3. Reasoning as Memory

Introduction

Item memory

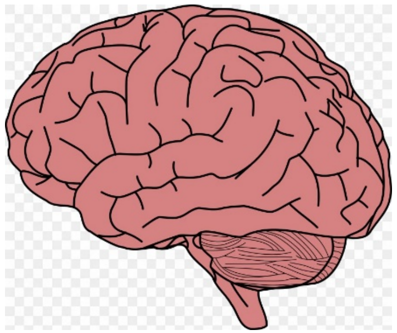
Relational memory

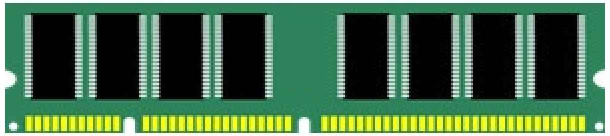
Program memory

Introduction

Memory is part of intelligence

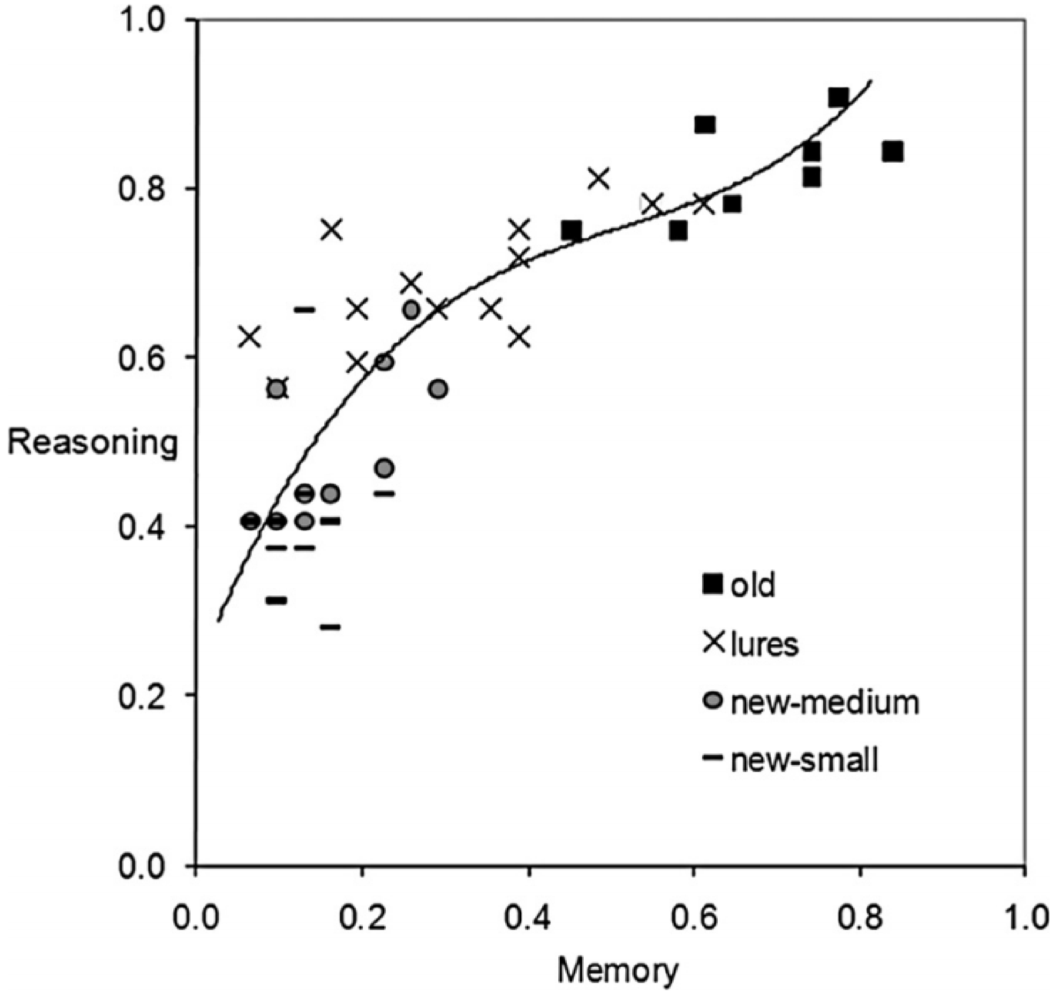
- Memory is the ability to store, retain and recall information
- Brain memory stores items, events and high-level structures
- Computer memory stores data and temporary variables





Memory-reasoning analogy

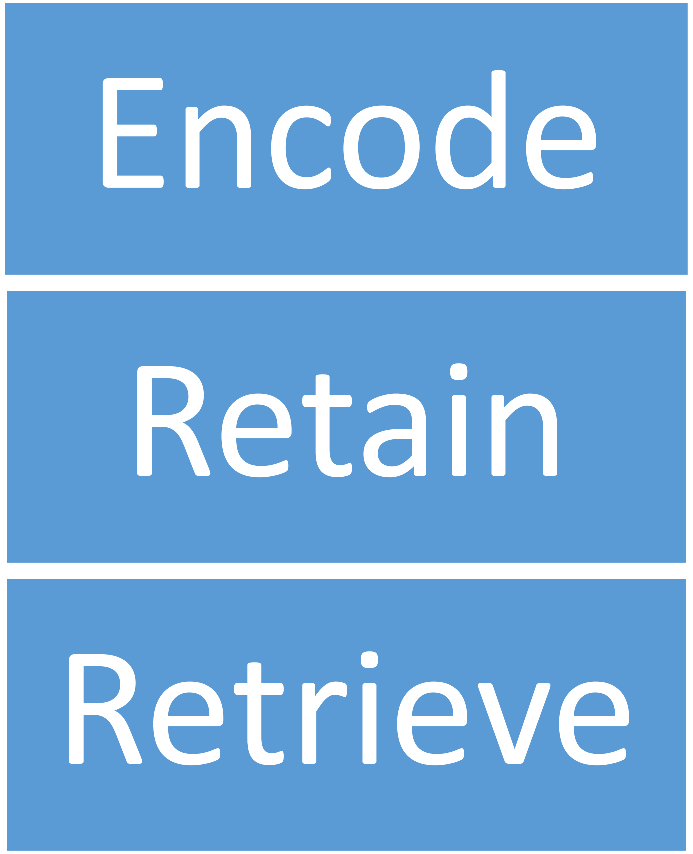
- 2 processes: fast-slow
  - Memory: familiarity-recollection
- Cognitive test:
  - Corresponding reasoning and memorization performance
  - Increasing # premises, inductive/deductive reasoning is affected



Heit, Evan, and Brett K. Hayes. "Predicting reasoning from memory." *Journal of Experimental Psychology: General* 140, no. 1 (2011): 76.

Common memory activities

- Encode: write information to the memory, often requiring compression capability
- Retain: keep the information overtime. This is often assumed in machinery memory
- Retrieve: read information from the memory to solve the task at hand



Memory taxonomy based on memory content

Item Memory	• Objects, events, items, variables, entities
Relational Memory	• Relationships, structures, graphs
Program Memory	• Programs, functions, procedures, how-to knowledge

Item memory

Associative memory

RAM-like memory

Independent memory

Distributed item memory as associative memory

Language

"Green" means "go," but what does "red" mean?

Semantic memory

Time

birthday party on 30th Jan

Episodic memory

Object

Where is my pen? What is the password?

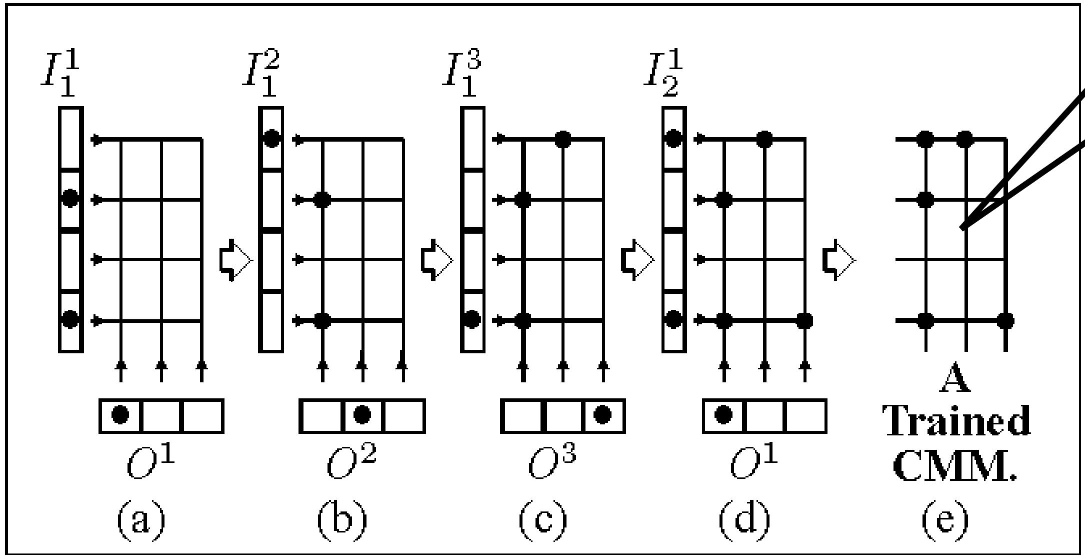
Working memory

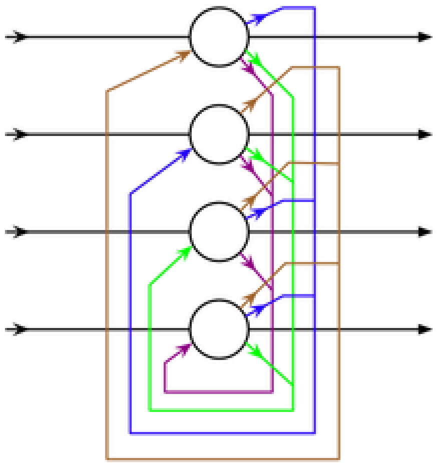
Behaviour

Motor memory

Associate memory can be implemented as Hopfield network

Fast-weight
$\widehat{M}$





Correlation matrix memory

Hopfield network

Encode $\hat{\mathbf{M}} = \sum_{k=1}^{q} \mathbf{b}_k \mathbf{a}_k^T$ Retrieve $\mathbf{b} = \hat{\mathbf{M}}\mathbf{a}_j$

Feed-forward retrieval

Retrieve $x_i(t+1) = \operatorname{sign}\left(\sum_{j=1}^{N} w_{ij} x_j(t)\right)$

Recurrent retrieval

Rule-based reasoning with associative memory

- Encode a set of rules: "pre-conditions

→post-conditions"

- Support variable binding, rule-conflict handling and partial rule input
- Example of encoding rule "A:1,B:3,C:4→X"

Conversion of lexical items to binary patterns.

```
A = 100010     1 = 100001     X = 0011
B = 011000     2 = 010010     Y = 0110
C = 000110     3 = 001100     Z = 1100
               4 = 000101
               5 = 000011
               6 = 110000
```

Outer product for binding

Bind tokens to form tensors.

```
A:1=          B:3 =         C:4 =
    100001        000000        000000
    000000        001100        000000
    000000        001100        000000
    000000        000000        000101
    100001        000000        000101
    000000        000000        000000
```

Assign separators.

S_1 = 00001
S_2 = 10000
S_3 = 00100

Superimpose to form SIBs.

$A{:}1\ .\ B{:}3\ .\ C{:}4 = SIB_1 =$

```
100001
001100
001100
000101
100101
000000
```

Train into arity 3 network.

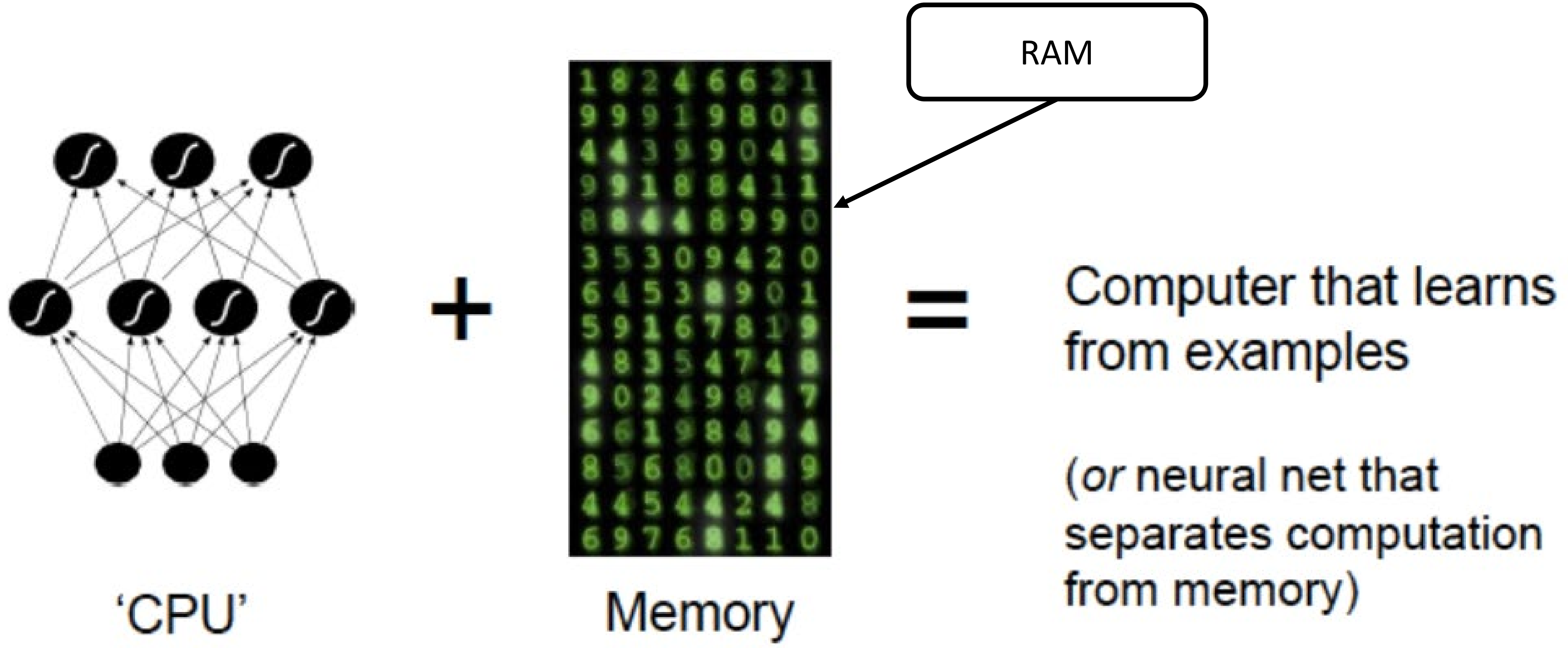
(too large to show).

$W_{new}{}^{in} = SIB_1 \times S_1$

$W_{new}{}^{out} = S_1 \times X$

Austin, Jim. "Distributed associative memories for high-speed symbolic reasoning." *Fuzzy Sets and Systems* 82, no. 2 (1996): 223-233.

