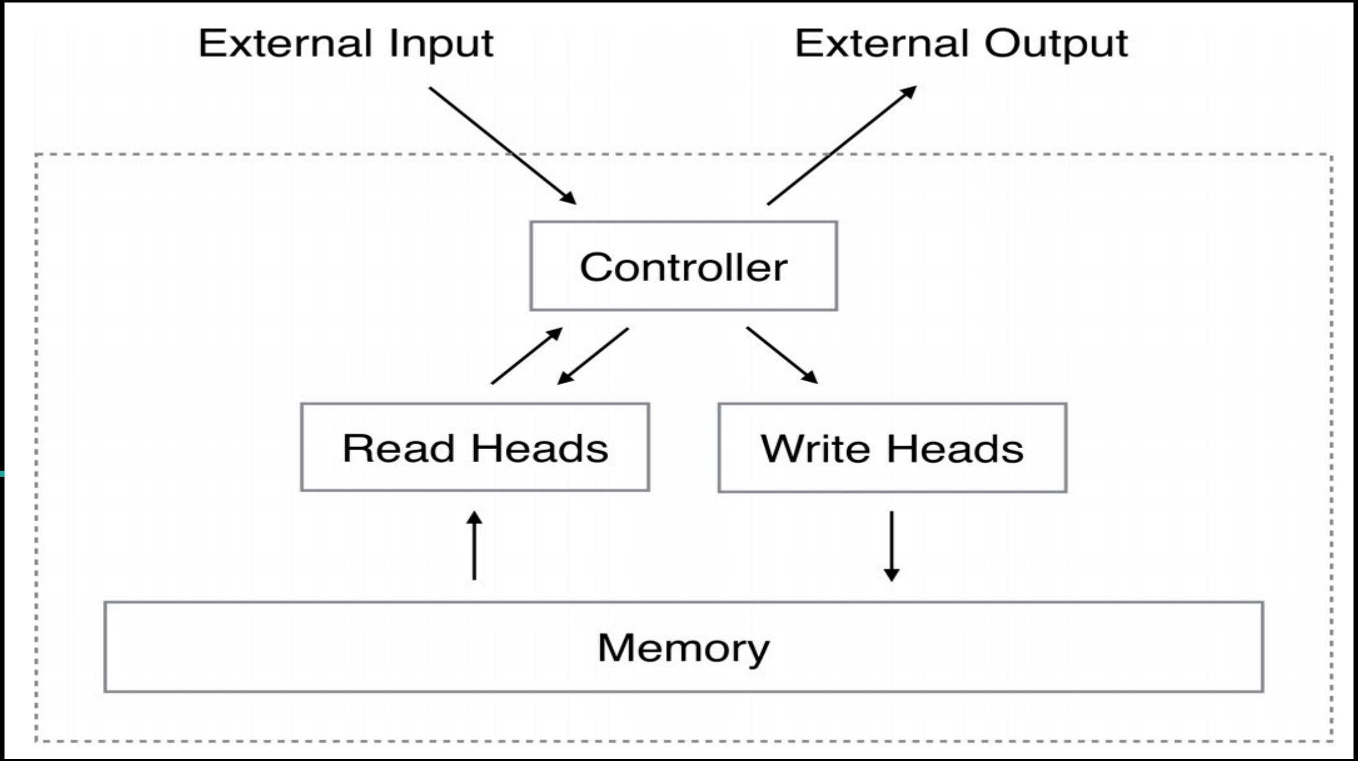
Main Idea: It is beneficial to include ‘external memory’ into artificial neural network?

