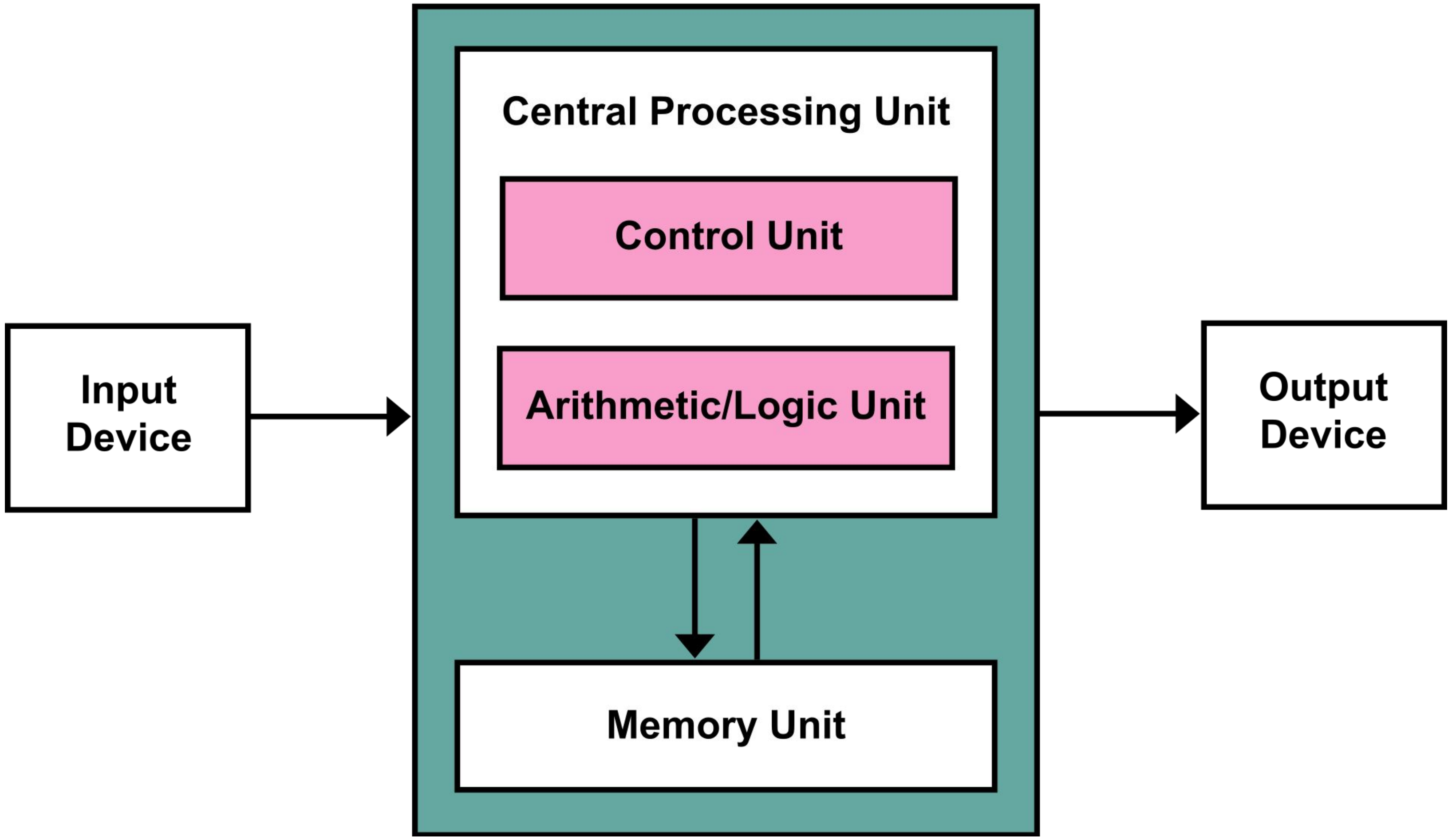
Differentiable Neural Computers

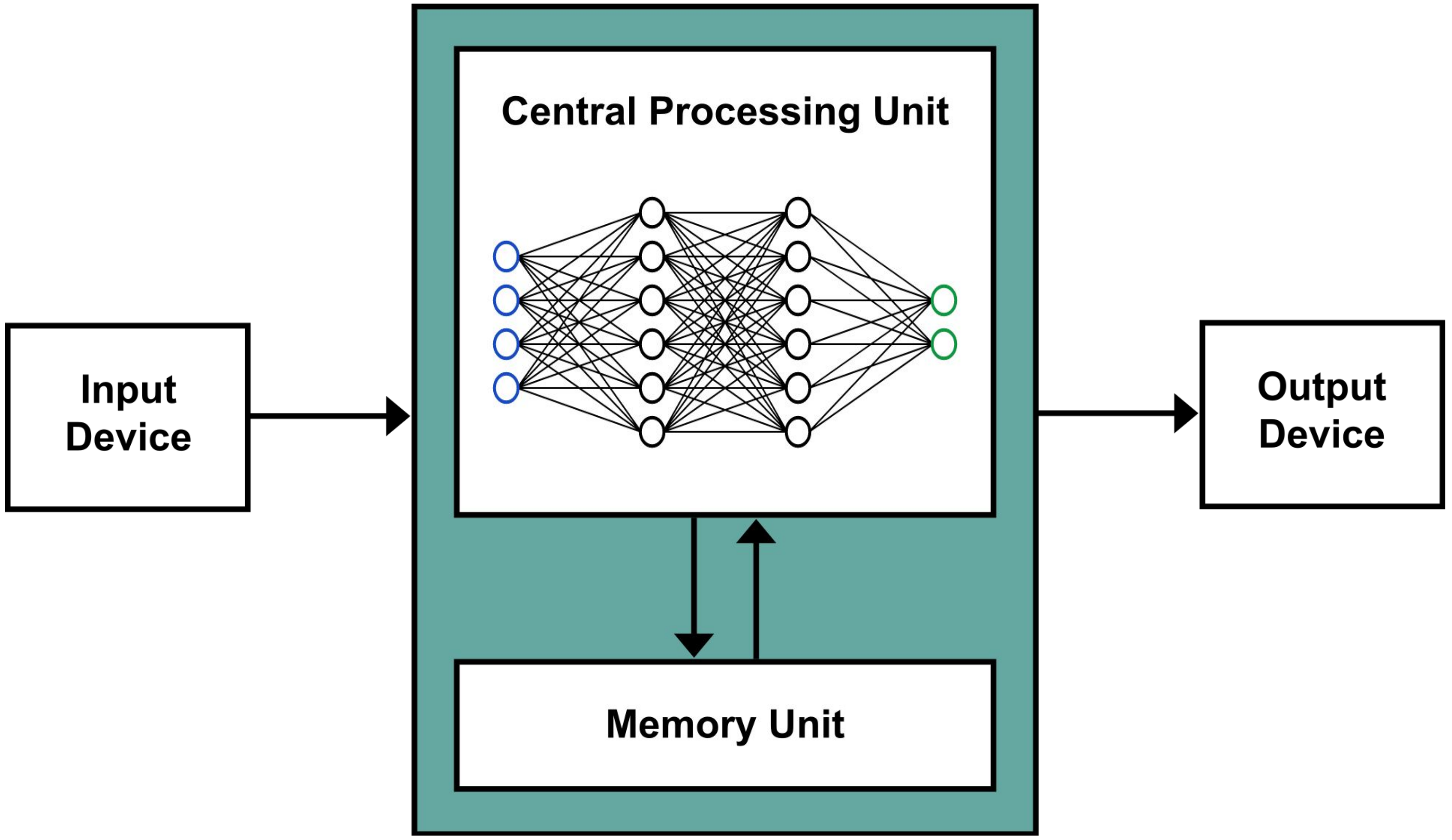
Shashank Shekhar
Masters in Applied Science candidate,
Machine Learning Research Group, University of Guelph
Vector Scholar, Vector Institute

