# Hybrid computing using a neural network with dynamic external memory

—

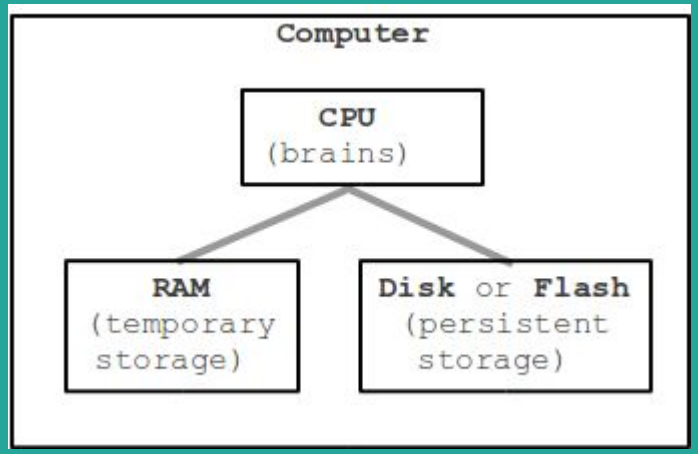
By Fangyuan Yu, Shuai Lu, Lukang Sun

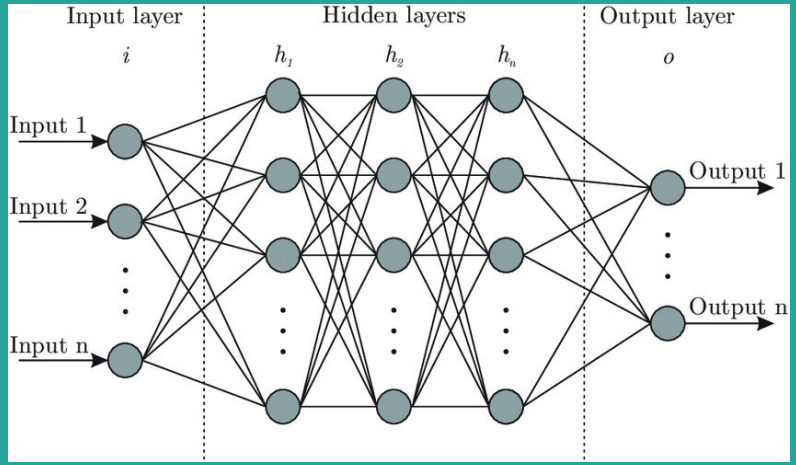
# Part I: Introduction

*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. .....