Part II: Algorithm

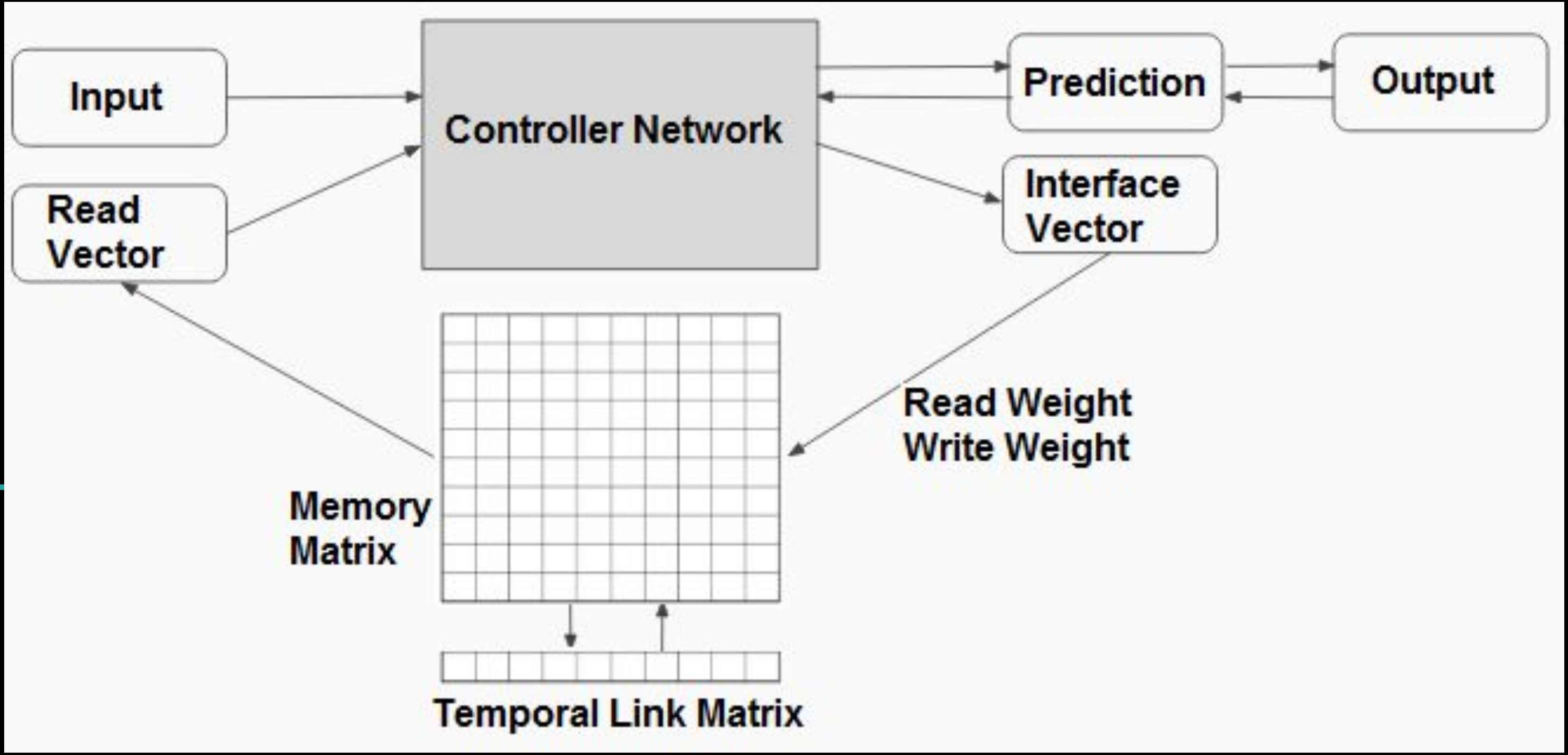
Basic Component:

1. Controller (neural network, trainable)
2. External memory (deterministic structure)

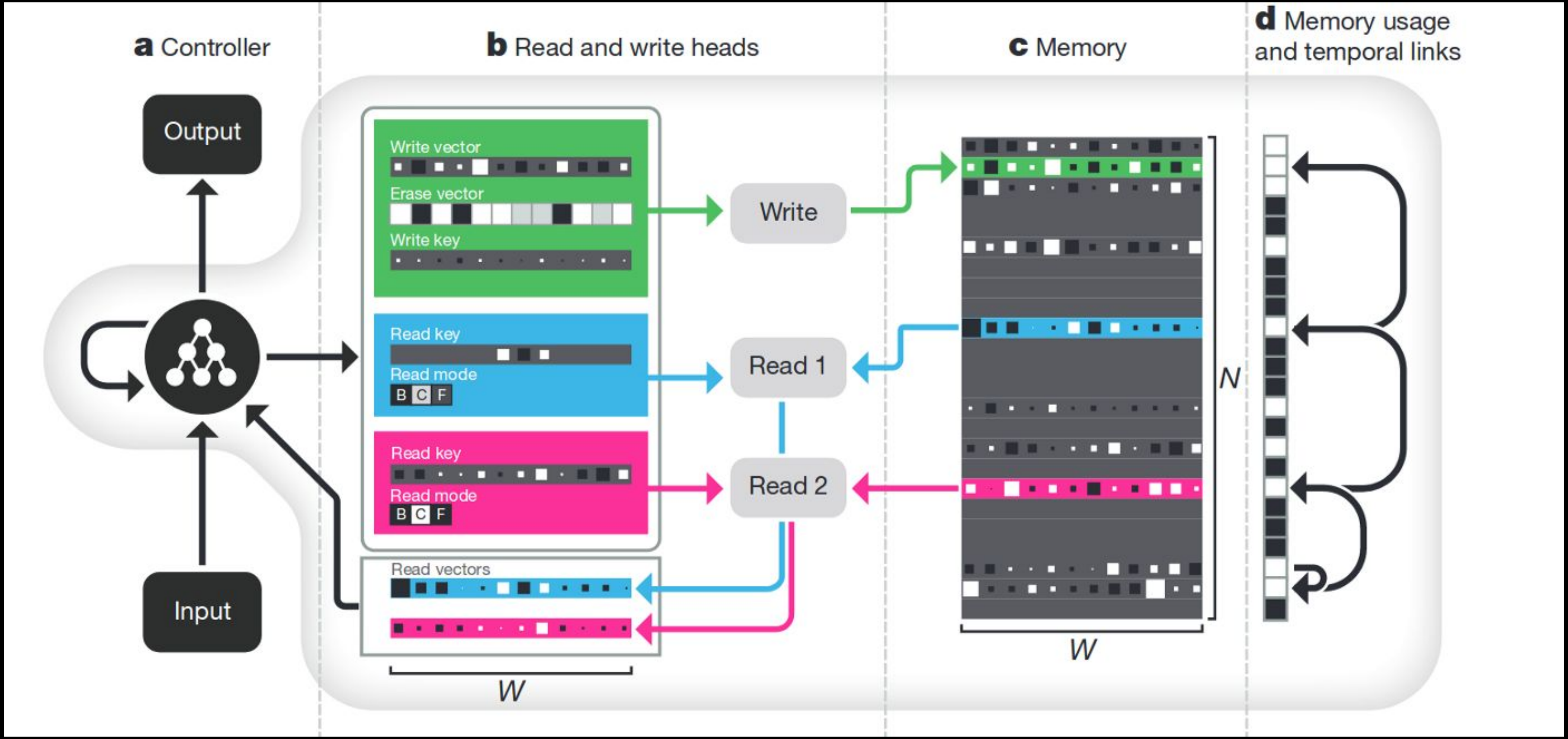
Preceding Work: Neural Turing Machine (Google Deepmind 2014)