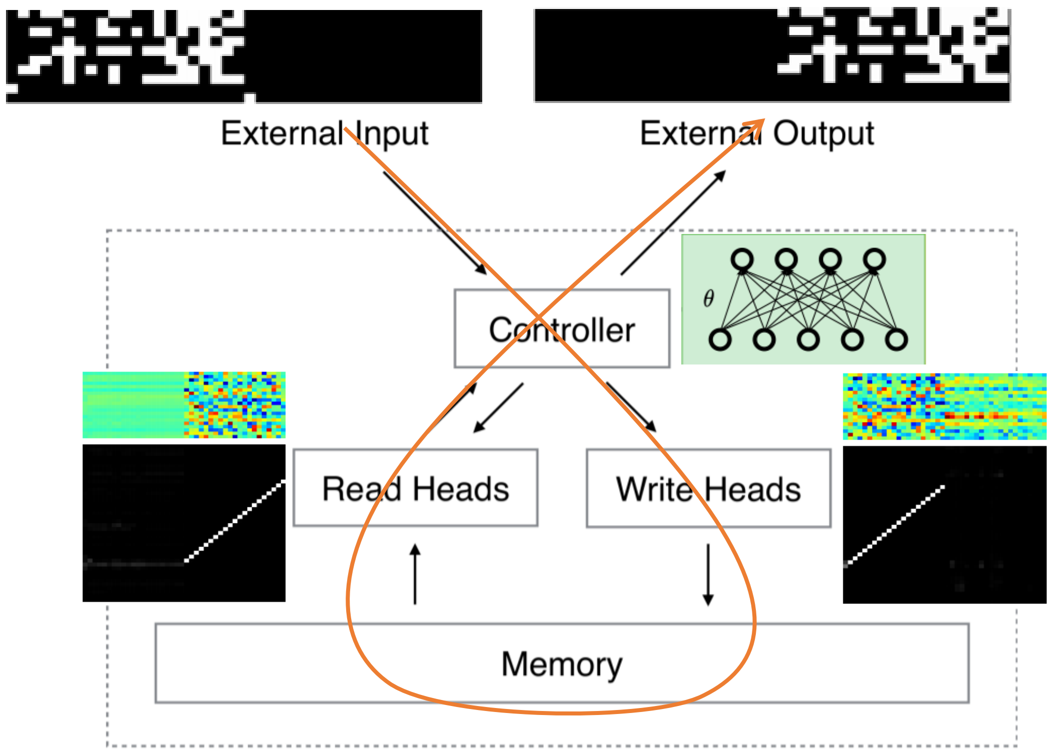
Memory-augmented neural networks: computation-storage separation



Neural Turing Machine (NTM)

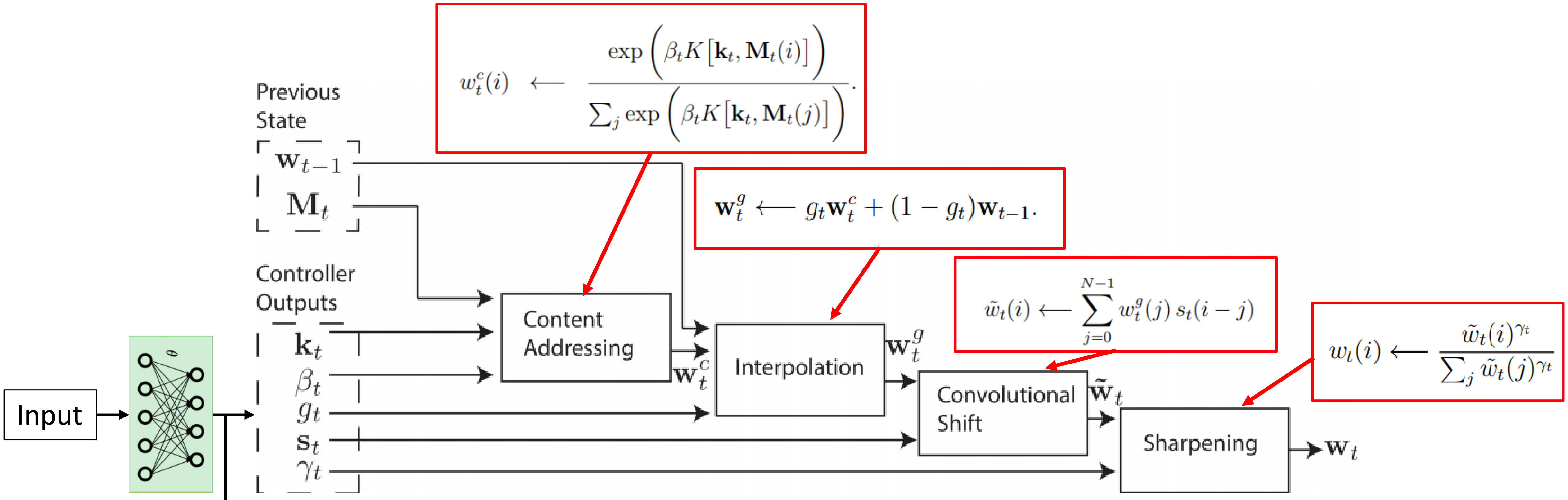
- Memory is a 2d matrix
- Controller is a neural network
- The controller read/writes to memory at certain addresses.
- Trained end-to-end, differentiable
- Simulate Turing Machine

→support symbolic reasoning, algorithm solving



Graves, Alex, Greg Wayne, and Ivo Danihelka. "Neural turing machines." *arXiv preprint arXiv:1410.5401* (2014).

Addressing mechanism in NTM



e_t, a_t

Memory writing

$$\tilde{\mathbf{M}}_t(i) \longleftarrow \mathbf{M}_{t-1}(i)\left[\mathbf{1} - w_t(i)\mathbf{e}_t\right]$$

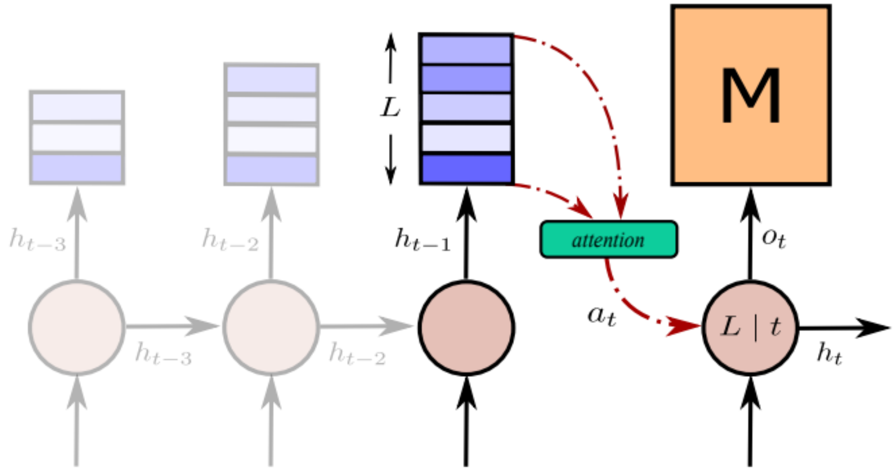
$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t.$$

Memory reading

$$\mathbf{r}_t \longleftarrow \sum_i w_t(i)\mathbf{M}_t(i),$$

Optimal memory writing for memorization

- Simple finding: writing too often deteriorates memory content (not retainable)
- Given input sequence of length T and only D writes, *when should we write to the memory?*



Theorem 3. *Given D memory slots, a sequence with length T, a decay rate $0 < \lambda \leq 1$, then the optimal intervals $\{l_i \in \mathbb{R}^+\}_{i=1}^{D+1}$ satisfying $T = \sum_{i=1}^{D+1} l_i$ such that the lower bound on the average contribution $I_\lambda = \frac{C}{T} \sum_{i=1}^{D+1} f_\lambda(l_i)$ is maximized are the following:*

$$l_1 = l_2 = ... = l_{D+1} = \frac{T}{D+1} \quad (7)$$

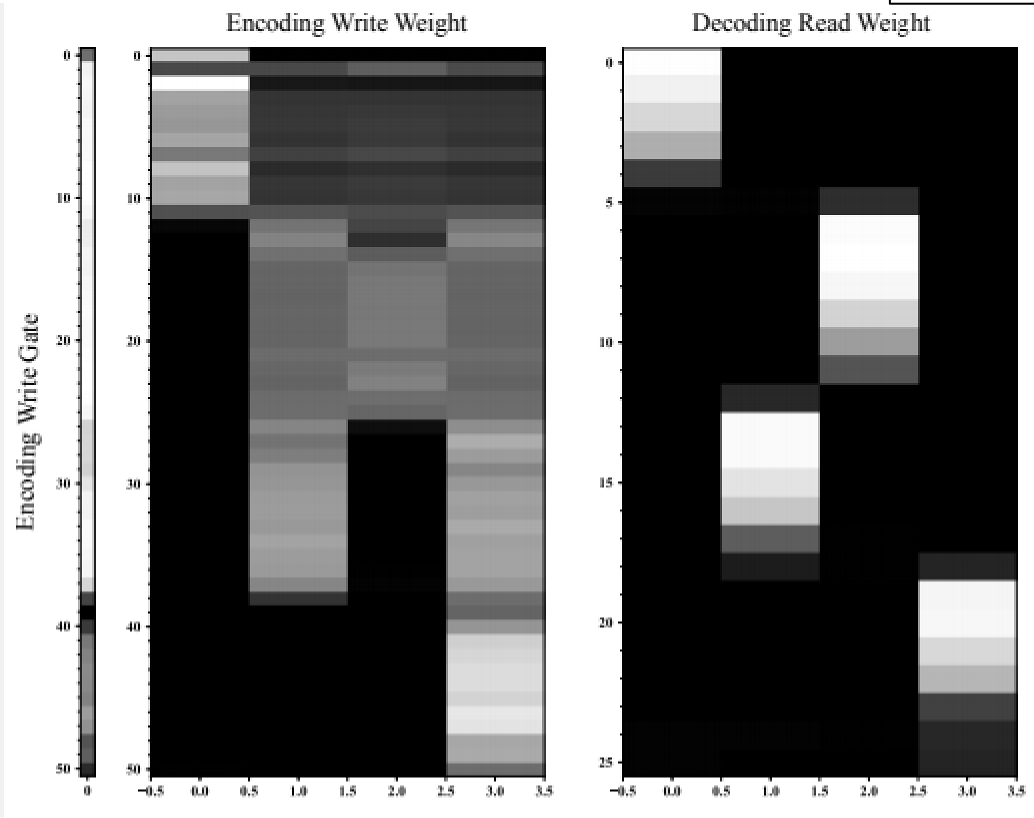
Uniform writing is optimal for memorization

Le, Hung, Truyen Tran, and Svetha Venkatesh. "Learning to Remember More with Less Memorization." In *International Conference on Learning Representations*. 2018.

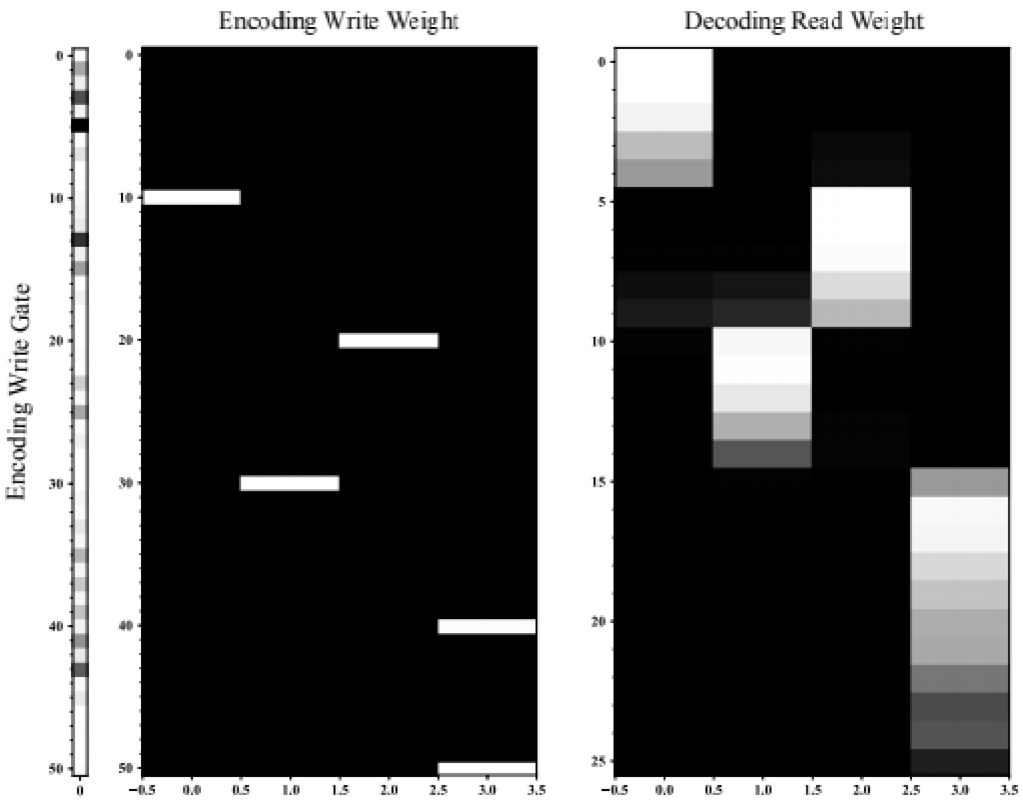
Better memorization means better algorithmic reasoning

Max	$x_1 x_2 \ldots x_T$	$\max(x_1, x_2) \max(x_3, x_4) \ldots \max(x_{T-1}, x_T)$

T=50, D=5



Regular



Uniform (cached)

Memory of independent entities

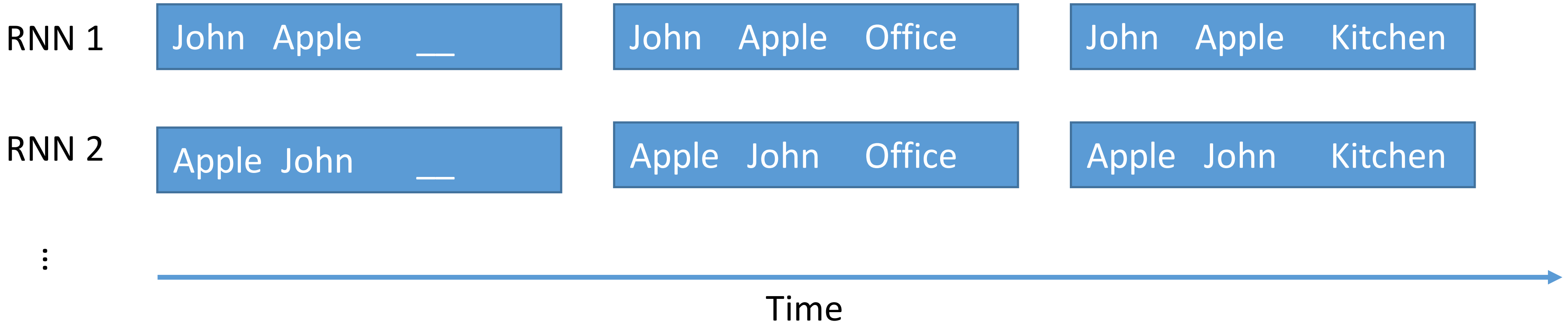
Weston, Jason, Bordes, Antoine, Chopra, Sumit, and Mikolov, Tomas.
Towards ai-complete question answering: A set of prerequisite toy tasks. CoRR, abs/1502.05698, 2015.

- Each slot store one or some entities
- Memory writing is done separately for each memory slot

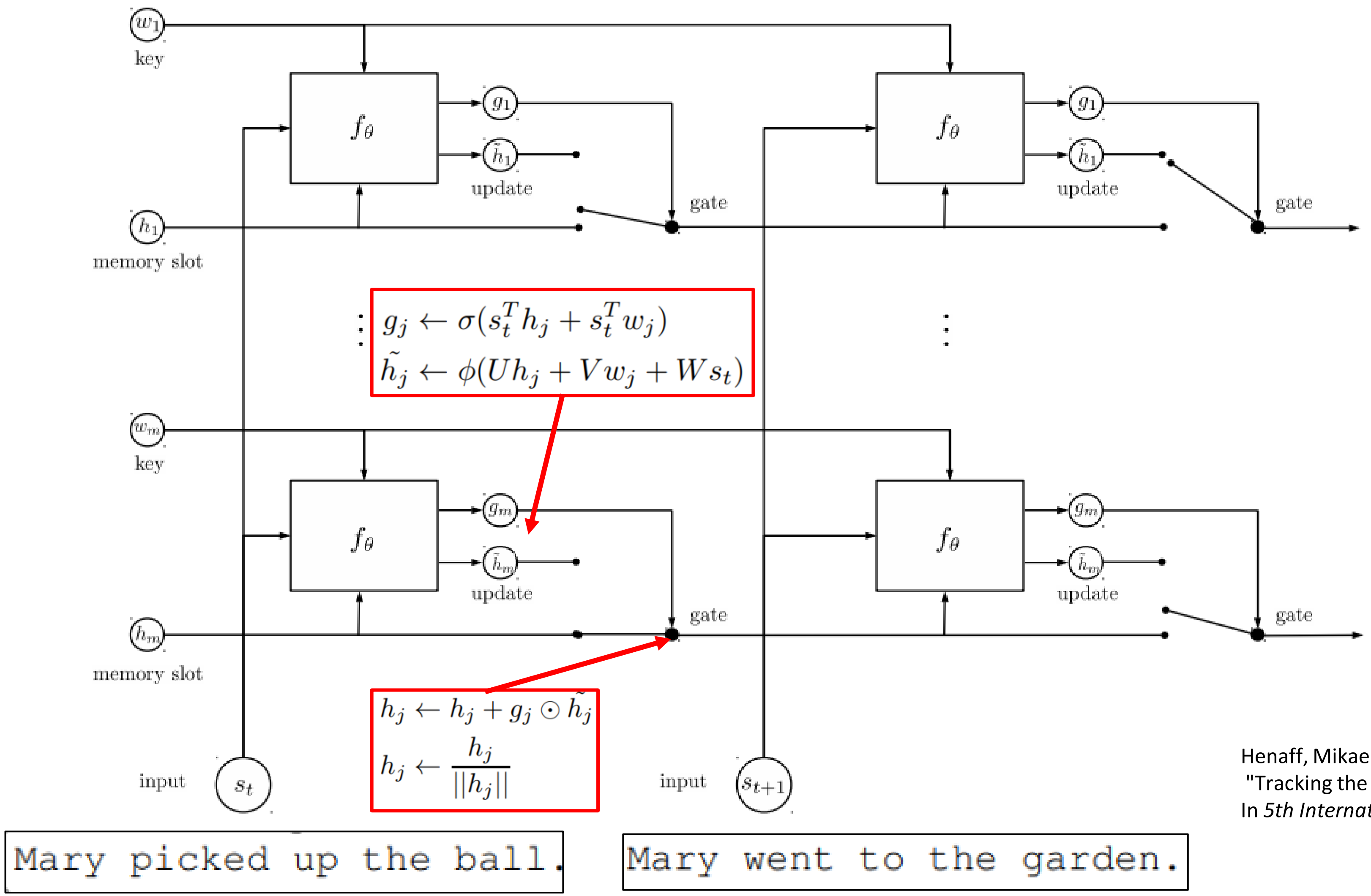
→each slot maintains the life of one or more entities

Task 3: Three Supporting Facts
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

- The memory is a set of N parallel RNNs



Recurrent entity network



"Where is the ball?"