Computer

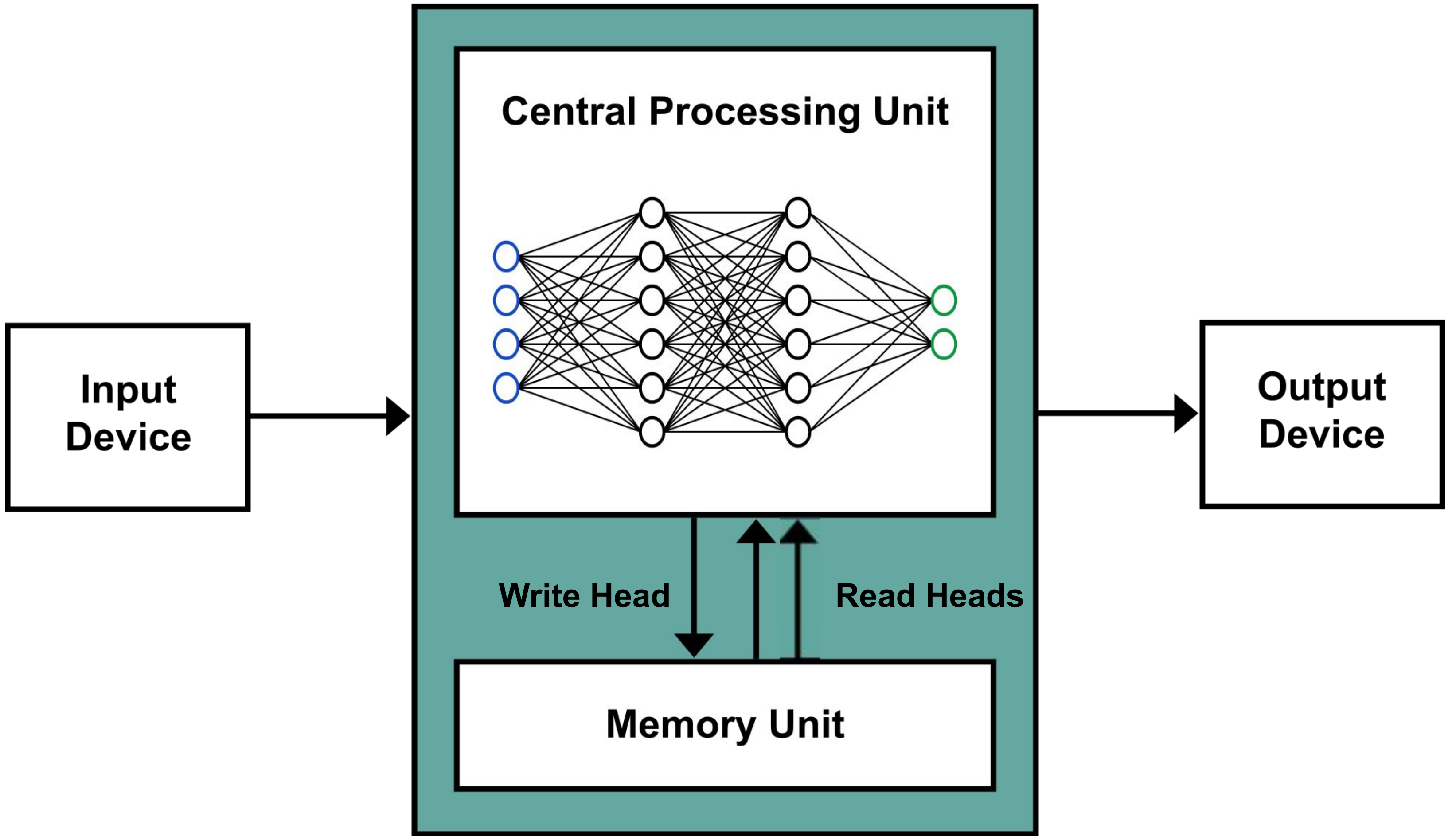


von Neumann architecture, Wikipedia

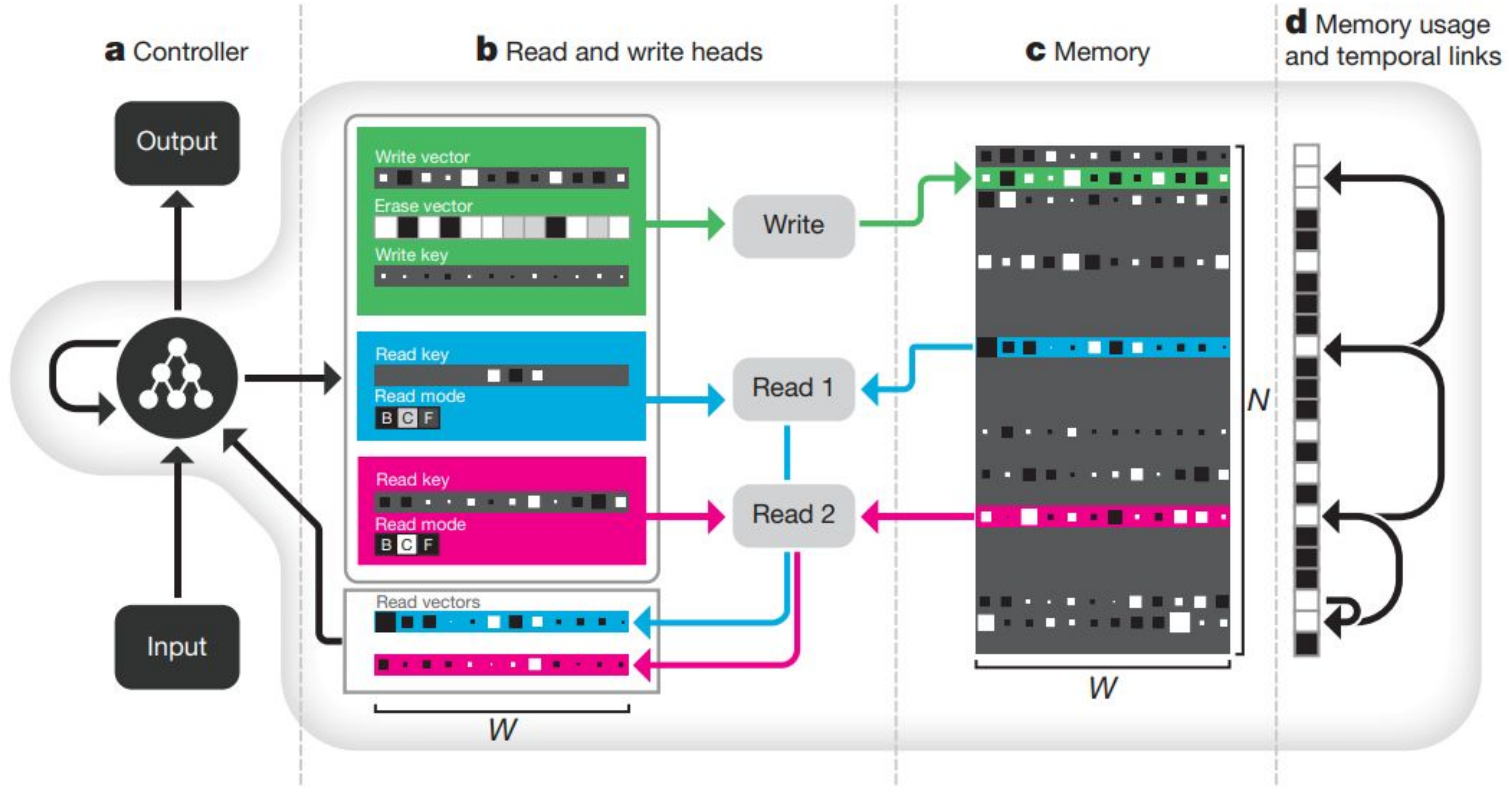
Neural Computer



Differentiable Neural Computer



Differentiable Neural Computer



DNC Controller

- Neural network N: Use LSTM neural network with L layers

- Input: input vector x_t of size X
- Output: output vector y_t of size Y
- Also receives, set of R read vectors: $r^1_{t-1}, ..., r^R_{t-1}$

- Controller input: $[x_t; r^1_{t-1}, \ldots; r^R_{t-1}]$
- Controller output:
    - $v_t = W_y[h^1_t, ..., h^R_t]$ (output vector)
    - $\xi_t = W_\xi[h^1_t, ..., h^R_t]$ (interface vector)