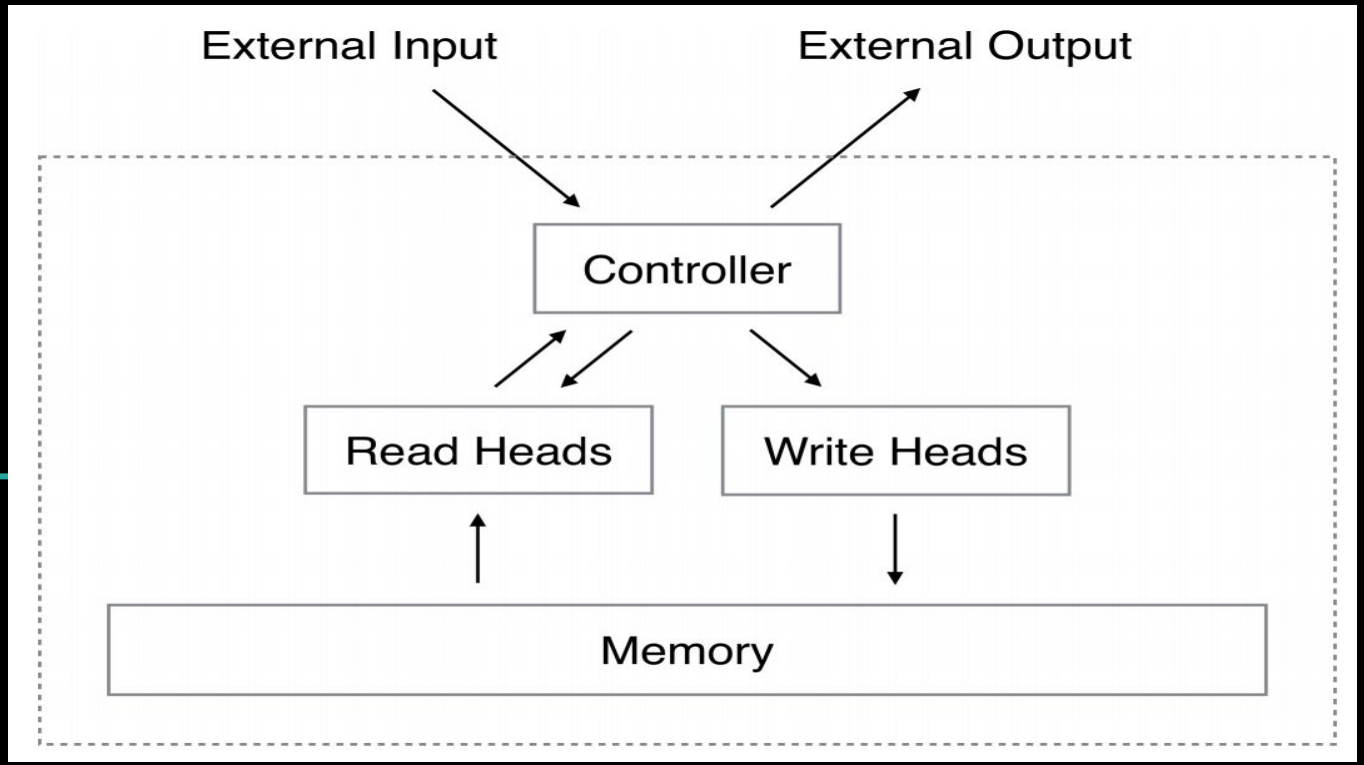
Main Idea: It is beneficial to include ‘external memory’ into artificial neural network?