$$p_j = \mathrm{Softmax}(q^T h_j)$$

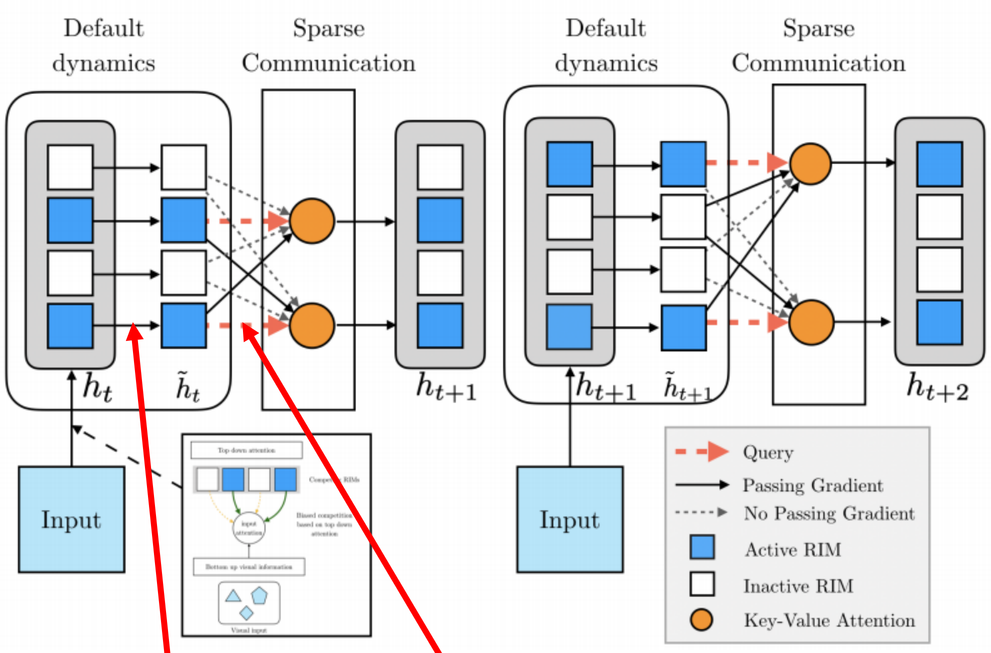
$$u = \sum_j p_j h_j$$

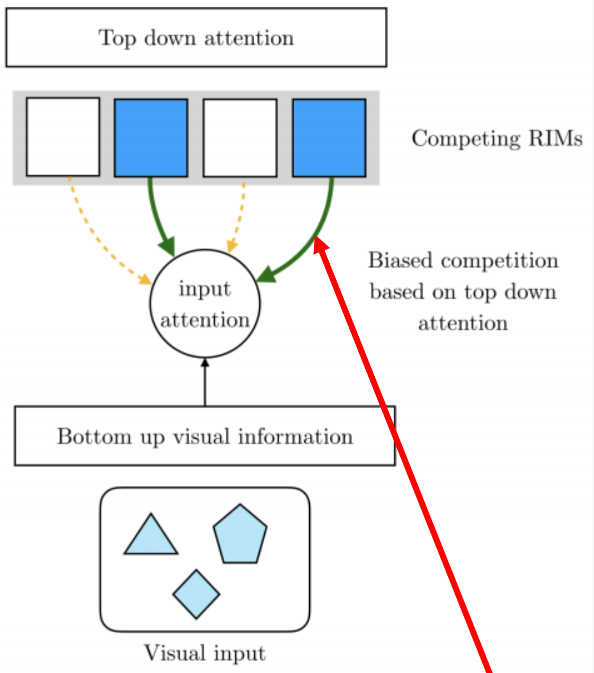
$$y = R\phi(q + Hu)$$

Garden

Henaff, Mikael, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. "Tracking the world state with recurrent entity networks." In *5th International Conference on Learning Representations, ICLR 2017*. 2017.

Recurrent Independent Mechanisms





$$\tilde{h}_{t,k} = D_k(h_{t,k}) = LSTM(h_{t,k}, A_k^{(in)}; \theta_k^{(D)}) \quad \forall k \in \mathcal{S}_t$$

$$A_k^{(in)} = \text{softmax}\left(\frac{h_t W_k^q (XW^e)^T}{\sqrt{d_e}}\right) XW^v,$$

$$Q_{t,k} = \tilde{W}_k^q \tilde{h}_{t,k}, \forall k \in \mathcal{S}_t \qquad K_{t,k} = \tilde{W}_k^e \tilde{h}_{t,k}, \forall k \qquad V_{t,k} = \tilde{W}_k^v \tilde{h}_{t,k}, \forall k$$

$$h_{t+1,k} = \text{softmax}\left(\frac{Q_{t,k}(K_{t,:})^T}{\sqrt{d_e}}\right) V_{t,:} + \tilde{h}_{t,k} \forall k \in \mathcal{S}_t.$$

Goyal, Anirudh, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. "Recurrent independent mechanisms." ICLR21.

Relational memory

Graph memory

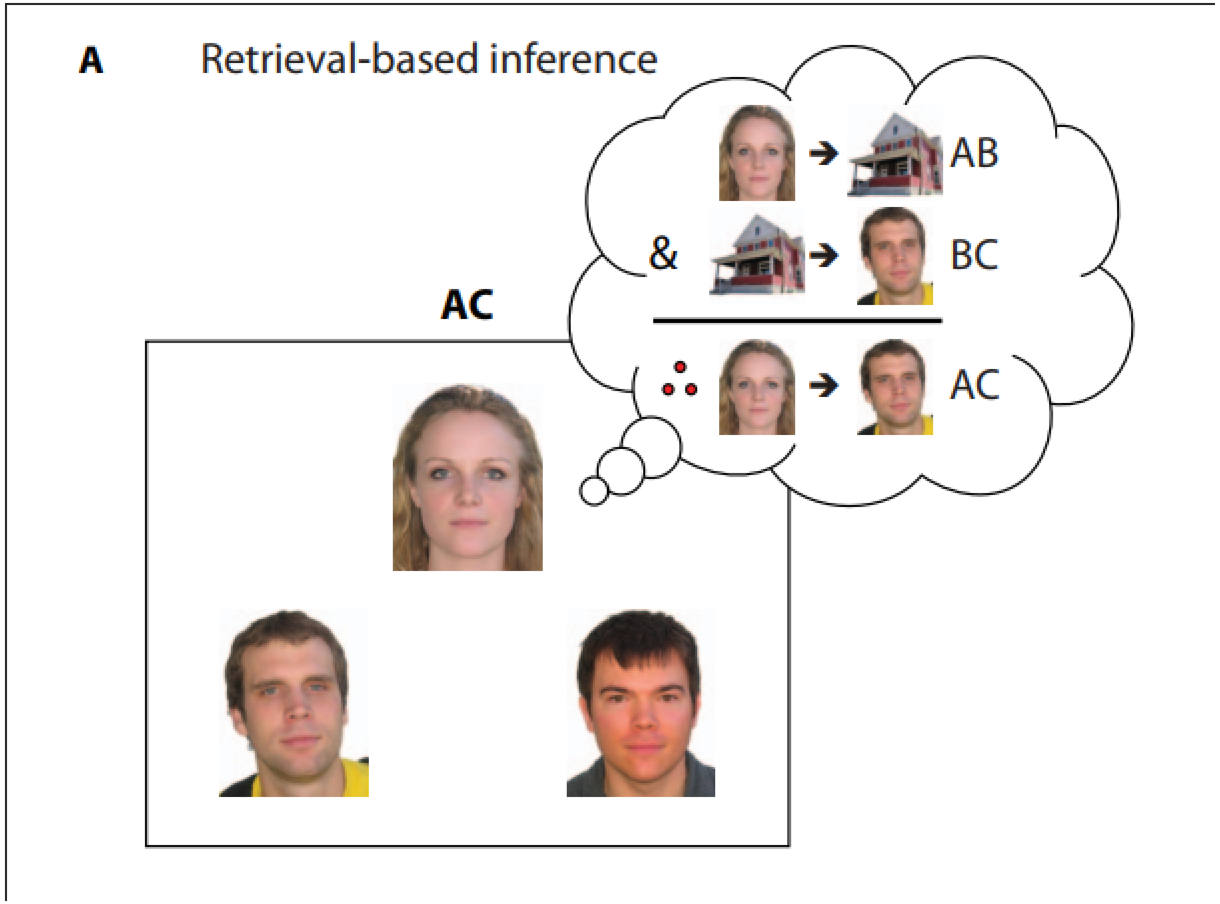
Tensor memory

Motivation for relational memory: item memory is weak at recognizing relationships

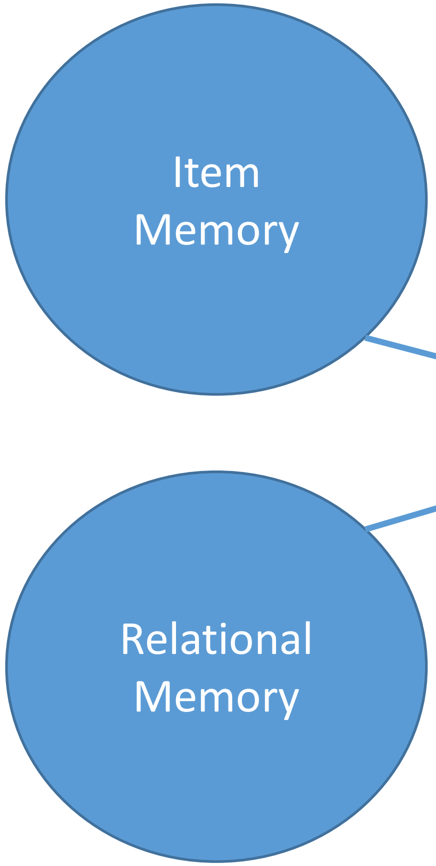
Item Memory

- Store and retrieve individual items
- Relate pair of items of the same time step
- Fail to relate temporally distant items

$$\hat{\mathbf{M}} = \sum_{k=1}^{q} \mathbf{b}_k \mathbf{a}_k^T$$

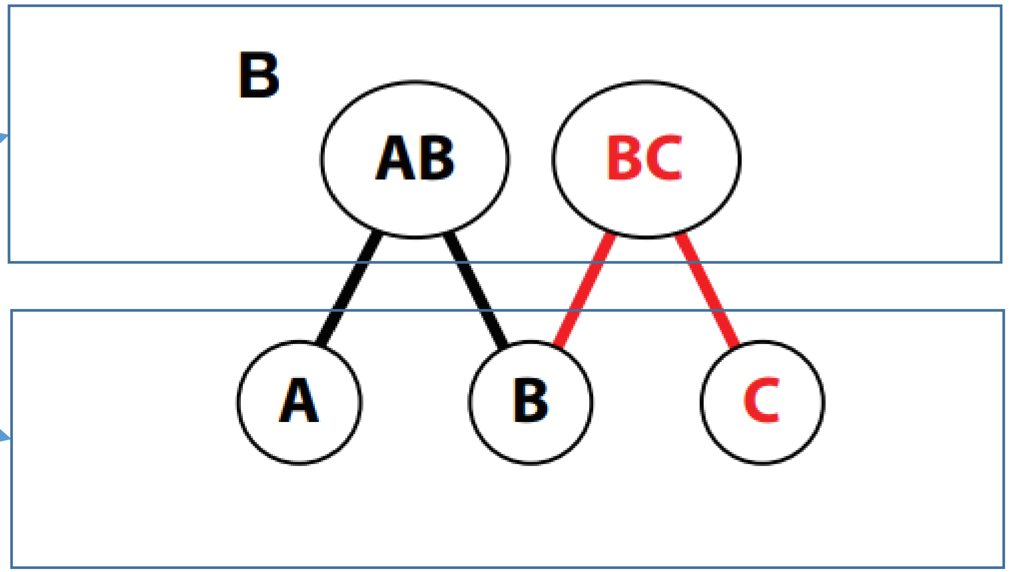


Dual process in memory



- Store items
- Simple, low-order
- System 1

- Store relationships between items
- Complicated, high-order
- System 2



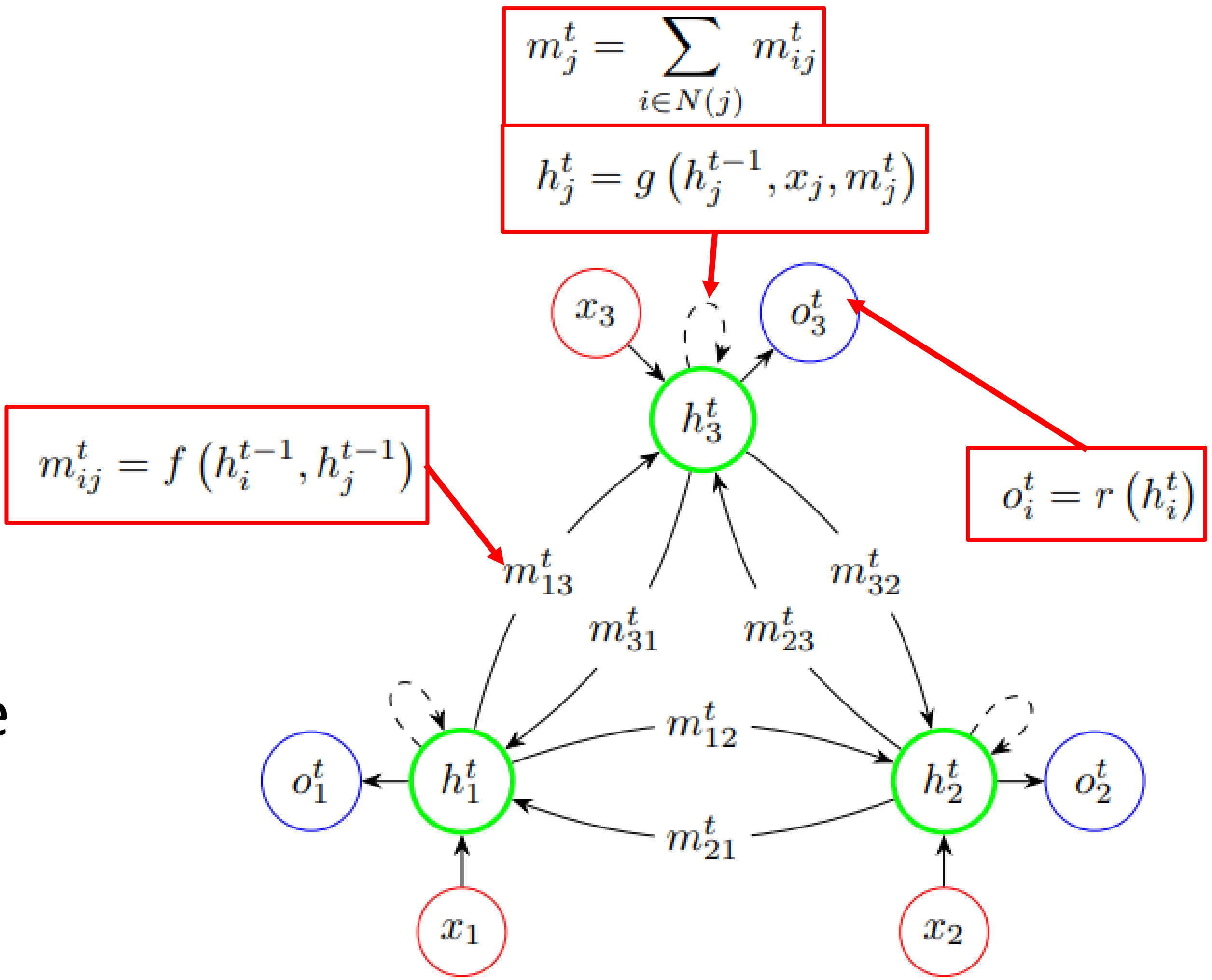
Howard Eichenbaum, *Memory, amnesia, and the hippocampal system* (MIT press, 1993).
Alex Konkel and Neal J Cohen, "Relational memory and the hippocampus: representations and methods", *Frontiers in neuroscience* 3 (2009).