Diff. Neural Computer (Google Deepmind 2016)



Here is a fancy version...



Controller

$\boldsymbol{\chi}_t = [\mathbf{x}_t; \mathbf{r}_{t-1}^1; \cdots; \mathbf{r}_{t-1}^R]$	Controller input matrix
Deep (layered) LSTM	$\forall\ 0 \le l \le L$
$\mathbf{i}_t^l = \sigma(W_i^l[\boldsymbol{\chi}_t; \mathbf{h}_{t-1}^l; \mathbf{h}_t^{l-1}] + \mathbf{b}_i^l)$	Input gate vector
$\mathbf{o}_t^l = \sigma(W_o^l[\boldsymbol{\chi}_t; \mathbf{h}_{t-1}^l; \mathbf{h}_t^{l-1}] + \mathbf{b}_o^l)$	Output gate vector
$\mathbf{f}_t^l = \sigma(W_f^l[\boldsymbol{\chi}_t; \mathbf{h}_{t-1}^l; \mathbf{h}_t^{l-1}] + \mathbf{b}_f^l)$	Forget gate vector
$\mathbf{s}_t^l = \mathbf{f}_t^l \mathbf{s}_{t-1}^l + \mathbf{i}_t^l \tanh(W_s^l[\boldsymbol{\chi}_t; \mathbf{h}_{t-1}^l; \mathbf{h}_t^{l-1}] + \mathbf{b}_s^l)$	State gate vector, $s_0 = 0$
$\mathbf{h}_t^l = \mathbf{o}_t^l \tanh(\mathbf{s}_t^l)$	Hidden gate vector, $h_0 = 0; h_t^0 = 0\ \forall\ t$
$\mathbf{y}_t = W_y[\mathbf{h}_t^1; \cdots; \mathbf{h}_t^L] + W_r[\mathbf{r}_t^1; \cdots; \mathbf{r}_t^R]$	DNC output vector

Read heads	$\forall\, 1 \leq i \leq R$
$\mathbf{k}_t^{r,i}$	Read keys
$\beta_t^{r,i} = \text{oneplus}(\hat{\beta}_t^{r,i})$	Read strengths
$f_t^i = \sigma(\hat{f}_t^i)$	Free gates
$\boldsymbol{\pi}_t^i = \text{softmax}(\hat{\boldsymbol{\pi}}_t^i)$	Read modes, $\boldsymbol{\pi}_t^i \in \mathbb{R}^3$
Write head	
$\mathbf{k}_t^w$	Write key
$\beta_t^w = \hat{\beta}_t^w$	Write strength
$\mathbf{e}_t = \sigma(\hat{\mathbf{e}}_t)$	Erase vector
$\mathbf{v}_t$	Write vector
$g_t^a = \sigma(\hat{g}_t^a)$	Allocation gate
$g_t^w = \sigma(\hat{g}_t^w)$	Write gate

Memory

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

Memory matrix,

Matrix of ones $E \in \mathbb{R}^{N \times W}$

$$\mathbf{u}_t = (\mathbf{u}_{t-1} + \mathbf{w}_{t-1}^w - \mathbf{u}_{t-1} \circ \mathbf{w}_{t-1}^w) \circ \boldsymbol{\psi}_t$$

Usage vector

$$\mathbf{p}_t = \left(1 - \sum_i \mathbf{w}_t^w[i]\right) \mathbf{p}_{t-1} + \mathbf{w}_t^w$$