# Part II: Algorithm

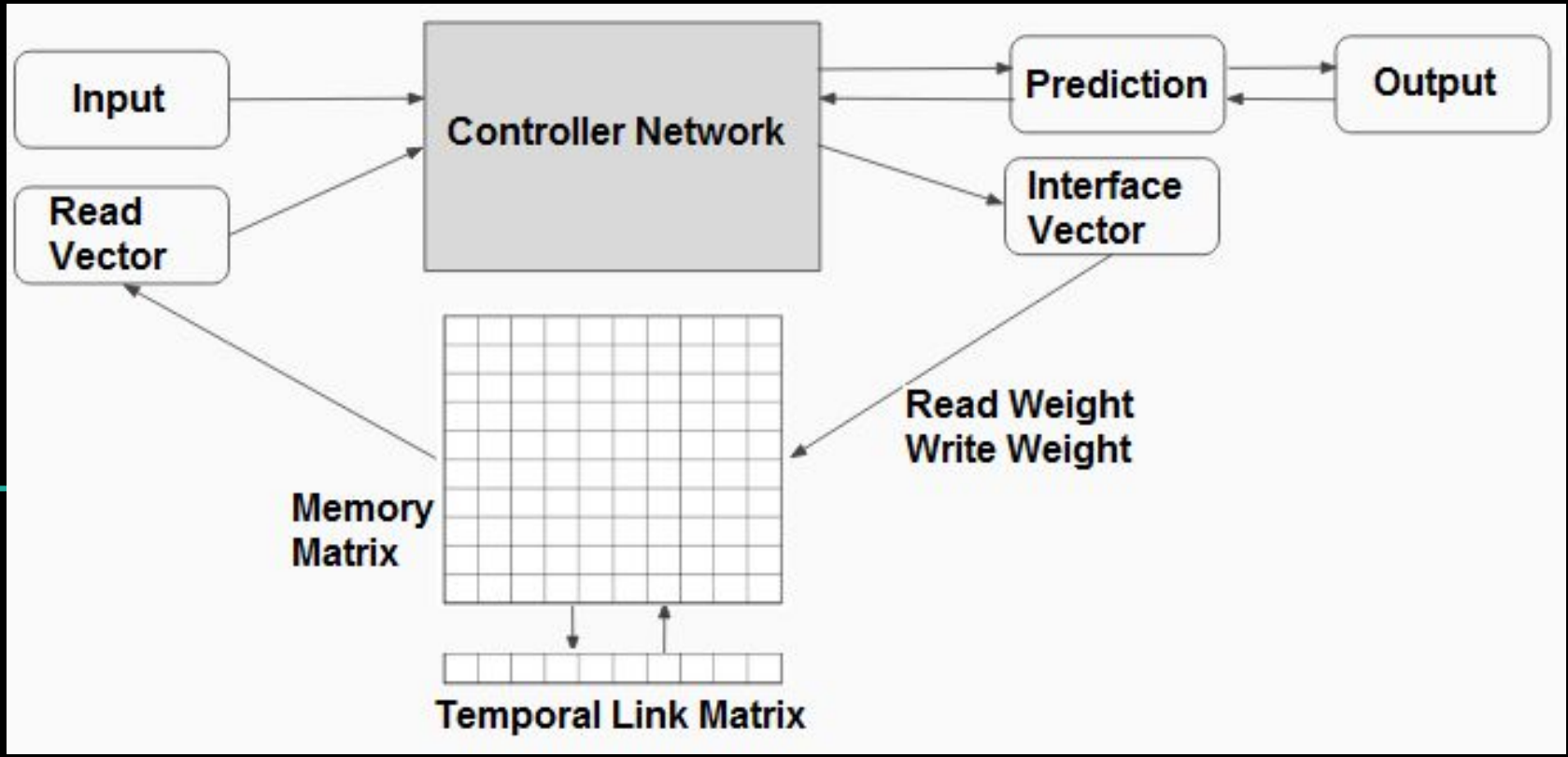
**Basic Component:**

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

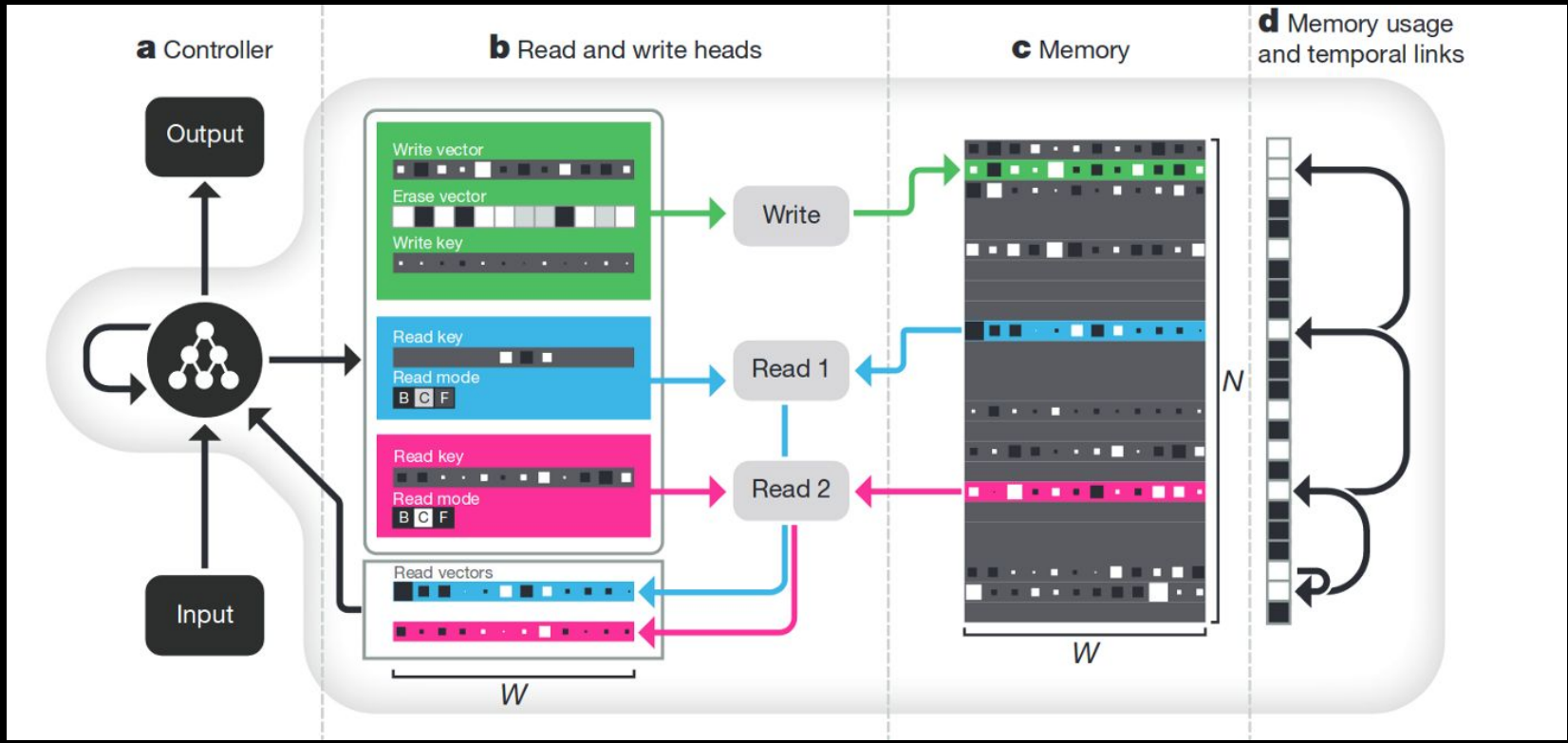
# Preceding Work: Neural Turing Machine (Google Deepmind 2014)