Memory as graph

- Memory is a static graph with fixed nodes and edges
- Relationship is somehow

known

- Each memory node stores the state of the graph's node
- Write to node via message passing
- Read from node via MLP



Palm, Rasmus Berg, Ulrich Paquet, and Ole Winther. "Recurrent Relational Networks." In *NeurIPS*. 2018.

bAbI

$$x_i = \mathrm{MLP}(\mathrm{concat}(\mathrm{last}(\mathrm{LSTM}_S(s_i)), \mathrm{last}(\mathrm{LSTM}_Q(q)), \mathrm{onehot}(p_i + o)))$$

Node

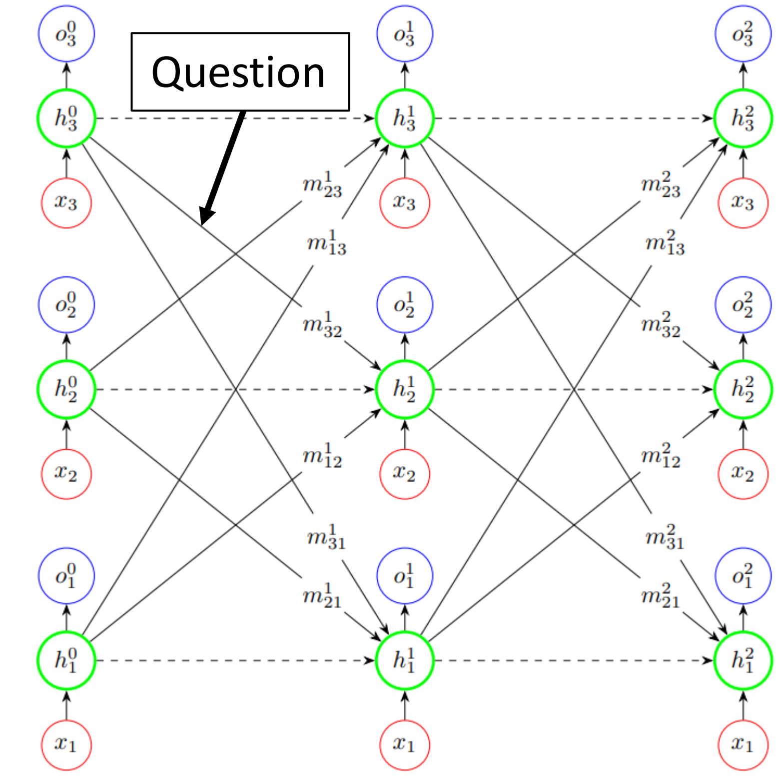
Edge

Fact 1

Fact 2

Fact 3

Question



$$o^t = r\left(\sum_i h_i^t\right)$$

Answer

$$m_{ij}^t = f\left(h_i^{t-1}, h_j^{t-1}, e_{ij}\right)$$

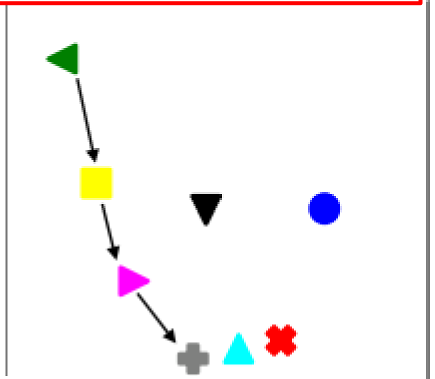
Method	N	Mean Error (%)	Failed tasks (err. >5%)
*RRN** (this work)	15	**0.46 ± 0.77**	**0.13 ± 0.35**

CLEVER

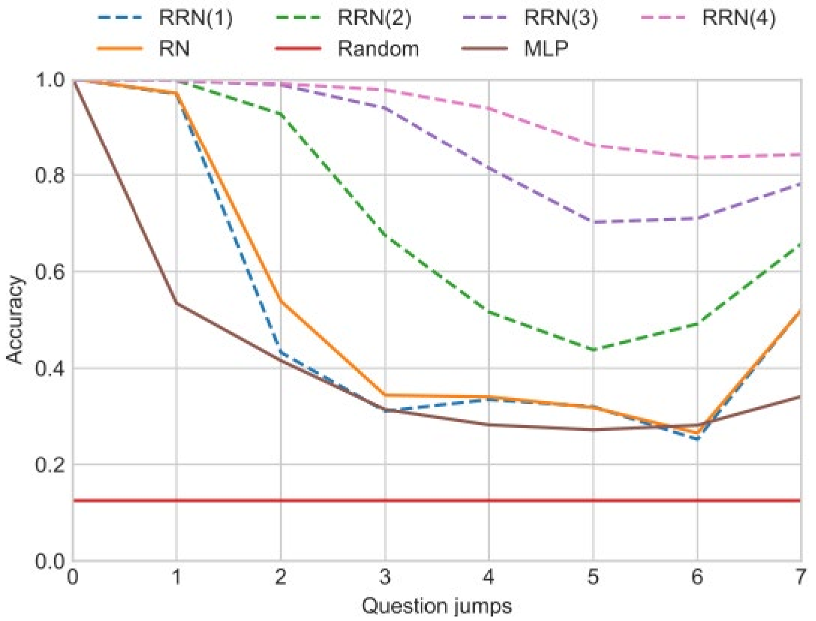
$$o_i = \mathrm{concat}(p_i, \mathrm{onehot}(c_i), \mathrm{onehot}(m_i))$$
$$q = \mathrm{concat}(\mathrm{onehot}(s), \mathrm{onehot}(n))$$
$$x_i = \mathrm{MLP}(\mathrm{concat}(o_i, q))$$

Node
(colour, shape. position)

Edge
(distance)

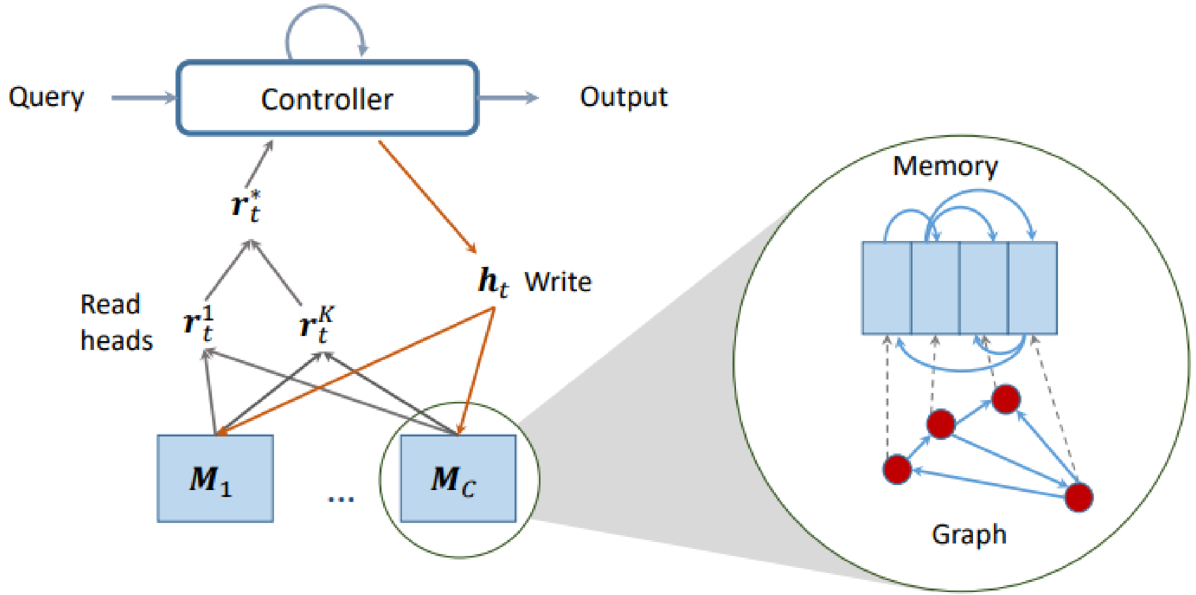


solution to the question: "green, 3 jumps", which is "plus",



Memory of graphs access conditioned on query

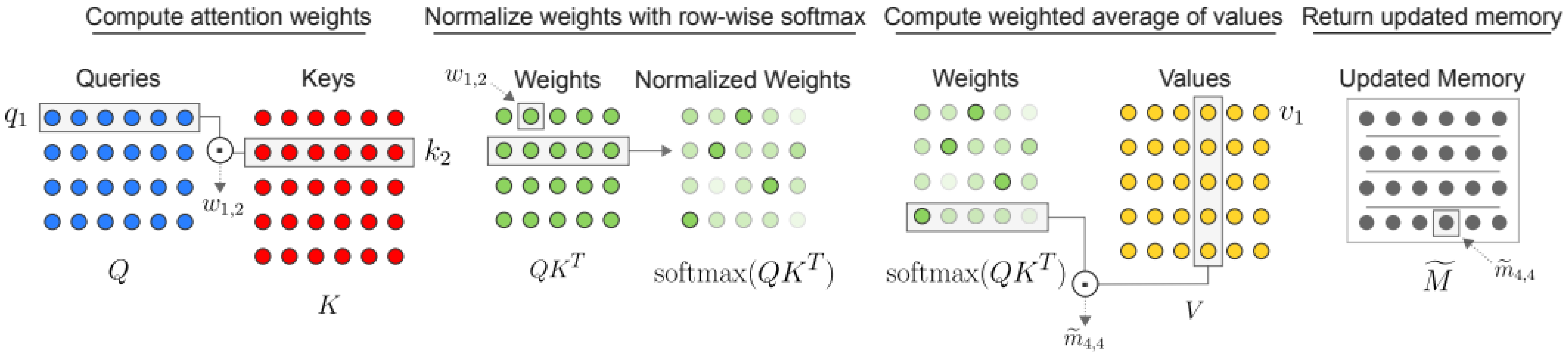
- Encode multiple graphs, each graph is stored in a set of memory row
- For each graph, the controller read/write to the memory:
  - Read uses content-based attention
  - Write use message passing
- Aggregate read vectors from all graphs to create output



Pham, Trang, Truyen Tran, and Svetha Venkatesh. "Relational dynamic memory networks." *arXiv preprint arXiv:1808.04247* (2018).

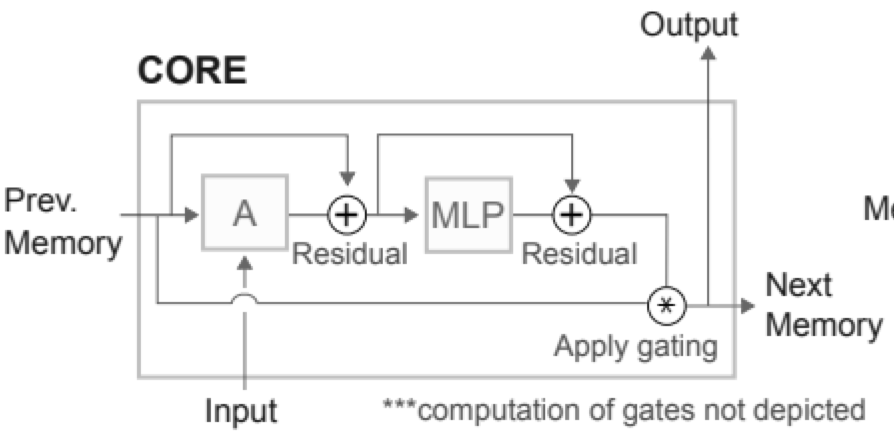
Capturing relationship can be done via memory slot interactions using attention

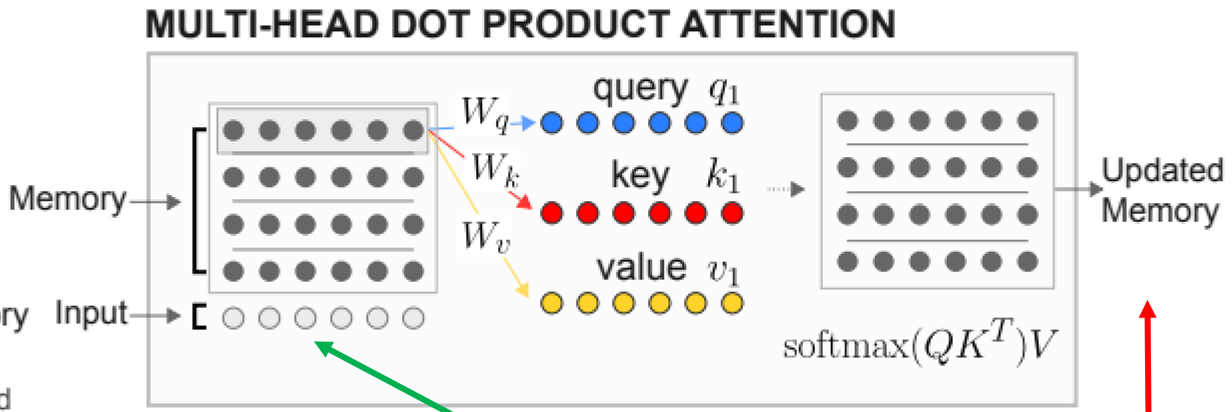
- Graph memory needs customization to an explicit design of nodes and edges
- Can we automatically learns structure with a 2d tensor memory?
- Capture relationship: each slot interacts with all other slots (self-attention)



Santoro, Adam, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Théophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap. "Relational recurrent neural networks." In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pp. 7310-7321. 2018.

Relational Memory Core (RMC) operation





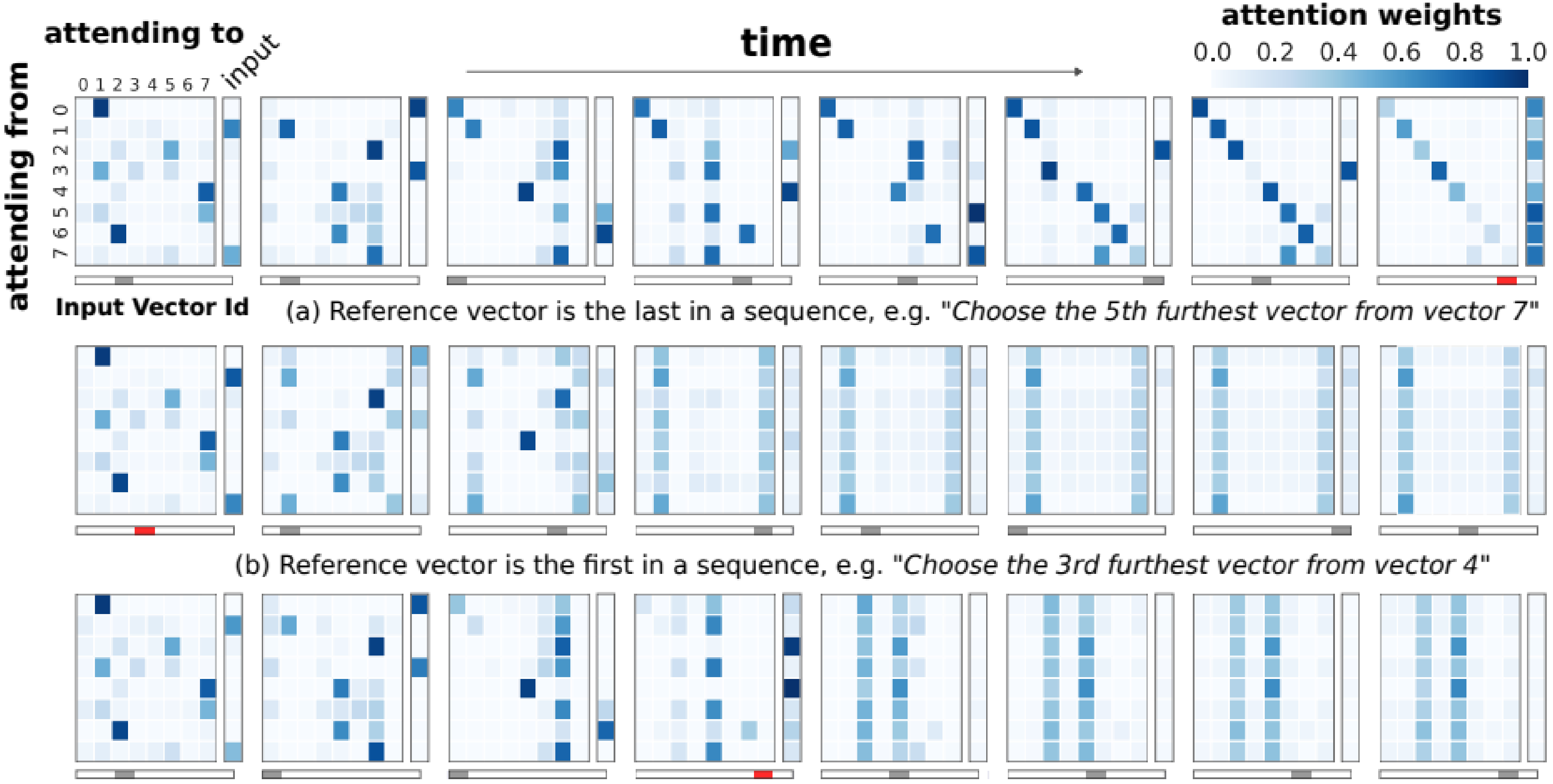
$$s_{i,t} = (h_{i,t-1}, m_{i,t-1})$$
$$f_{i,t} = W^f x_t + U^f h_{i,t-1} + b^f$$
$$i_{i,t} = W^i x_t + U^i h_{i,t-1} + b^i$$
$$o_{i,t} = W^o x_t + U^o h_{i,t-1} + b^o$$
$$m_{i,t} = \sigma(f_{i,t} + \tilde{b}^f) \circ m_{i,t-1} + \sigma(i_{i,t}) \circ \underbrace{g_\psi(\tilde{m}_{i,t})}$$
$$h_{i,t} = \sigma(o_{i,t}) \circ \tanh(m_{i,t})$$
$$s_{i,t+1} = (m_{i,t}, h_{i,t})$$

row/memory-wise MLP with layer normalisation

RNN-like Interface

$$\boxed{\widetilde{M} = \mathrm{softmax}\left(\frac{MW^q(\boxed{[M;x]}W^k)^T}{\sqrt{d^k}}\right)[M;x]W^v,}$$

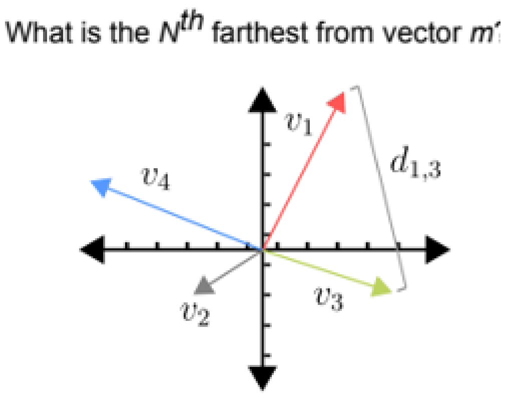
Allowing pair-wise interactions can answer questions on temporal relationship