Precedence weighting,

$\mathbf{p}_0 = \mathbf{0}$

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

Temporal link matrix,

$L_0 = \mathbf{0}$

$$\mathbf{w}_t^w = g_t^w [g_t^a \mathbf{a}_t + (1 - g_t^a) \mathbf{c}_t^w]$$

Write weighting

$$\mathbf{w}_t^{r,i} = \boldsymbol{\pi}_t^i[1] \mathbf{b}_t^i + \boldsymbol{\pi}_t^i[2] c_t^{r,i} + \boldsymbol{\pi}_t^i[3] f_t^i$$

Read weighting

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

Read vectors

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

Content-based addressing, Lookup key $\mathbf{k}$, key strength β

$$\phi_t$$

Indices of $\mathbf{u}_t$, sorted in ascending order of usage

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

Allocation weighting

$$\mathbf{c}_t^w = \mathcal{C}(M_{t-1}, \mathbf{k}_t^w, \beta_t^w)$$

Write content weighting

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

Read content weighting

$$\mathbf{f}_t^i = L_t \mathbf{w}_{t-1}^{r,i}$$

Forward weighting

$$\mathbf{b}_t^i = L_t^\top \mathbf{w}_{t-1}^{r,i}$$

Backward weighting

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

Memory retention vector

(I) Memory Write & Read (NTM & DNC)

1. Intuition: We need the process to be differentiable.
2. Trick: Every W&R use all the locations in memory, but according to different weights, decided by the controller network (trainable).
3. Issue: That complicates the entire process.

(II) Dynamic Memory Allocation (DNC)

1. Intuition: Write to memory which are least used.
2. Trick: Sort indices of memory locations according to usage & Make writing to least used locations easier (scaling up write weight).
3. Issue: Sort is not differentiable --- Ignore!

(III) Temporal Memory Linkage (DNC)

1. Intuition: Record the order in which memory locations are written to.
2. Trick: We record the ‘***degree***’ to which one location is written to ***after*** another.
3. But how: Linear combination (previous pic).

(IV) Read Weighting (DNC)

1. Intuition: Use the location written order info to read memory.
2. Trick: Allow controller to control degree to which the order matters in memory read.

Part III: Experiment

Synthetic question answering experiments

bAbI dataset: 20 synthetic question answering tasks.

Each task: a training set with 10,000 questions and a test set with 1,000 questions.

Example:
mary journeyed to the kitchen. mary moved to the bedroom. john went back to the hallway. john picked up the milk there. what is john carrying ? - john travelled to the garden. john journeyed to the bedroom. what is john carrying ? - mary travelled to the bathroom. john took the apple there. what is john carrying ? - -