# Diff. Neural Computer (Google Deepmind 2016)



# Here is a fancy version...



1. How do you read/write?
   By ***weighted sum*** of memory vectors.
   Quote: "*Taking inspiration from the input and forget gates in LSTM...*" - *A.Graves*

2. What is the difference between NTM & DHC?

*The way they compute read/forget/write weights.*

# 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**

| Task | bAbI Best Results | | | | | | | bAbI Mean Results | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 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.

# Synthetic question answering experiments

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

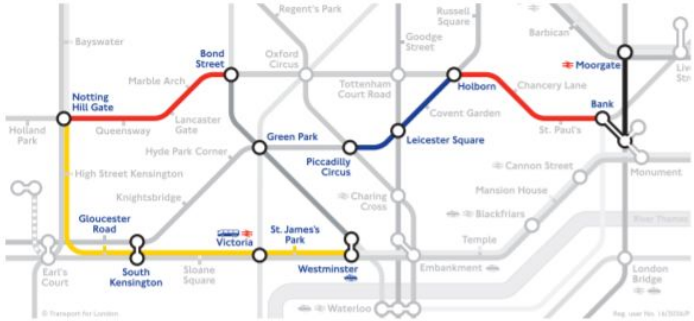
| | bAbI | | | | Graph Tasks | | | Mini-SHRDLU | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | LSTM | NTM | DNC1 | DNC2 | Shortest Path | Traversal | Inference Tasks | Fig 4 a DNC | Fig 4 a LSTM | Figure 5 DNC |
| LSTM Size | 512 | 256 | 256 | 256 | 2 × 256 | 3 × 256 | 3 × 256 | 2 × 250 | 2 × 250 | 2 × 250 |
| Batch Size | 1 | 1 | 1 | 1 | 1 | 2 | 32 | 32 | 32 | 32 |
| Learning Rate | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ | $1 \times 10^{-4}$ | $3 \times 10^{-6}$ | $1 \times 10^{-5}$ | $1 \times 10^{-5}$ | $3 \times 10^{-5}$ | $3 \times 10^{-5}$ | $3 \times 10^{-5}$ |
| 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).