- Final output: $y_t = v_t + W_r[r^1_t, ..., r^R_t]$

DNC Memory Interfacing

- Interface vector ξ_t

$$\xi_t = \Big[\overbrace{k_t^{r,1}; \dots; k_t^{r,R}}^{R \text{ read keys } \in \mathbb{R}^W}; \overbrace{\hat{\beta}_t^{r,1}; \dots, \hat{\beta}_t^{r,R}}^{R \text{ read strengths } \in [1, \infty)}; \underset{\hookrightarrow \text{write key } \in \mathbb{R}^W}{k_t^w}; \underset{\hookrightarrow \text{write strength } \in [1, \infty)}{\hat{\beta}_t^w}, \underset{\hookrightarrow \text{Erase vector } \in [0, 1]^W}{\hat{e}_t}, \overbrace{v_t}^{\text{write vector } \in \mathbb{R}^W};$$

$$\underbrace{\hat{f}_t^1; \dots, \hat{f}_t^R}_{R \text{ free gates } \in [0, 1]}; \underbrace{\hat{g}_t^a}_{\text{allocation gate } \in [0, 1]}; \underbrace{\hat{g}_t^w}_{\text{write gate } \in [0, 1]}; \underbrace{\hat{\pi}_t^1; \dots; \hat{\pi}_t^R}_{R \text{ read modes}}\Big]$$

DNC memory operations

- **Where to read:**
  - Content based addressing (similarity)
  - Temporal memory linkage (linked list of memory locations in sequence they were written)

  For each read operation, read weightings are calculated $[w^{r,1}_t, \ldots; w^{r,R}_t]$ which are used to define read vectors: $r^i_t = M^T_t w^{r,i}_t$

- **Where to write:**
  - Content based addressing (similarity)
  - Dynamic memory allocation (allocating empty memory locs)

    Write operation uses the write weighting, erase and write vectors emitted by the controller to update memory as:

$$M_t = M_{t-1} \circ (E - w^w_t e^T_t) + w^w_t v^T_t$$

VECTOR INSTITUTE | UNIVERSITY of GUELPH

DNC memory operations: Content based addressing

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

- Take lookup key ($k \in R^W$), find its cosine similarity D with each memory location and then weigh it by the key strength β

- The weighting C(M, k, β) defines a normalized probability distribution over all the addresses in the memory

DNC memory operations: Dynamic memory allocation

- $u_t \in [0, 1]^N$ represents memory usage at time t. $u_0 = 0$.
- Before writing to memory, controller emits set of free gates f^i_t (one per each read head) This determines whether the most recently read location can be freed.
- Memory retention vector: $\psi_t = \prod^R(1 - f^i_t w^{r,i}_{t-1})$ -> how much of each location not freed
- Usage vector is updated as:
  $$u_t = (u_{t-1} + w^w_{t-1} - u_{t-1} \circ w^w_{t-1}) \circ \psi_t$$
- Once we have the usage vectors, we can sort the indices in ascending order of usage to determine which ones are 'most free' in the free list φ_t. $\varphi[1]$ = least used location
- Allocation weighings return provide new locations for writing:
  $$a_t[\varphi_t[j]] = (1 - u_t \varphi_t[j]) \prod^{j-1} u_t \varphi_t[i]$$

DNC memory operations: Write weighting

- The write head computes write content weighting using write key:

$$c^w_t = C(M_{t-1}, k^w_t, \beta^w_t)$$

- It is interpolated with the allocation weighting to determine the write weighting:

$$w^w_t = g^w_t[g^a_t a_t + (1 - g^a_t)c^w_t]$$

DNC memory operations: Temporal memory linkage

$L_t \in [0, 1]^{NxN}$ is a temporal link matrix which tracks consecutively stored memory locations:

- Require a precedence weighting p_t
  $p_t[i]$ represents the degree to which location i was the last location written to:

$$p_0 = 0$$
$$p_t = (1 - \sum w^w_t[i])p_{t-1} + w^w_t$$

- Recurrence relation implements the memory linkage logic:

$$L_0[i, j] = 0, L_t[i, i] = 0$$
$$L_t[i, j] = (1 - w^w_t[i] - w^w_t[j])L_{t-1}[i, j] + w^w_t[i]p_{t-1}[j]$$