{milk}, {milk}, {milk apple}

Synthetic question answering experiments

Extended Data Table 1 | bAbI best and mean results

	bAbI Best Results							bAbI Mean Results			
Task	LSTM (Joint)	NTM (Joint)	DNC1 (Joint)	DNC2 (Joint)	MemN2N (Joint) [21]	MemN2N (Single) [21]	DMN (Single) [20]	LSTM	NTM	DNC1	DNC2
1: 1 supporting fact	24.5	31.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	28.4 ± 1.5	40.6 ± 6.7	**9.0 ± 12.6**	16.2 ± 13.7
2: 2 supporting facts	53.2	54.5	1.3	0.4	1.0	**0.3**	1.8	56.0 ± 1.5	56.3 ± 1.5	**39.2 ± 20.5**	47.5 ± 17.3
3: 3 supporting facts	48.3	43.9	2.4	**1.8**	6.8	2.1	4.8	51.3 ± 1.4	47.8 ± 1.7	**39.6 ± 16.4**	44.3 ± 14.5
4: 2 argument rels.	0.4	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.8 ± 0.5	0.9 ± 0.7	**0.4 ± 0.7**	**0.4 ± 0.3**
5: 3 argument rels.	3.5	0.8	**0.5**	0.8	6.1	0.8	0.7	3.2 ± 0.5	1.9 ± 0.8	**1.5 ± 1.0**	1.9 ± 0.6
6: yes/no questions	11.5	17.1	**0.0**	**0.0**	0.1	0.1	**0.0**	15.2 ± 1.5	18.4 ± 1.6	**6.9 ± 7.5**	11.1 ± 7.1
7: counting	15.0	17.8	**0.2**	0.6	6.6	2.0	3.1	16.4 ± 1.4	19.9 ± 2.5	**9.8 ± 7.0**	15.4 ± 7.1
8: lists/sets	16.5	13.8	**0.1**	0.3	2.7	0.9	3.5	17.7 ± 1.2	18.5 ± 4.9	**5.5 ± 5.9**	10.0 ± 6.6
9: simple negation	10.5	16.4	**0.0**	0.2	**0.0**	0.3	**0.0**	15.4 ± 1.5	17.9 ± 2.0	**7.7 ± 8.3**	11.7 ± 7.4
10: indefinite knowl.	22.9	16.6	0.2	0.2	0.5	**0.0**	**0.0**	28.7 ± 1.7	25.7 ± 7.3	**9.6 ± 11.4**	14.7 ± 10.8
11: basic coreference	6.1	15.2	**0.0**	**0.0**	**0.0**	0.1	0.1	12.2 ± 3.5	24.4 ± 7.0	**3.3 ± 5.7**	7.2 ± 8.1
12: conjunction	3.8	8.9	0.1	**0.0**	0.1	**0.0**	**0.0**	5.4 ± 0.6	21.9 ± 6.6	**5.0 ± 6.3**	10.1 ± 8.1
13: compound coref.	0.5	7.4	**0.0**	0.1	**0.0**	**0.0**	0.2	7.2 ± 2.3	8.2 ± 0.8	**3.1 ± 3.6**	5.5 ± 3.4
14: time reasoning	55.3	24.2	0.3	0.4	**0.0**	0.1	**0.0**	55.9 ± 1.2	44.9 ± 13.0	**11.0 ± 7.5**	15.0 ± 7.4
15: basic deduction	44.7	47.0	**0.0**	**0.0**	0.2	**0.0**	**0.0**	47.0 ± 1.7	46.5 ± 1.6	**27.2 ± 20.1**	40.2 ± 11.1
16: basic induction	52.6	53.6	52.4	55.1	**0.2**	51.8	0.6	**53.3 ± 1.3**	53.8 ± 1.4	53.6 ± 1.9	54.7 ± 1.3
17: positional reas.	39.2	25.5	24.1	**12.0**	41.8	18.6	40.4	34.8 ± 4.1	**29.9 ± 5.2**	32.4 ± 8.0	30.9 ± 10.1
18: size reasoning	4.8	2.2	4.0	**0.8**	8.0	5.3	4.7	5.0 ± 1.4	4.5 ± 1.3	**4.2 ± 1.8**	4.3 ± 2.1
19: path finding	89.5	4.3	**0.1**	3.9	75.7	2.3	65.5	90.9 ± 1.1	86.5 ± 19.4	**64.6 ± 37.4**	75.8 ± 30.4
20: agent motiv.	1.3	1.5	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	1.3 ± 0.4	1.4 ± 0.6	**0.0 ± 0.1**	**0.0 ± 0.0**
Mean Err. (%)	25.2	20.1	4.3	**3.8**	7.5	4.2	6.4	27.3 ± 0.8	28.5 ± 2.9	**16.7 ± 7.6**	20.8 ± 7.1
Failed (err. > 5%)	15	16	**2**	**2**	6	3	**2**	17.1 ± 1.0	17.3 ± 0.7	**11.2 ± 5.4**	14.0 ± 5.0

To compare with previous results we report error rates for the single best network across all tasks (measured on the validation set) over 20 runs. The lowest error rate for each task is shown in bold. Results for MemN2N are from ref. 21; those for DMN are from ref. 20. The mean results are reported with ±s.d. for the error rates over all 20 runs for each task. The lowest mean error rate for each task is shown in bold.

The mean error and failed number of the bAbI best result of differentiable neural computer are much less than that of others. The same conclusion can be drawn from the mean results.

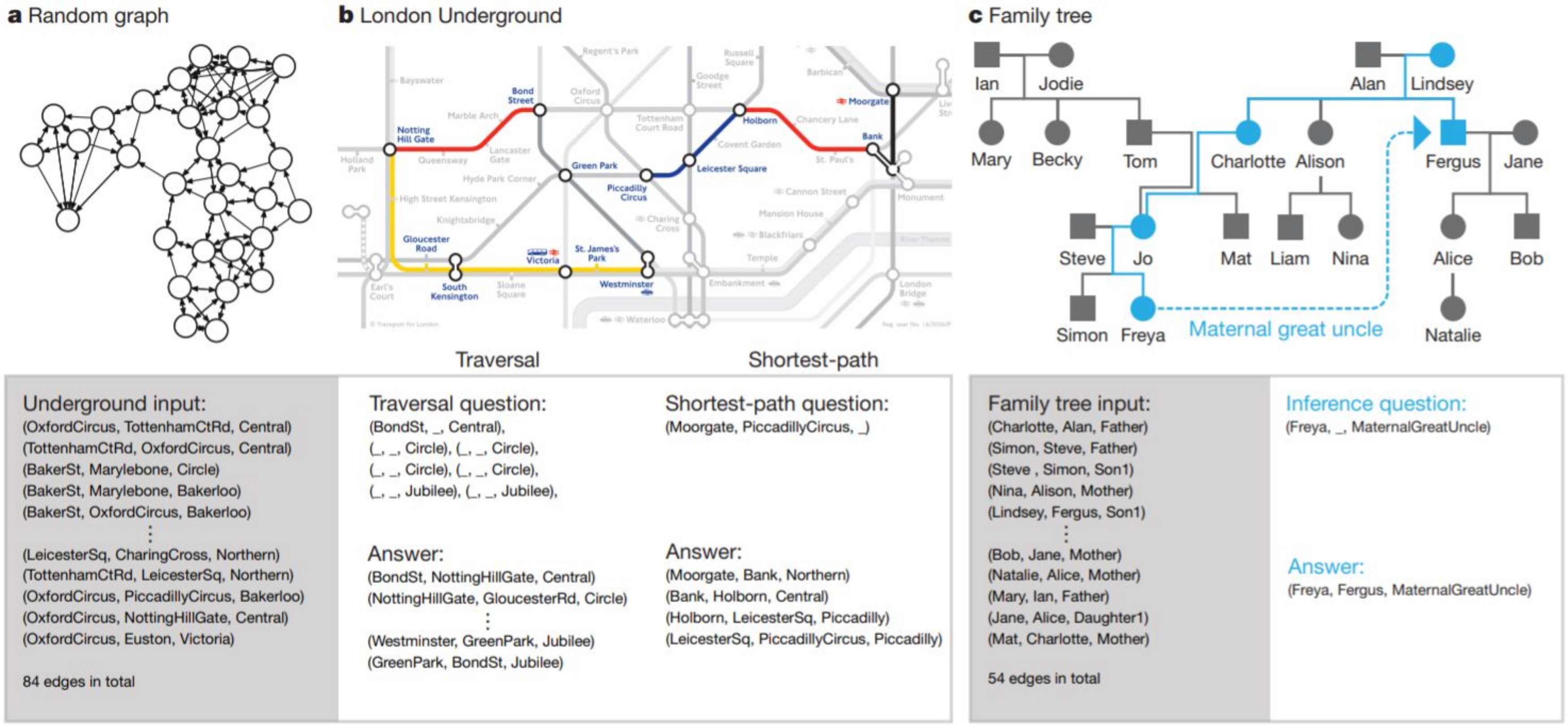
Synthetic question answering experiments