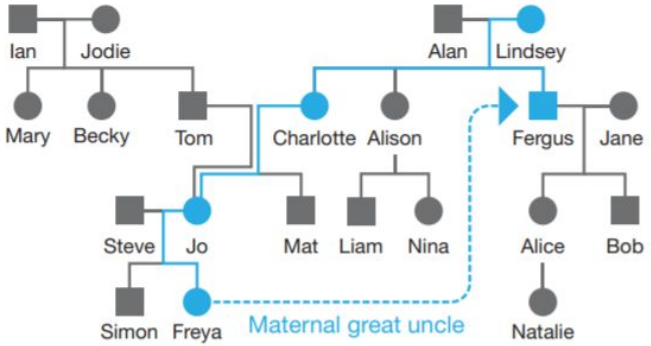
# Graph experiments

**a** Random graph

**b** London Underground




**c** Family tree




Traversal

Shortest-path

**Underground input:**
(OxfordCircus, TottenhamCtRd, Central)
(TottenhamCtRd, OxfordCircus, Central)
(BakerSt, Marylebone, Circle)
(BakerSt, Marylebone, Bakerloo)
(BakerSt, OxfordCircus, Bakerloo)
⋮
(LeicesterSq, CharingCross, Northern)
(TottenhamCtRd, LeicesterSq, Northern)
(OxfordCircus, PiccadillyCircus, Bakerloo)
(OxfordCircus, NottingHillGate, Central)
(OxfordCircus, Euston, Victoria)

84 edges in total

**Traversal question:**
(BondSt, _, Central),
(_, _, Circle), (_, _, Circle),
(_, _, Circle), (_, _, Circle),
(_, _, Jubilee), (_, _, Jubilee),

**Answer:**
(BondSt, NottingHillGate, Central)
(NottingHillGate, GloucesterRd, Circle)
⋮
(Westminster, GreenPark, Jubilee)
(GreenPark, BondSt, Jubilee)

**Shortest-path question:**
(Moorgate, PiccadillyCircus, _)

**Answer:**
(Moorgate, Bank, Northern)
(Bank, Holborn, Central)
(Holborn, LeicesterSq, Piccadilly)
(LeicesterSq, PiccadillyCircus, Piccadilly)

**Family tree input:**
(Charlotte, Alan, Father)
(Simon, Steve, Father)
(Steve , Simon, Son1)
(Nina, Alison, Mother)
(Lindsey, Fergus, Son1)
⋮
(Bob, Jane, Mother)
(Natalie, Alice, Mother)
(Mary, Ian, Father)
(Jane, Alice, Daughter1)
(Mat, Charlotte, Mother)

54 edges in total

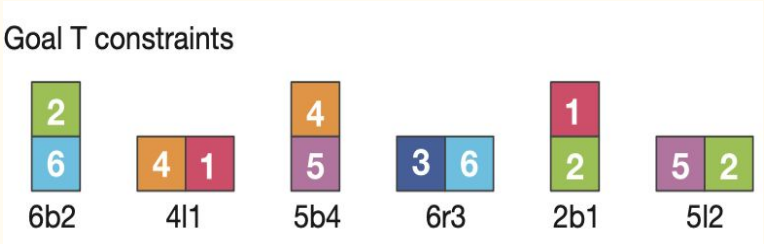
**Inference question:**
(Freya, _, MaternalGreatUncle)

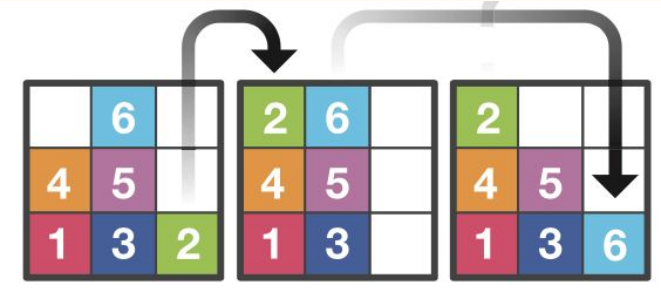
**Answer:**
(Freya, Fergus, MaternalGreatUncle)

# Graph experiments

# Block Puzzle Experiments

- A grid board and a set of numbered blocks.
- A 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...)