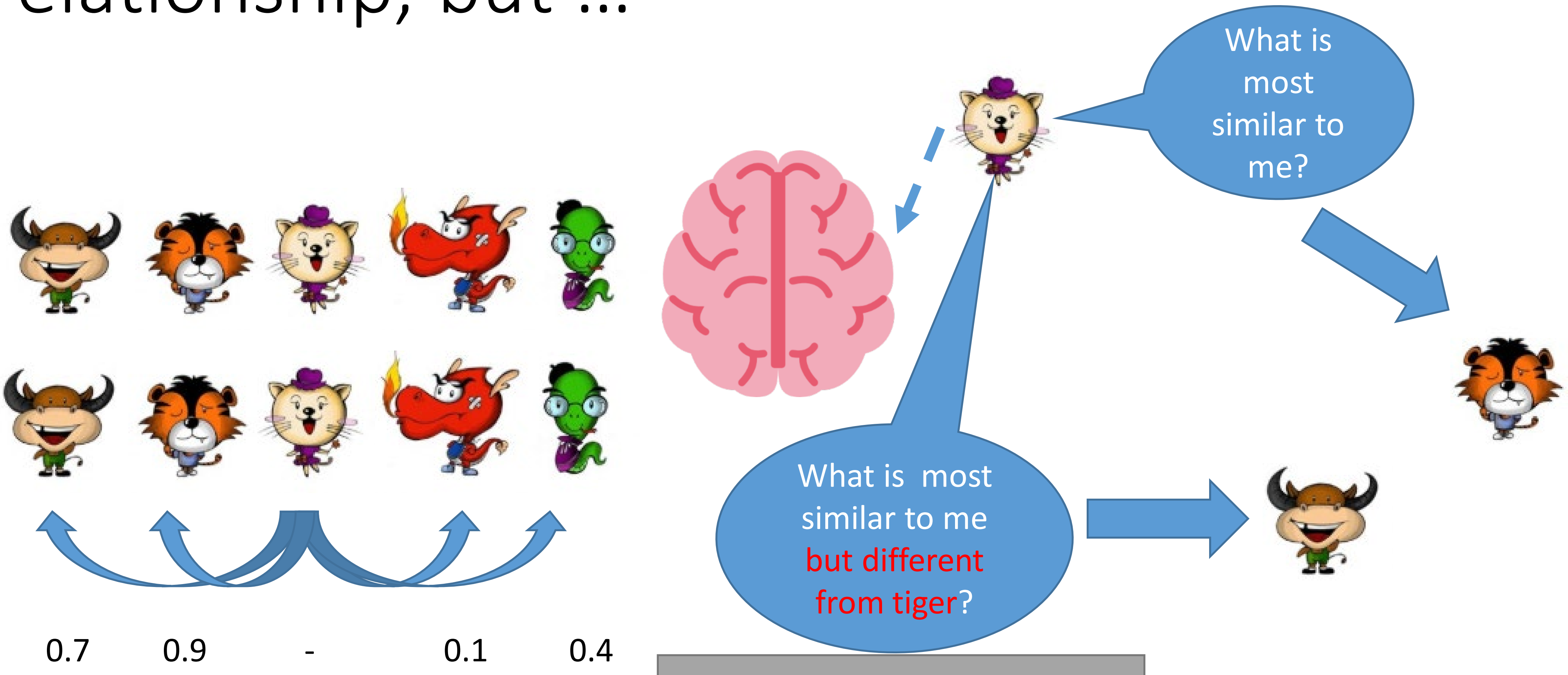
(a) Reference vector is the last in a sequence, e.g. "*Choose the 5th furthest vector from vector 7*"

(b) Reference vector is the first in a sequence, e.g. "*Choose the 3rd furthest vector from vector 4*"

(c) Reference vector comes in the middle of a sequence, e.g. "*Choose the 6th furthest vector from vector 6*"



Dot product attention works for simple relationship, but …



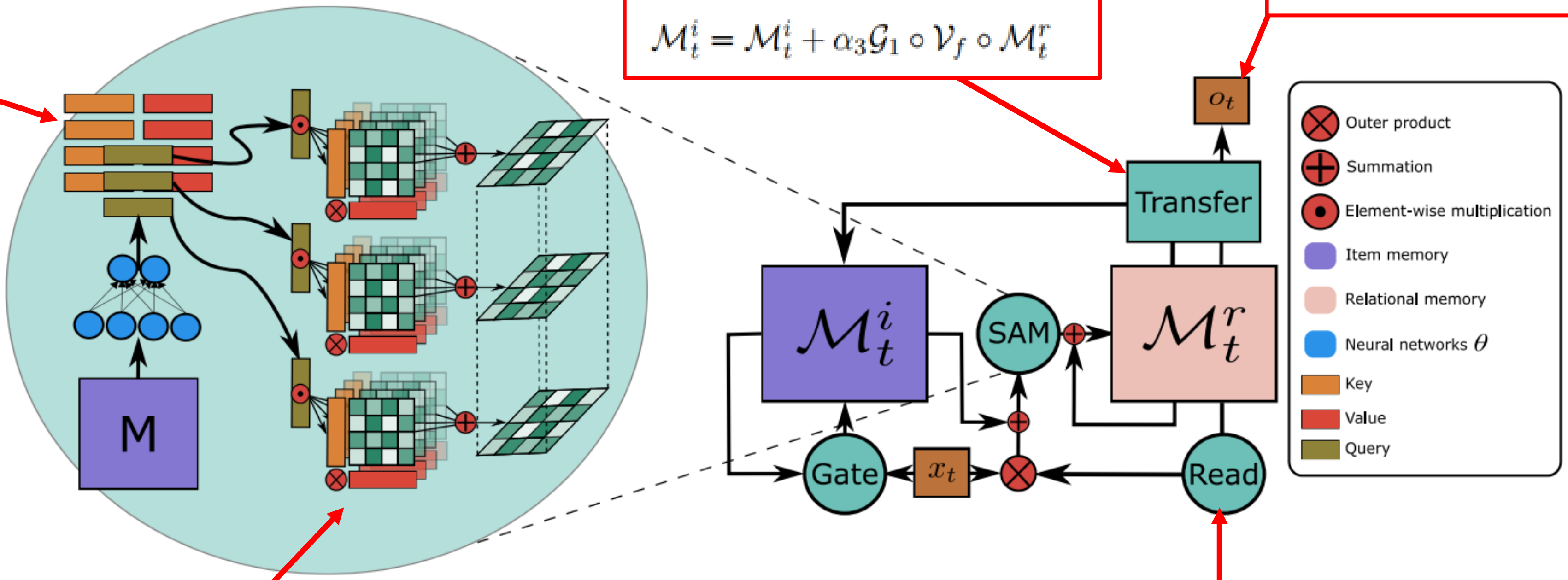
For hard relationship, scalar representation is limited

Self-attentive associative memory

$$M_q = \mathcal{LN}\left(W_q M\right)$$
$$M_k = \mathcal{LN}\left(W_k M\right)$$
$$M_v = \mathcal{LN}\left(W_v M\right)$$

$$\mathcal{M}_t^i = \mathcal{M}_t^i + \alpha_3 \mathcal{G}_1 \circ \mathcal{V}_f \circ \mathcal{M}_t^r$$

$$o_t = \mathcal{G}_3 \circ \mathcal{V}_l \circ \mathcal{G}_2 \circ \mathcal{V}_l \circ \mathcal{M}_t^r$$



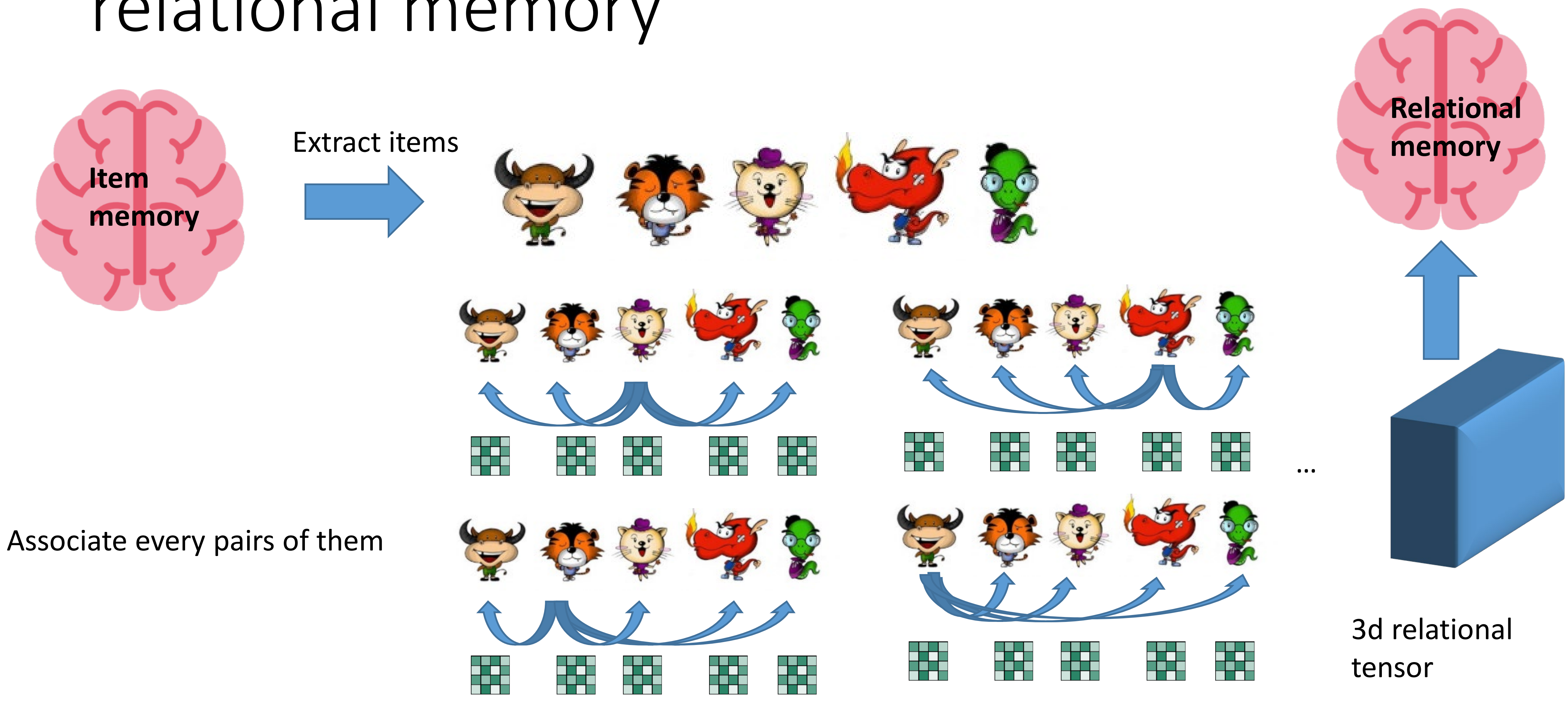
$$\mathrm{SAM}_\theta\left(M\right)[s] = A^{\otimes}\left(M_q[s], M_k, M_v\right) \quad (7)$$
$$= \sum_{j=1}^{n_{kv}} \mathcal{F}\left(M_q[s] \odot M_k[j]\right) \otimes M_v[j] \quad (8)$$

$$v_t^r = \mathrm{softmax}\left(f_3\left(x_t\right)^\top\right) \mathcal{M}_{t-1}^r f_2\left(x_t\right) \quad (11)$$

$$\mathcal{M}_t^r = \mathcal{M}_{t-1}^r + \alpha_1 \mathrm{SAM}_\theta\left(\mathcal{M}_t^i + \alpha_2 v_t^r \otimes f_2\left(x_t\right)\right) \quad (12)$$

Le, Hung, Truyen Tran, and Svetha Venkatesh. "Self-attentive associative memory." In *International Conference on Machine Learning*, pp. 5682-5691. PMLR, 2020.

Complicated relationship needs high-order relational memory



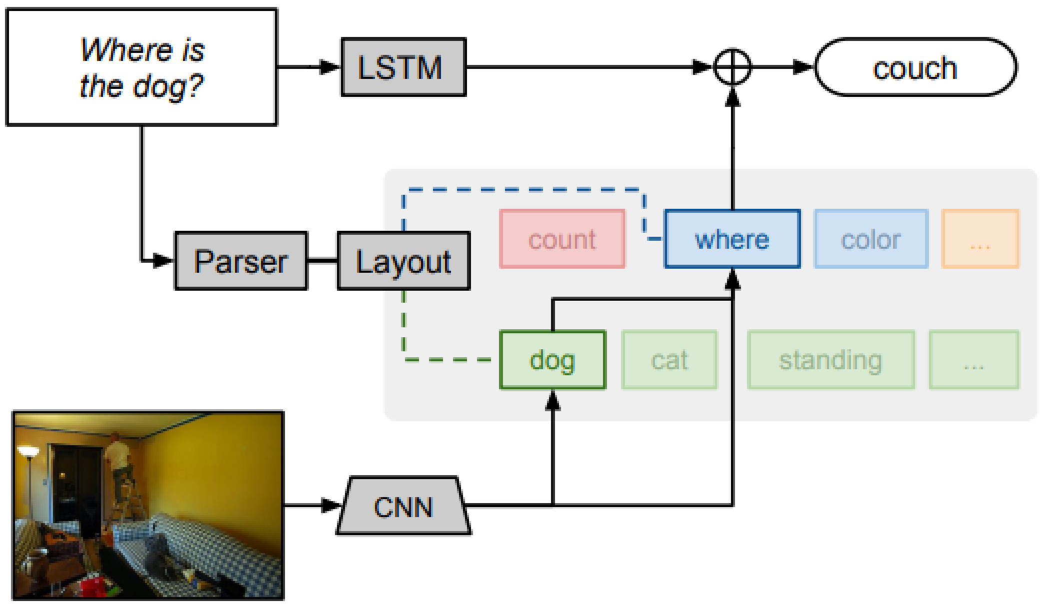
Program memory

Module memory

Stored-program memory

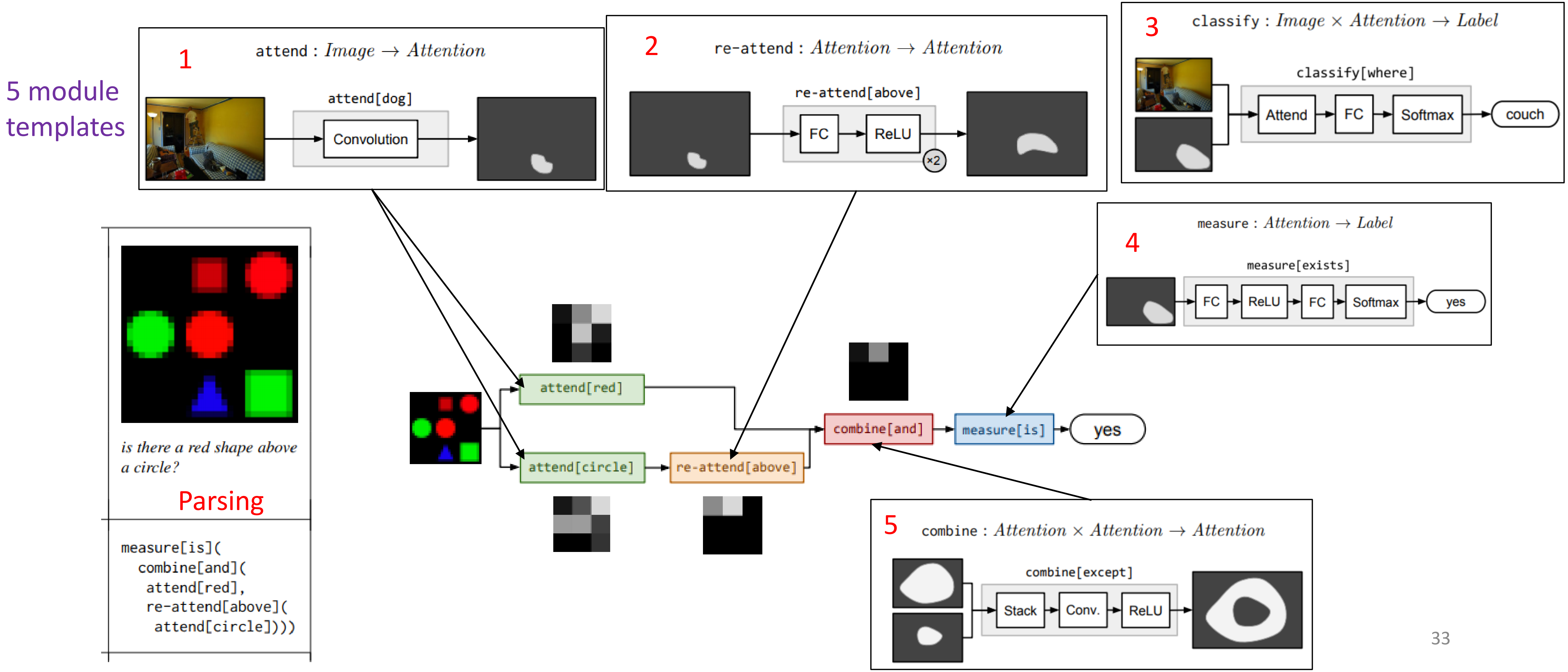
Predefining program for subtask

- A program designed for a task becomes a module
- Parse a question to module layout (order of program execution)
- Learn the weight of each module to master the task



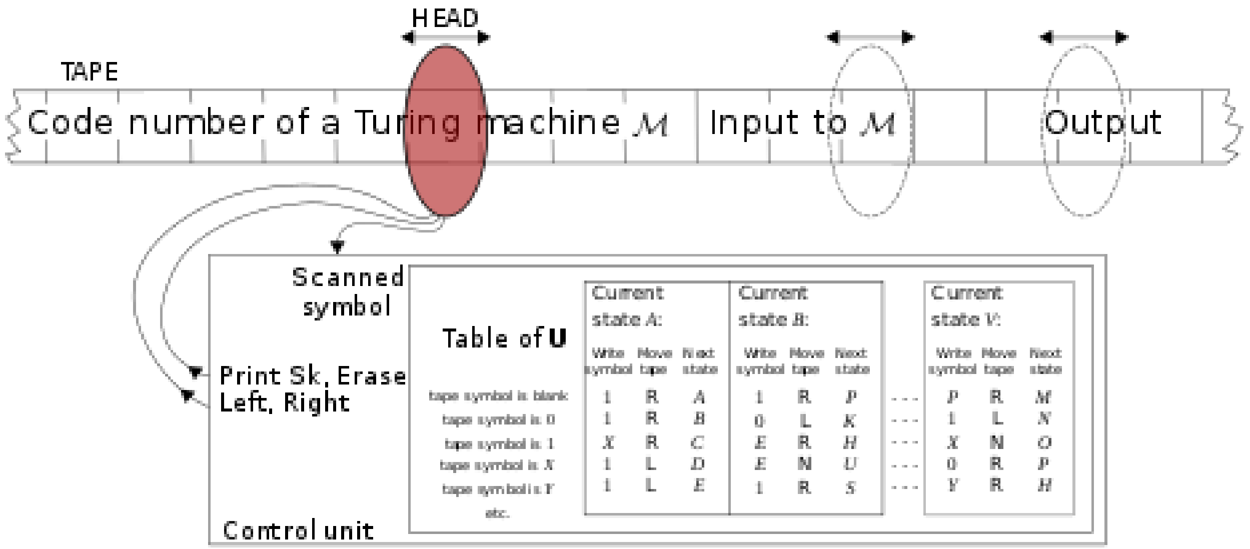
Andreas, Jacob, Marcus Rohrbach, Trevor Darrell, and Dan Klein. "Neural module networks." In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 39-48. 2016.