Extended Data Table 2 | Hyper-parameter settings for bAbI, graph tasks and Mini-SHRDLU

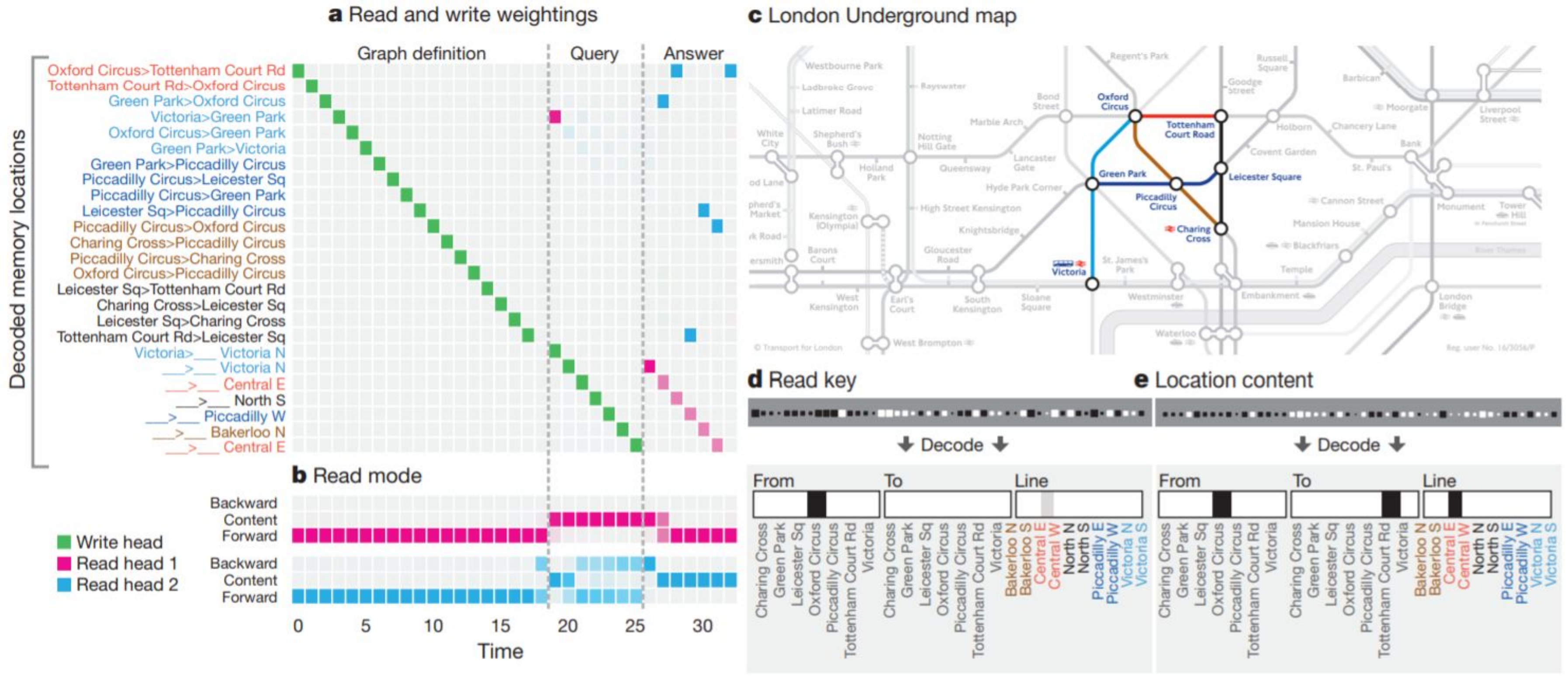
	bAbI				Graph Tasks			Mini-SHRDLU		
	LSTM	NTM	DNC1	DNC2	Shortest Path	Traversal	Inference Tasks	Fig 4 a DNC	Fig 4 a LSTM	Figure 5 DNC
LSTM Size	512	256	256	256	2×256	3×256	3×256	2×250	2×250	2×250
Batch Size	1	1	1	1	1	2	32	32	32	32
Learning Rate	1×10^{-4}	1×10^{-4}	1×10^{-4}	1×10^{-4}	3×10^{-6}	1×10^{-5}	1×10^{-5}	3×10^{-5}	3×10^{-5}	3×10^{-5}
Memory Dimensions	–	256×64	256×64	256×32	128×50	256×50	128×50	32×100	–	32×100
Read Heads	–	4	4	8	5	5	5	3	–	2
Async. Workers	16	16	16	16	–	–	–	–	–	–
DAGGER β	–	–	–	–	0.8	–	–	–	–	–
λ	–	–	–	–	–	–	–	0.75	0.5	0.5
Entropy Cost Coeff.	–	–	–	–	–	–	–	0.5	0.5	0.5