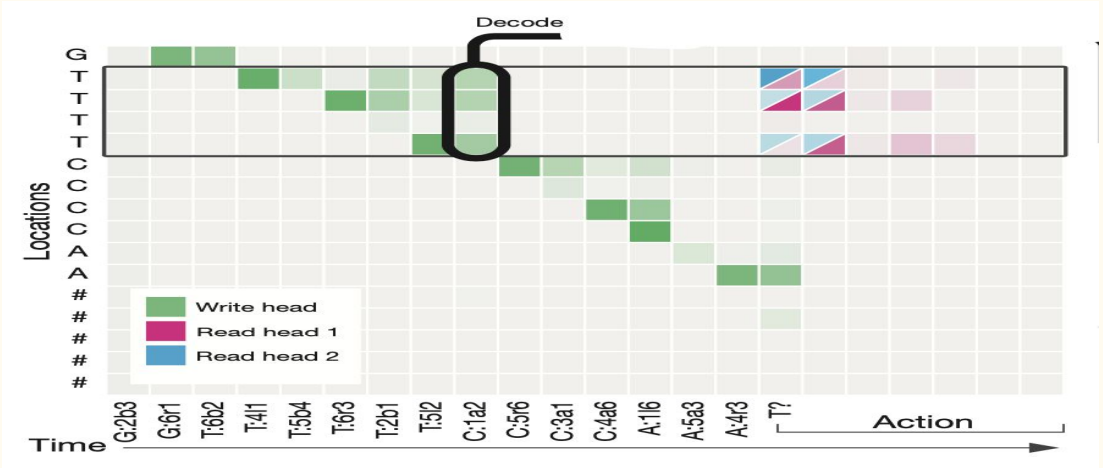


GOAL ‘T’

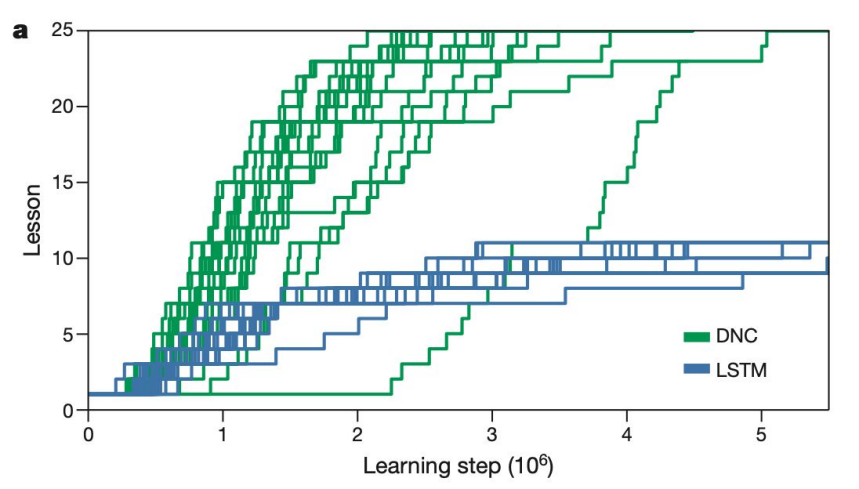
# Block Puzzle Experiments

- Train DNC so that it can move blocks to achieve any goal(satisfy all constraints) .
- Observe that at the time a goal was written but many steps before execution was required, the first action could be decoded from memory. This indicates that DNC has written its decision to memory before acting upon it. DNC could make a plan!!