Program selection is based on parser, others are end2end trained



The most powerful memory is one that stores both program and data

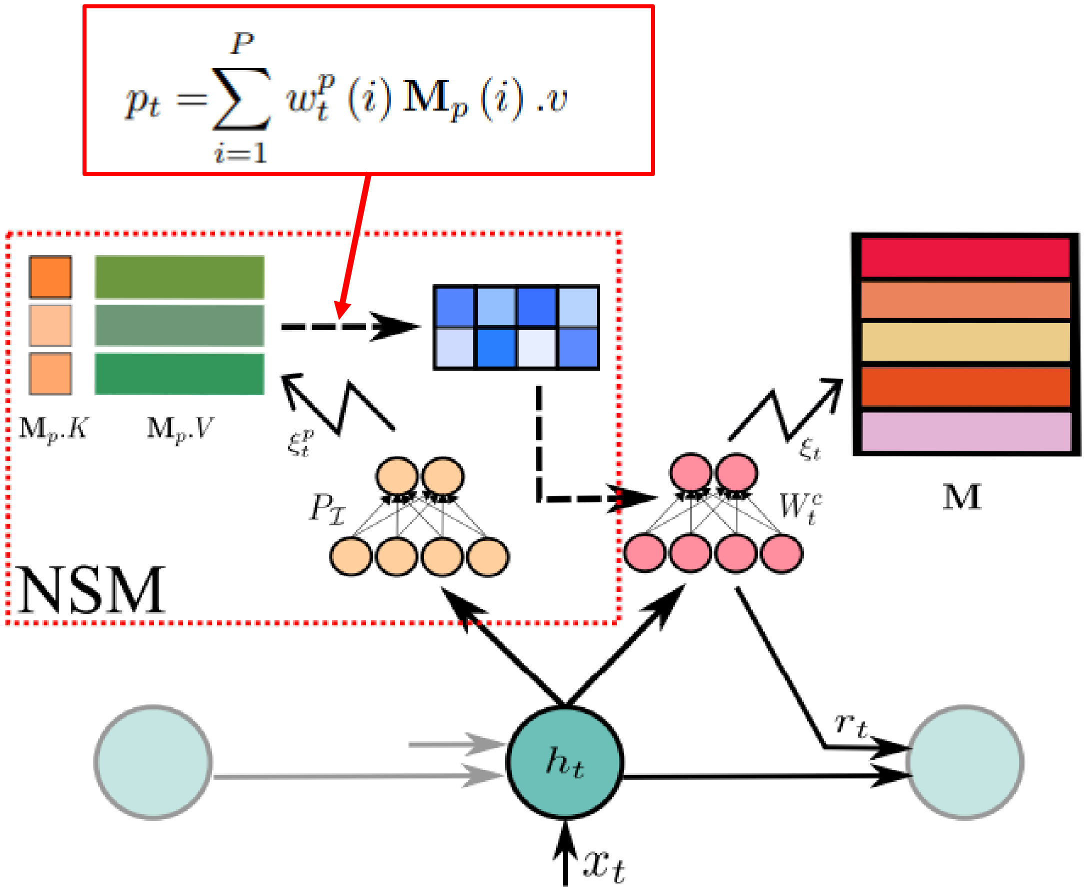
- Computer architecture: Universal Turing Machines/Harvard/VNM
- Stored-program principle
- Break a big task into subtasks, each can be handled by a TM/single purposed program stored in a <u>program memory</u>



https://en.wikipedia.org/

NUTM: Learn to select program (neural weight) via program attention

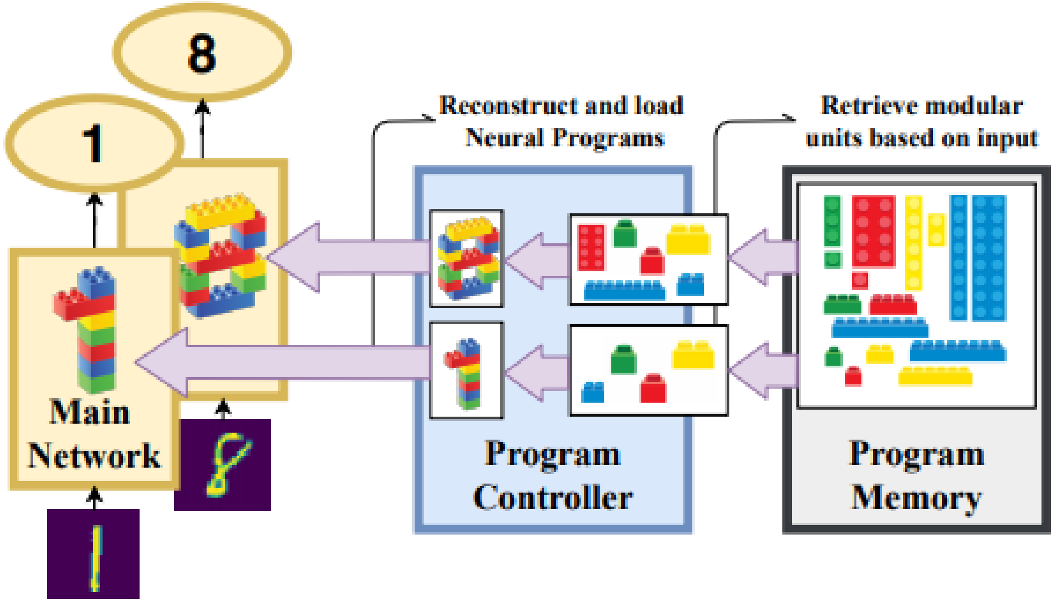
- Neural stored-program memory (NSM) stores key (the address) and values (the weight)
- The weight is selected and loaded to the controller of NTM
- The stored NTM weights and the weight of the NUTM is learnt end-to-end by backpropagation



Le, Hung, Truyen Tran, and Svetha Venkatesh. "Neural Stored-program Memory." In *International Conference on Learning Representations*. 2019.

Scaling with memory of mini-programs

- Prior, 1 program = 1 neural network (millions of parameters)
- Parameter inefficiency since the programs do not share common parameters
- Solution: store sharable mini-programs to compose infinite number of programs



it is analogous to building Lego structures corresponding to inputs from basic Lego bricks.

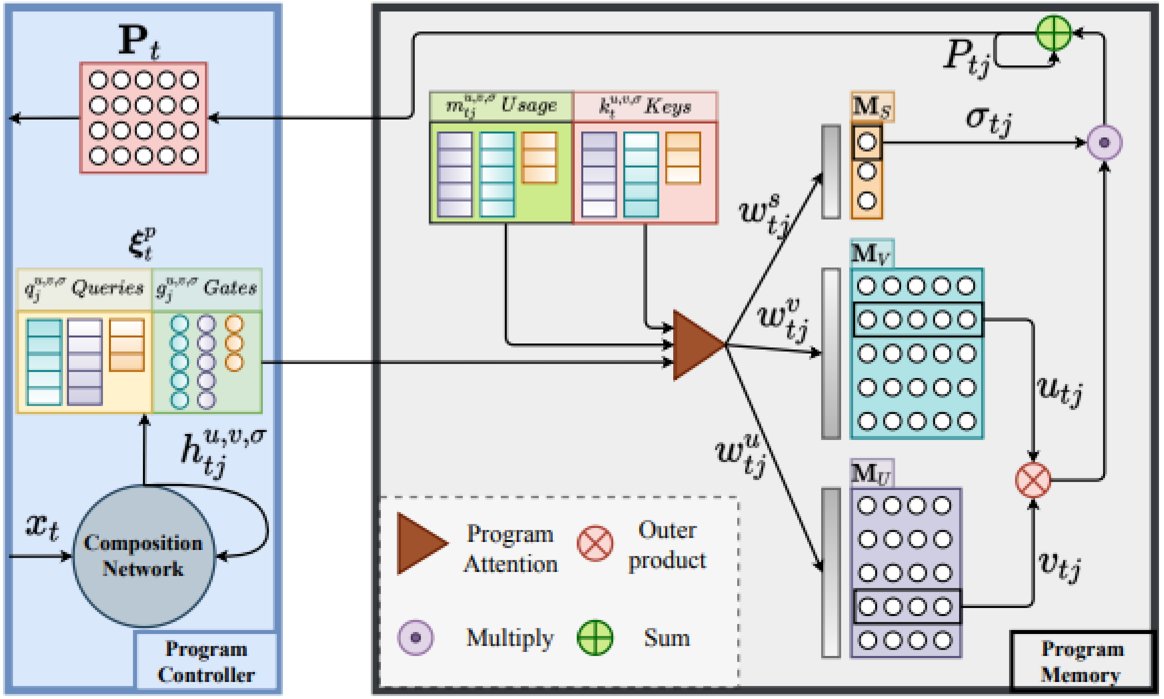
Recurrent program attention to retrieve singular components of a program

$$\mathbf{P}_t = \mathbf{USV}^T$$
$$= \sum_{n}^{r_m} \sigma_{tn} u_{tn} v_{tn}^{\top}$$

$$u_{tn} = \sum_{i=1}^{P_u} w_{tin}^{u} \mathbf{M}_U(i)$$

$$v_{tn} = \sum_{i=1}^{P_v} w_{tin}^{v} \mathbf{M}_V(i)$$

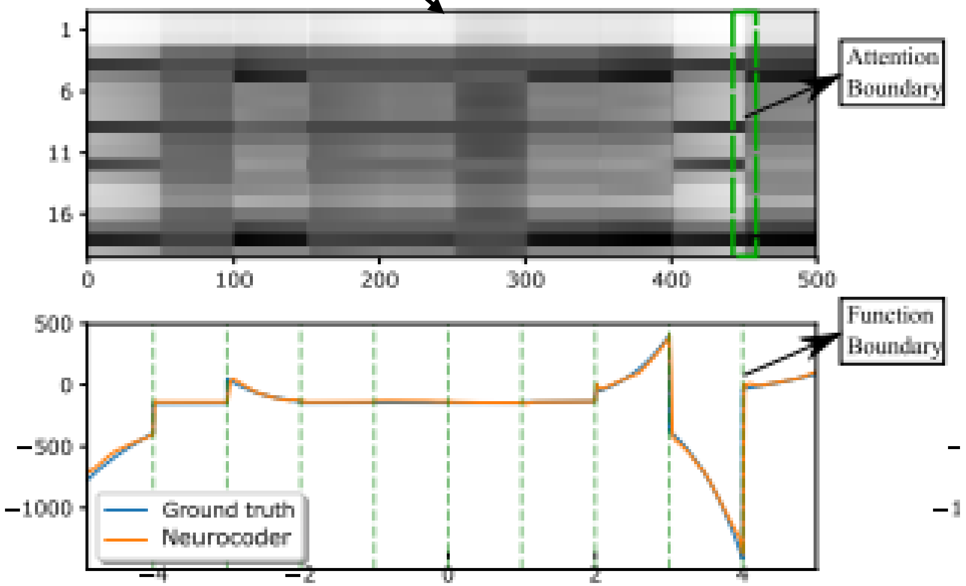
$$\sigma_{tn} = \begin{cases} \text{softplus}\left(\sum_{i=1}^{P_s} w_{tin}^{\sigma} \mathbf{M}_S(i)\right) & n = r_m \\ \sigma_{tn+1} + \text{softplus}\left(\sum_{i=1}^{P_s} w_{tin}^{\sigma} \mathbf{M}_S(i)\right) & n < r_m \end{cases}$$

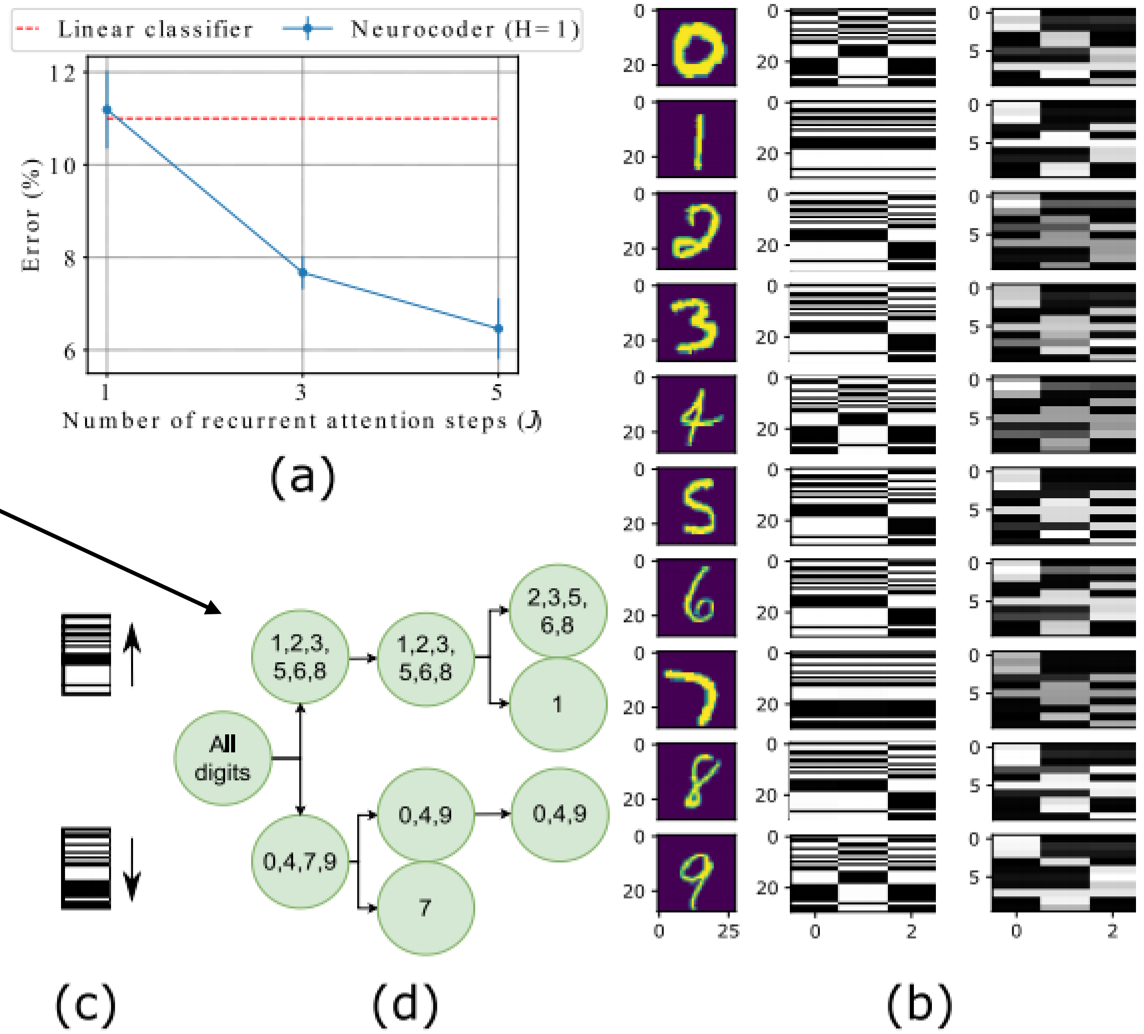


Le, Hung, and Svetha Venkatesh. "Neurocoder: Learning General-Purpose Computation Using Stored Neural Programs." *arXiv preprint arXiv:2009.11443* (2020).