In bAbI experiments, for all models (LSTM, NTM and DNC) we kept the hyper-parameter settings that (1) gave the lowest average validation error rate and (2) gave the single best validation error rate for a single model. For LSTM and NTM the same setting was best for both criteria, but for DNC two different settings were found (DNC1 for criterion 1 and DNC2 for criterion 2).

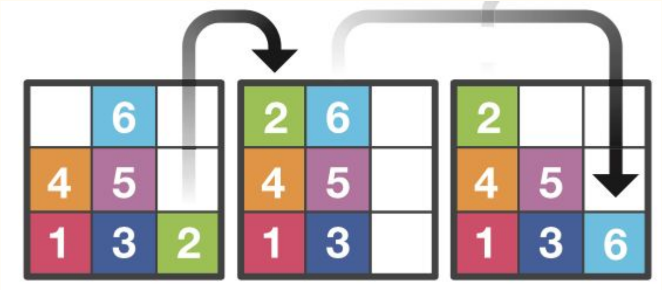
Graph experiments



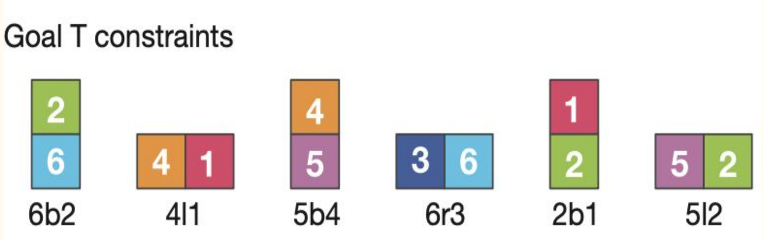
Graph experiments



Block Puzzle Experiments



- A grid board and a set of numbered blocks.
- An agent can move the top block from a column and deposit it on top of a stack in another column.
- A goal is denoted by a single-letter label and is composed of several individual constraints.(example: goal ‘T’ is 6 below 2, 4 left 1, 5 below 4,6 right 3...)



GOAL ‘T’

Block Puzzle Experiments

- The agent acts T steps to create an episode: s1,a1,s2,a2,...sT,aT. A reward function is given by $r(s_t, a_t)$, the goal of the agent is to maximize the total expected reward over an episode. $J(\pi) = E[\sum_{t=1}^{T} r(s_t, a_t)|\pi]$.
- The architecture of the reinforcement learning agent here contains two DNC networks: a policy network that selects an action and a value network that estimates the expected future reward given the policy network and current state.
- The value network updates its parameter ϕ using gradient descent on the loss function:

$$C(\phi) = \frac{1}{2L} \sum_{l=1}^{L} \sum_{t=1}^{T} \left\| \sum_{\tau=t}^{T} r(s_\tau^l, a_\tau^l) - V^\pi(o_1, \ldots, o_\tau; \phi) \right\|^2$$