- The backward and forward weighting of the read head are given by:

$f^i_t = L_t w^{r,i}_{t-1}$ **(not to be confused with free gate)**

$$b^i_t = L'_t w^{r,i}_{t-1}$$

DNC memory operations: Read weighting

- Each read head i computes content weighting using read key:

$$c^{r,i}_t = C(M_t, k^{r,i}_t, \beta^{r,i}_t)[i]$$

- Each read head also receives read mode vector π^i_t which interpolates between backward weighting, forward weighting and the content read:

$$w^{r,i}_t = \pi^i_t[1]b^i_t + \pi^i_t[2]c^{r,i}_t + \pi^i_t[2]f^i_t$$

Task: bAbI

- Input: Text (as one-hot vectors)

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 ? - -

- Output: Answers at the - (as softmax over vocabulary size)

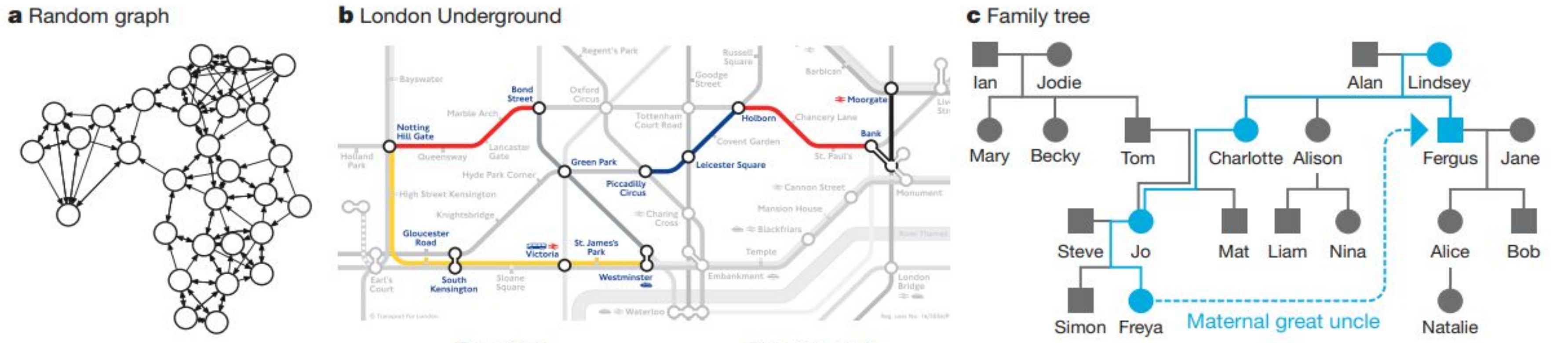
{milk}, {milk}, {milk apple}

- Optimize: cross-entropy of softmax outputs

- Evaluate: per-task (20 tasks) question error rate i.e. incorrectly answered questions

VECTOR INSTITUTE | UNIVERSITY of GUELPH

Task: Graphs



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)

Figure 2 | Graph tasks. a, An example of a randomly generated graph used for training. **b**, Zone 1 interchange stations of the London Underground map, used as a generalization test for the traversal and shortest-path tasks. Random seven-step traversals (an example of which is shown on the left) were tested, yielding an average accuracy of 98.8%. Testing on all possible four-step shortest paths (example shown on the right) gave an average accuracy of 55.3%. **c**, The family tree that was used as a generalization test for the inference task; four-step relations such as the one shown in blue (from Freya to Fergus, her maternal great uncle) were tested, giving an average accuracy of 81.8%. The symbol sequences processed by the network during the test examples are shown beneath the graphs. The input is an unordered list of ('from node', 'to node', 'edge') triple vectors that describes the graph. For each task, the question is a sequence of triples with missing elements (denoted '_') and the answer is a sequence of completed triples.

VECTOR INSTITUTE | UNIVERSITY of GUELPH

Questions and Comments?

Credits

Eric for slide layout