Program attention is equivalent to binary decision tree reasoning

Recurrent program attention auto detects task boundary





QA

10. Combinatorics reasoning

RNN

MANN

GNN

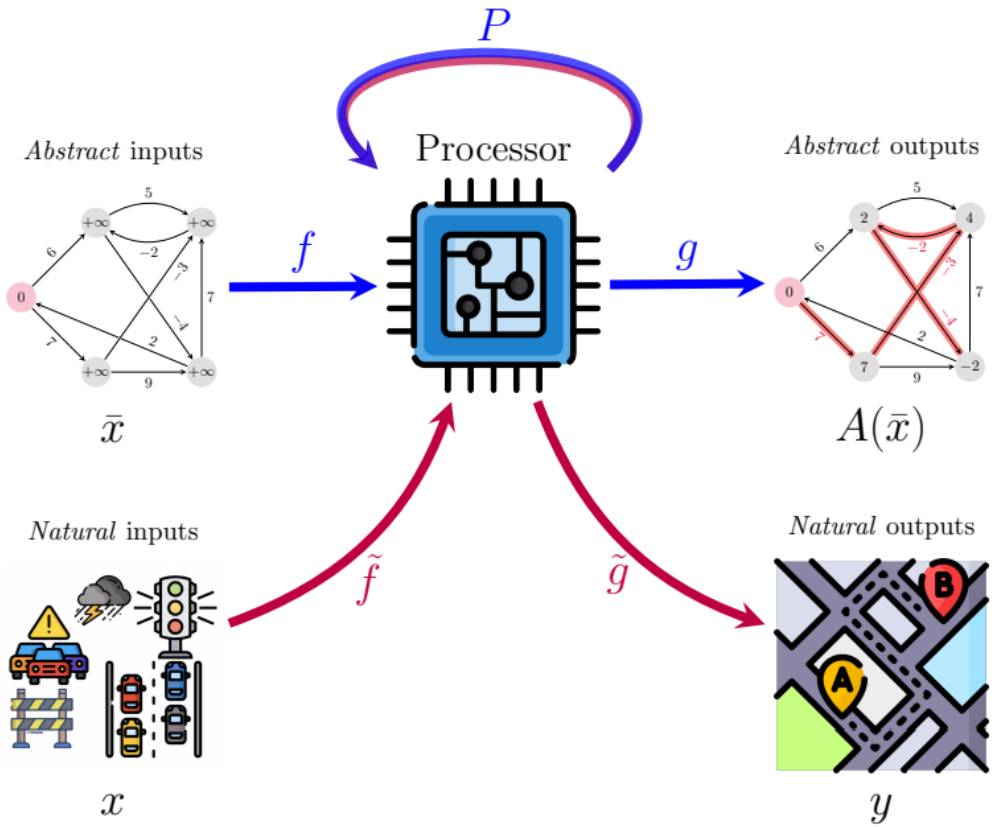
Transformer

Implement combinatorial algorithms with neural networks

Processor P:

(a) aligned with the computations of the target algorithm;
(b) operates by matrix multiplications, hence natively admits useful gradients;
(c) operates over high-dimensional latent spaces

Train neural processor P to imitate algorithm A



Generalizable
Inflexible

Noisy
High dimensional

Veličković, Petar, and Charles Blundell. "Neural Algorithmic Reasoning." *arXiv preprint arXiv:2105.02761* (2021).

Processor as RNN

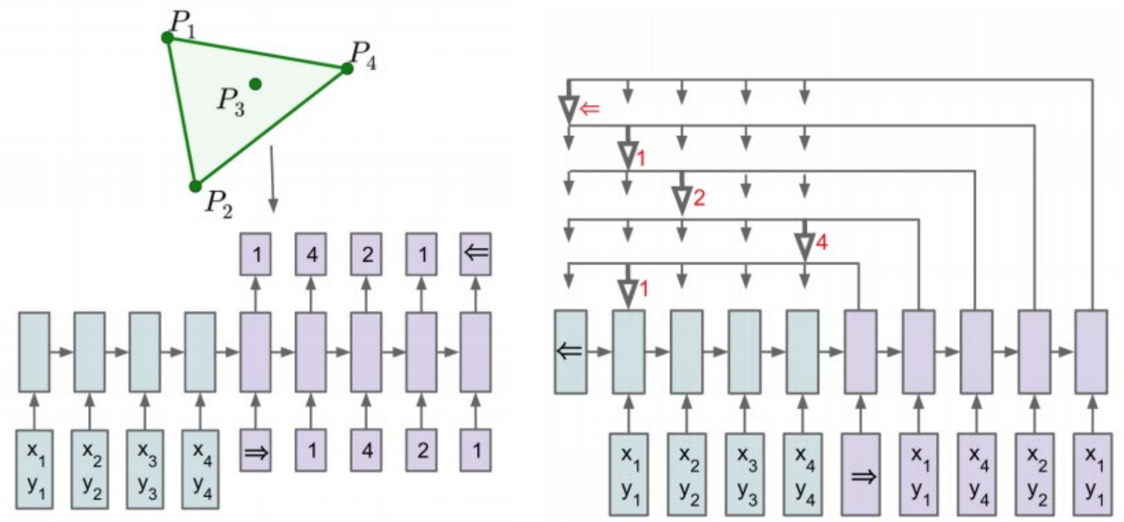
$$
\begin{aligned}
u^i_j &= v^T \tanh(W_1 e_j + W_2 d_i) \quad j \in (1, \ldots, n) \\
p(C_i | C_1, \ldots, C_{i-1}, \mathcal{P}) &= \mathrm{softmax}(u^i)
\end{aligned}
$$

- Do not assume knowing the structure of the input, input as a sequence

→not really reasonable, harder to generalize

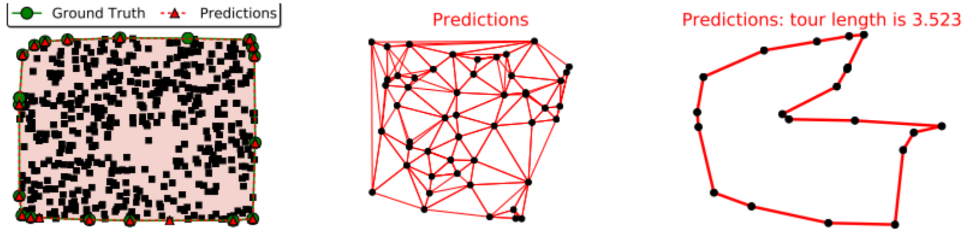
- RNN is Turing-complete → can simulate any algorithm
- But, it is not easy to learn the simulation from data (input-output)

→Pointer network



(a) Sequence-to-Sequence

(b) Ptr-Net



(d) Ptr-Net, m=5-50, n=500

(e) Ptr-Net , m=50, n=50

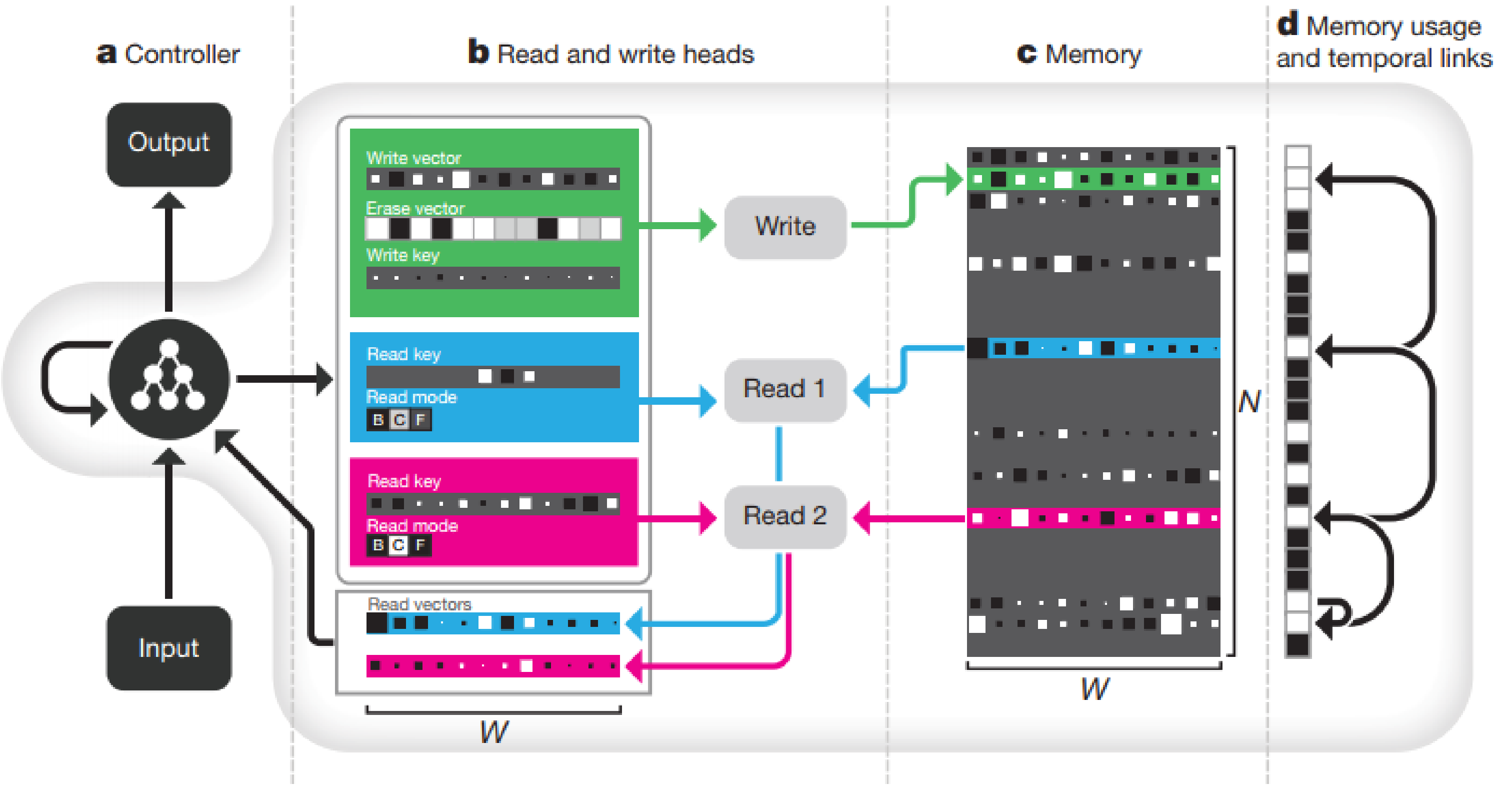
(f) Ptr-Net , m=5-20, n=20

Assume O(N) memory
And O(N^2) computation
N is the size of input

Vinyals, Oriol, Meire Fortunato, and Navdeep Jaitly. "Pointer networks." In *Proceedings of the 28th International Conference on Neural Information Processing Systems-Volume 2*, pp. 2692-2700. 2015.

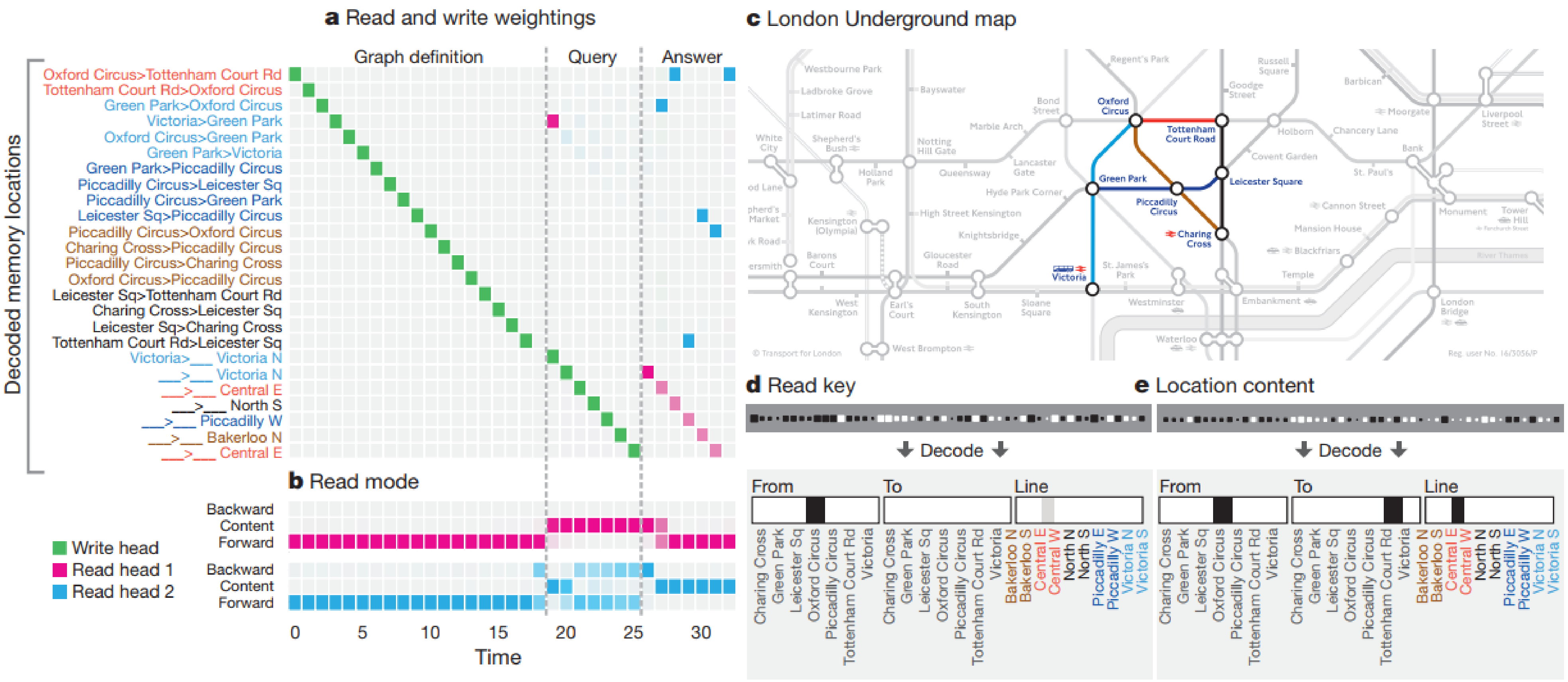
Processor as MANN

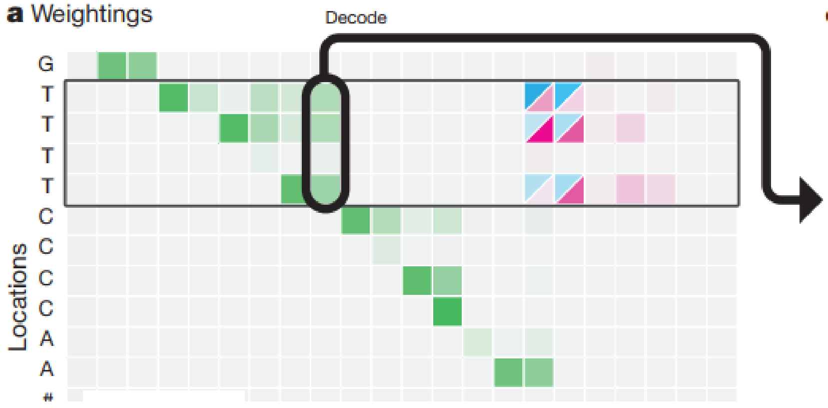
- MANN simulates neural computers or Turing machine→ ideal for implement algorithms
- Sequential input, no assumption on input structure
- Assume O(1) memory and O(N) computation

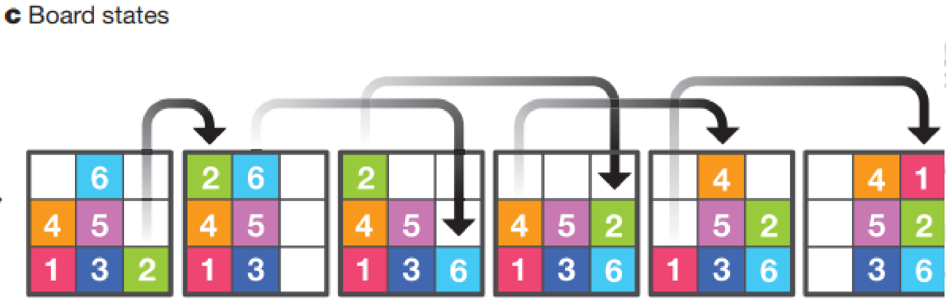


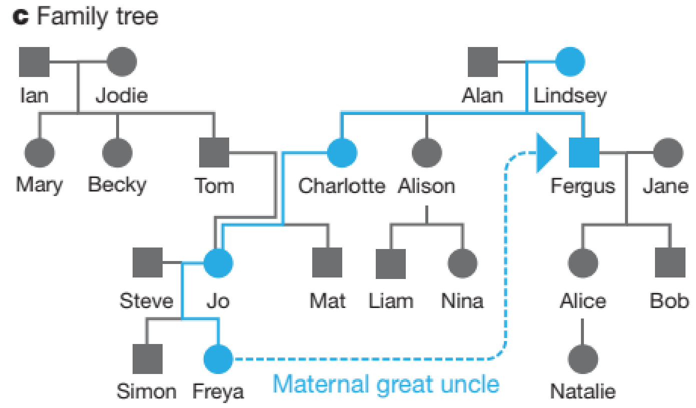
Graves, A., Wayne, G., Reynolds, M. *et al.* Hybrid computing using a neural network with dynamic external memory. *Nature* **538,** 471–476 (2016)