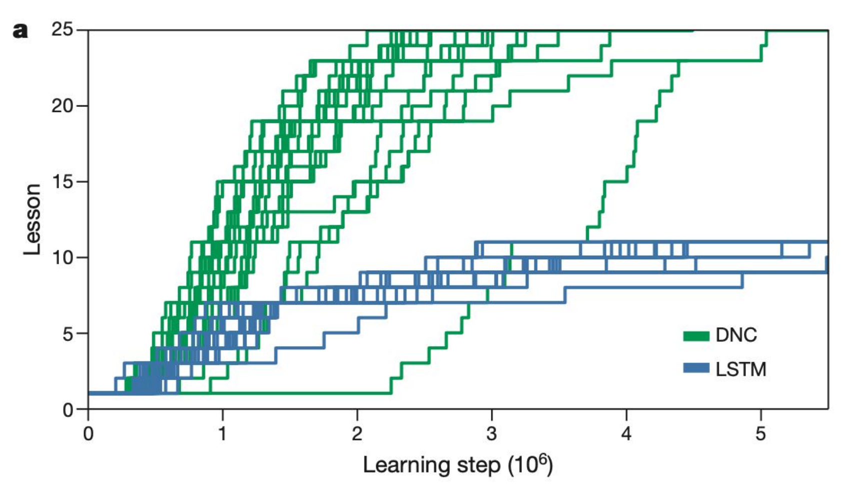
Where $V^\pi(o_1, \ldots, o_t; \phi)$ is the sum of the future rewards for this policy given the the current history of observations o1,o2,...,ot.

- The policy network update its parameter θ using gradient assent on the expected reward function. The policy gradient estimate is :

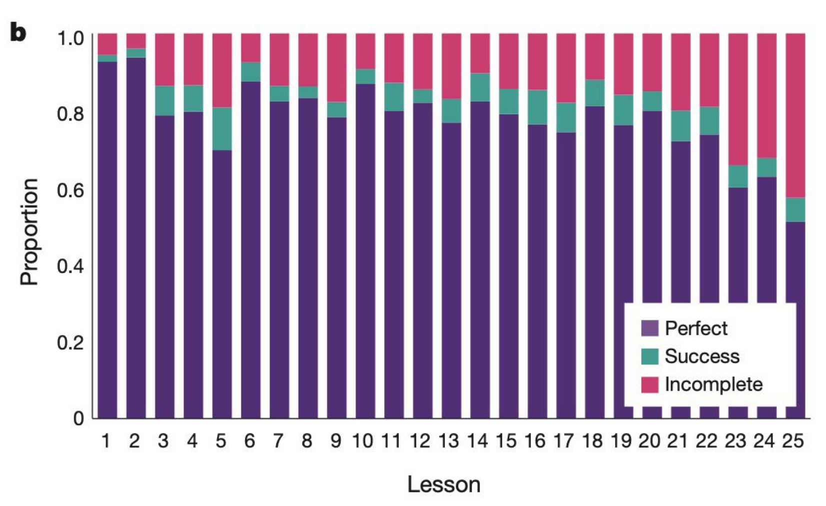
$$\nabla_\theta J(\pi) \approx \frac{1}{L} \sum_{l=1}^{L} \sum_{t=1}^{T} \nabla_\theta \log[\pi(a_t^l | o_1^l, \ldots, o_t^l; \theta)] \sum_{\tau=t}^{T} \lambda^{\tau-t} \delta_\tau^l$$

$\delta_t^l = r(s_t^l, a_t^l) + V^\pi(o_1^l, \ldots, o_{t+1}^l; \phi) - V^\pi(o_1^l, \ldots, o_t^l; \phi)$ is the temporal difference error

Block Puzzle Experiments



20 replicated training runs with different random-number seeds for DNC and LSTM, only the DNC was able to complete the learning curriculum.



A single DNC was able to solve a large percentage of problems optimally from each previous lesson(perfect), with a few episodes solved in extra moves(success), and some failures to satisfy all constraints(incomplete).

Block Puzzle Experiments

a. DNC Percent Optimal

Minimum Required Moves \ Number of Constraints	1	2	3	4	5	6
1	77	94	95	95	93	94
2	65	79	93	97	97	97
3	51	63	78	85	92	94
4	42	46	58	76	81	85
5	39	33	46	62	72	81
6	33	22	32	51	65	68
7	34	17	18	30	44	50

b. LSTM Percent Optimal

Minimum Required Moves \ Number of Constraints	1	2	3	4	5	6
1	47	48	47	48	48	52
2	39	38	34	34	31	32
3	32	42	43	46	44	43
4	25	22	18	14	12	14
5	19	10	3	0.47	0	0.16
6	20	4.7	1.1	0.16	0	0
7	18	3	1.1	0	0	0

Probability of achieving optimal solution.

Conclusion

- **Differentiable neural computer(DNC) is like a conventional computer, it can use its memory to represent and manipulate complex data structure, but, like a neural network, it can learn to do so from data.**

- **When trained with supervised learning, this paper demonstrates that a DNC can successfully answer synthetic questions designed to emulate reasoning and inference problems in natural language and it can learn tasks such as finding the shortest path between specified points in randomly generated graphs.When trained with reinforcement learning, a DNC can complete a moving blocks puzzle much better than traditional LSTM.**

- **Taken together, this paper demonstrates that DNC has the capacity to solve complex, structured tasks that are inaccessible to neural networks without external read–write memory.**