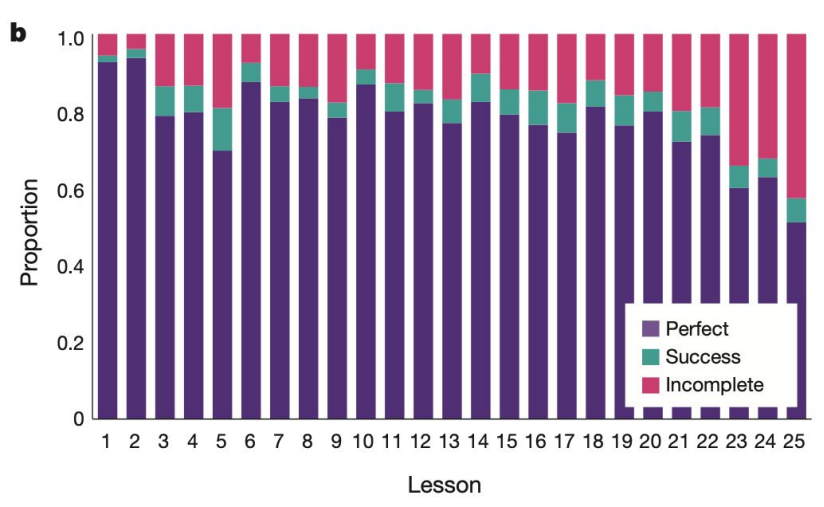



The DNC used its memory to store the instructions by iteratively writing goals to locations

# Block Puzzle Experiments

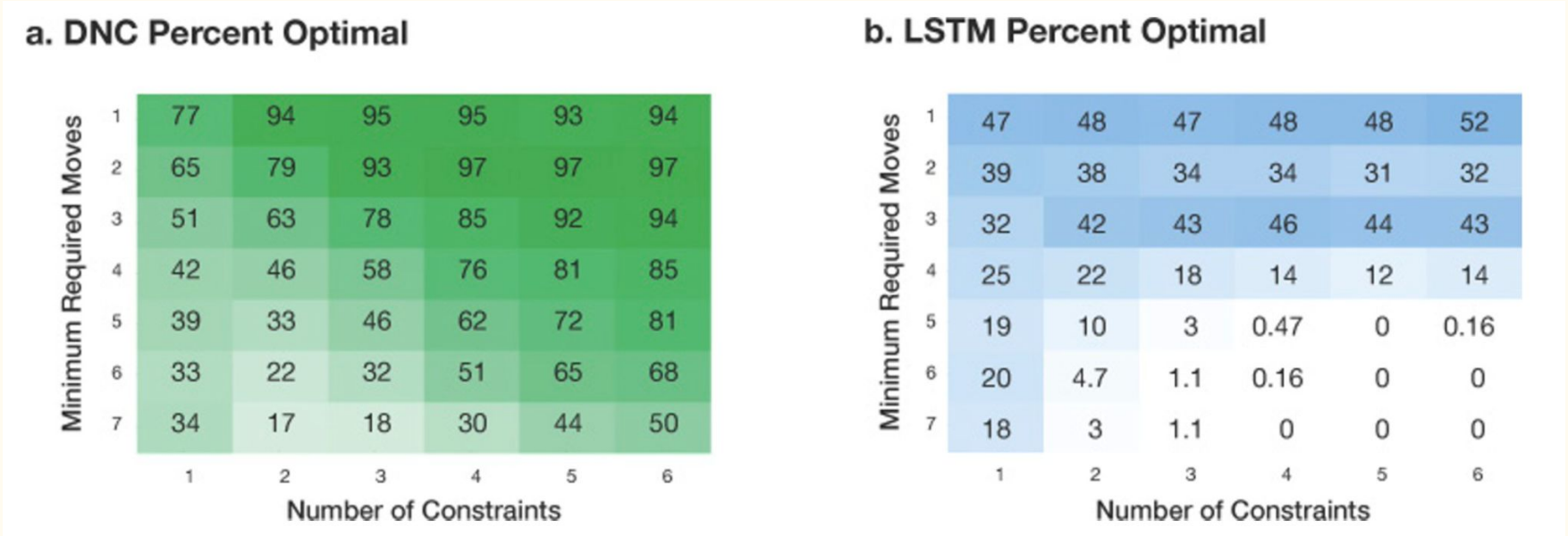



20 replicated training runs with different random-number seeds for DNC and LSTM, only the DNC was able to complete the learning curriculum.




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




**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 DNCs have the capacity to solve complex, structured tasks that are inaccessible to neural networks without external read–write memory.**