DNC: item memory for graph reasoning



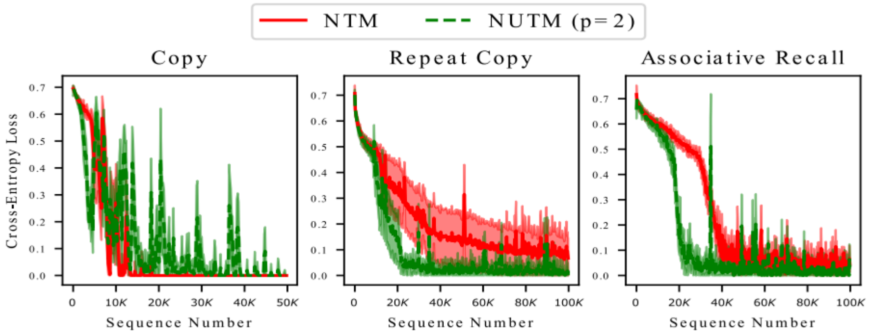


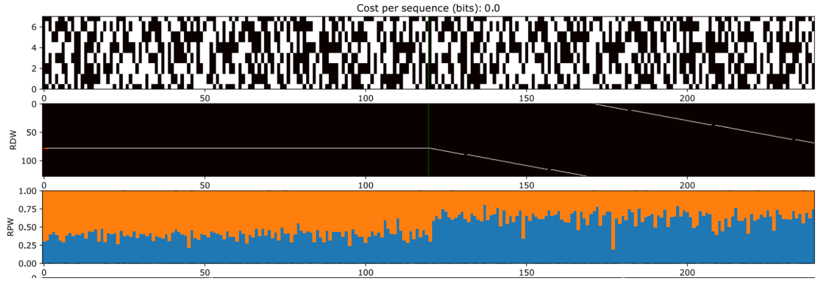


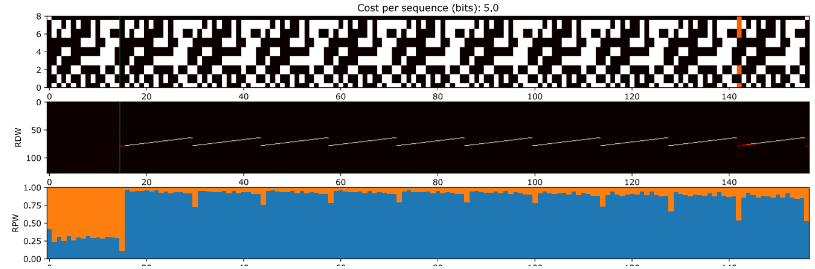


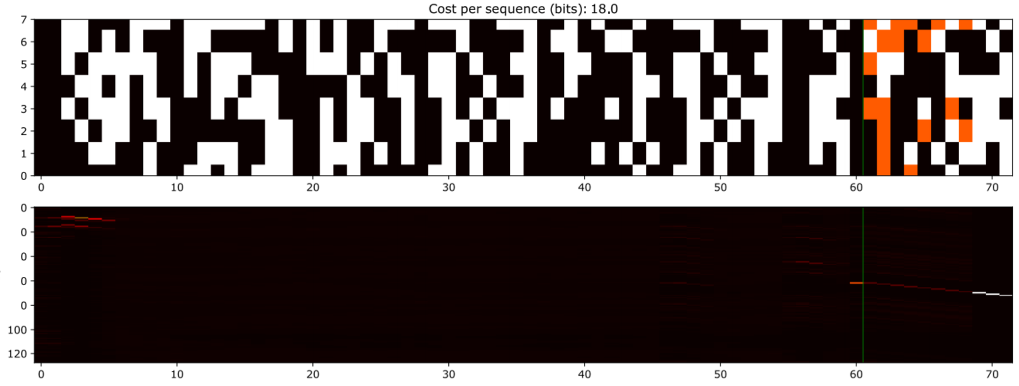
Graves, A., Wayne, G., Reynolds, M. *et al.* Hybrid computing using a neural network with dynamic external memory. *Nature* **538,** 471–476 (2016)

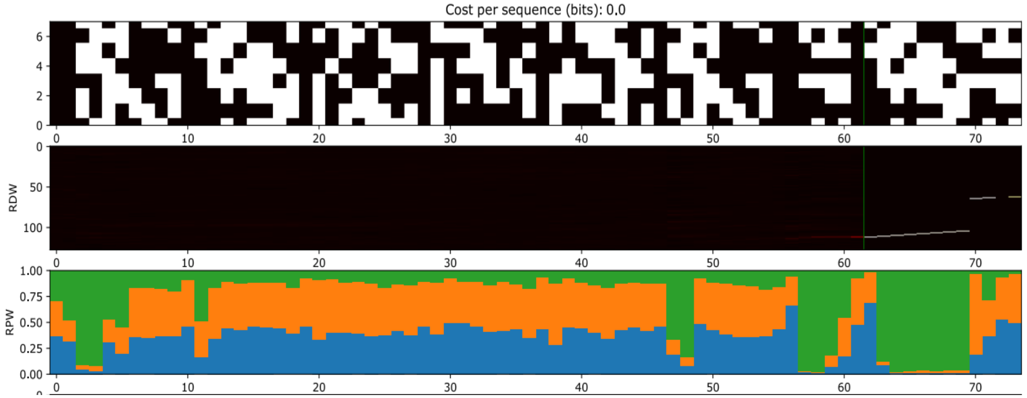
NUTM: implementing multiple algorithms at once

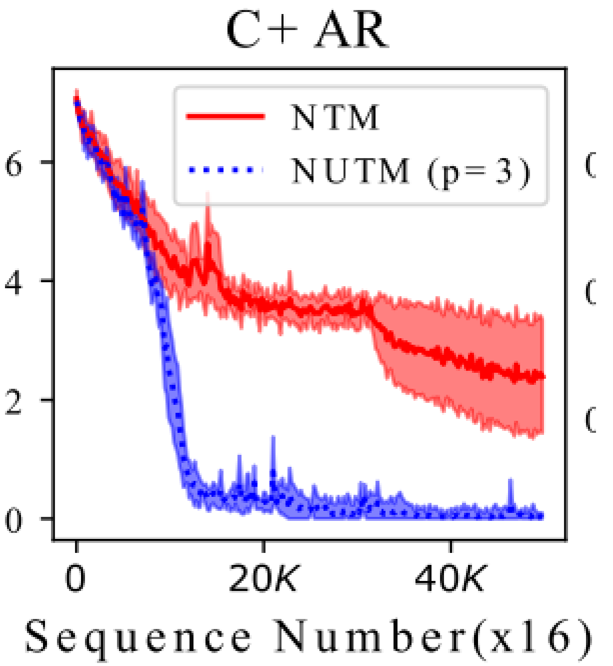












Le, Hung, Truyen Tran, and Svetha Venkatesh. "Neural Stored-program Memory." In *International Conference on Learning Representations*. 2019.

STM: relational memory for graph reasoning

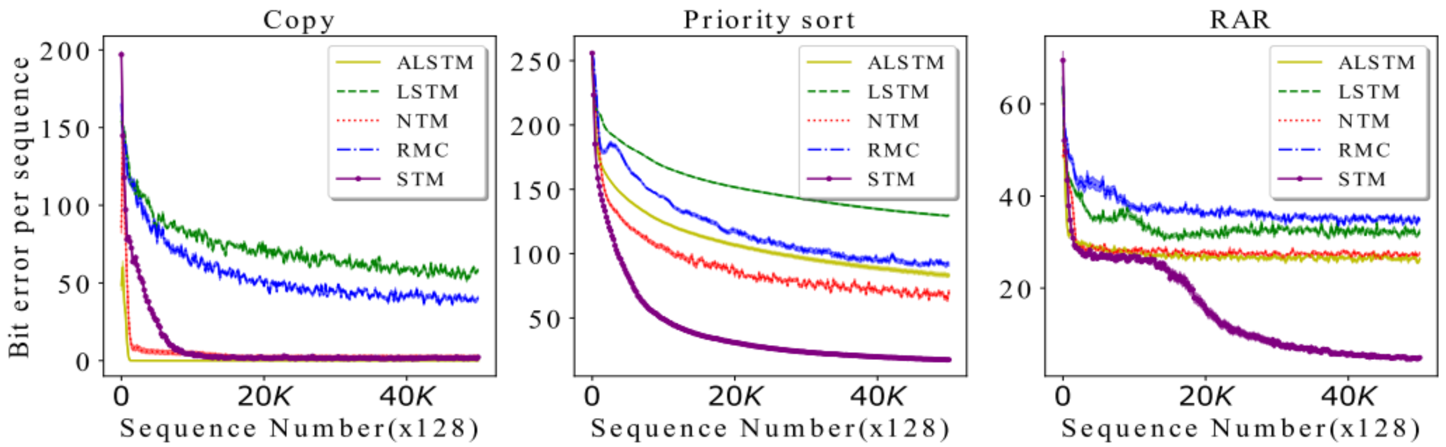


Figure 2. Learning curves on algorithmic synthetic tasks.

Model	#Parameters	Convex hull $N=5$	Convex hull $N=10$	TSP $N=5$	TSP $N=10$	Shortest path	Minimum spanning tree
LSTM	4.5 M	89.15	82.24	73.15 (*2.06*)	62.13 (*3.19*)	72.38	80.11
ALSTM	3.7 M	89.92	85.22	71.79 (*2.05*)	55.51 (*3.21*)	76.70	73.40
DNC	1.9 M	89.42	79.47	73.24 (*2.05*)	61.53 (*3.17*)	83.59	82.24
RMC	2.8 M	93.72	81.23	72.83 (*2.05*)	37.93 (*3.79*)	66.71	74.98
STM	1.9 M	**96.85**	**91.88**	**73.96 (*2.05*)**	**69.43 (*3.03*)**	**93.43**	**94.77**

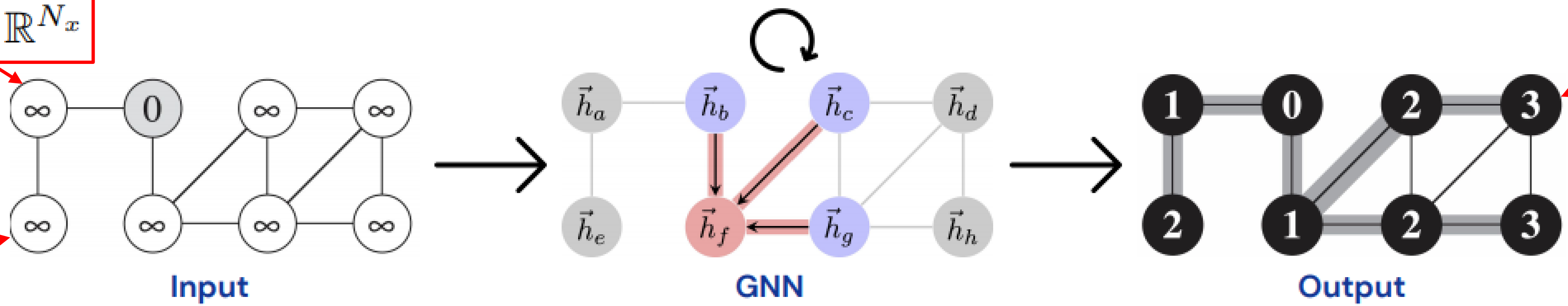
Table 3. Prediction accuracy (%) for geometry and graph reasoning tasks with random *one-hot* associated features. Italic numbers are tour length–additional metric for TSP. Average optimal tour lengths found by brute-force search for $N = 5$ and 10 are 2.05 and 2.88, respectively.

Le, Hung, Truyen Tran, and Svetha Venkatesh. "Self-attentive associative memory." In *International Conference on Machine Learning*, pp. 5682-5691. PMLR, 2020.

Processor as graph neural network (GNN)

$\vec{x}_i^{(t)} \in \mathbb{R}^{N_x}$

$\vec{y}_i^{(t)} \in \mathbb{R}^{N_y}$



$\vec{e}_{ij}^{(t)} \in \mathbb{R}^{N_e}$

https://petar-v.com/talks/Algo-WWW.pdf
Veličković, Petar, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. "Neural Execution of Graph Algorithms." In *International Conference on Learning Representations*. 2019.

Motivation:

- Many algorithm operates on graphs
- Supervise graph neural networks with algorithm operation/step/final output
- Encoder-Process-Decode framework:

$$\vec{z}_i^{(t)} = f_A\left(\vec{x}_i^{(t)}, \vec{h}_i^{(t-1)}\right)$$

$$\mathbf{H}^{(t)} = P\left(\mathbf{Z}^{(t)}, \mathbf{E}^{(t)}\right)$$

$$\vec{y}_i^{(t)} = g_A\left(\vec{z}_i^{(t)}, \vec{h}_i^{(t)}\right)$$

$$\tau^{(t)} = \sigma\left(T_A\left(\mathbf{H}^{(t)}, \overline{\mathbf{H}^{(t)}}\right)\right)$$

Attention

$$\vec{h}_i^{(t)} = \mathrm{ReLU}\left(\sum_{(j,i)\in E} a\left(\vec{z}_i^{(t)}, \vec{z}_j^{(t)}, \vec{e}_{ij}^{(t)}\right) \mathbf{W} \vec{z}_j^{(t)}\right) \qquad \vec{h}_i^{(t)} = U\left(\vec{z}_i^{(t)}, \bigoplus_{(j,i)\in E} M\left(\vec{z}_i^{(t)}, \vec{z}_j^{(t)}, \vec{e}_{ij}^{(t)}\right)\right)$$

Message passing

Example: GNN for a specific problem (DNF counting)

- Count #assignments that satisfy disjuntive normal form (DNF) formula
- Classical algorithm is P-hard O(mn)
- m: #clauses, n: #variables
- Supervised training

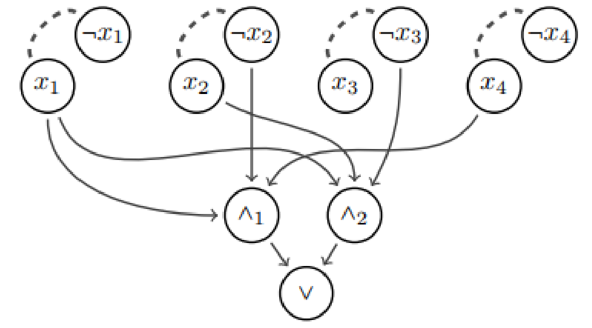


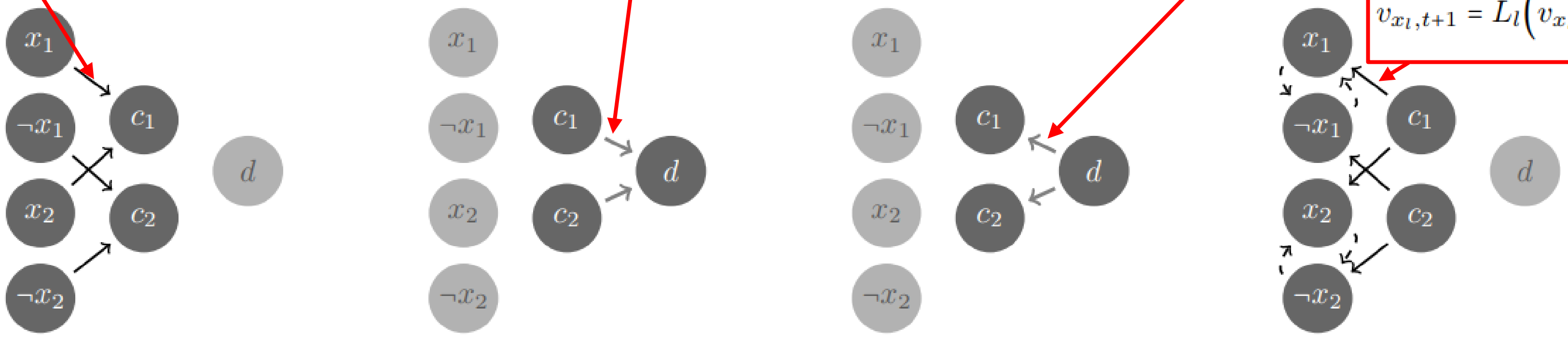
Figure 2: Graph encoding of the DNF formula $\phi = (x_1 \wedge \neg x_2 \wedge x_4) \vee (x_1 \wedge x_2 \wedge \neg x_3)$.

$$\hat{v}_{x_c,t+1} = L_{c_1}\Big(v_{x_c,t}, \sum_{x_l \in N(x_l)} M_l(v_{x_l,t})\Big)$$

$$v_{x_d,t+1} = L_d\Big(v_{x_d,t}, \sum_{x_c \in N(x_d)} M_c(\hat{v}_{x_c,t+1})\Big).$$

$$v_{x_c,t+1} = L_{c_2}\Big(\hat{v}_{x_c,t+1}, M_d(v_{x_d,t+1})\Big).$$

$$v_{x_l,t+1} = L_l\Big(v_{x_l,t}, \big(\sum_{x_c \in N(x_l)} M_c(v_{x_c,t+1}) \| M_l(v_{\neg x_l,t})\big)\Big).$$



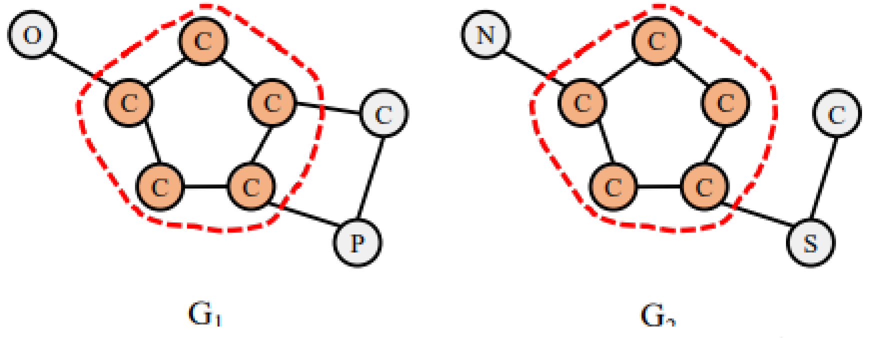
(a) (b) (c) (d)

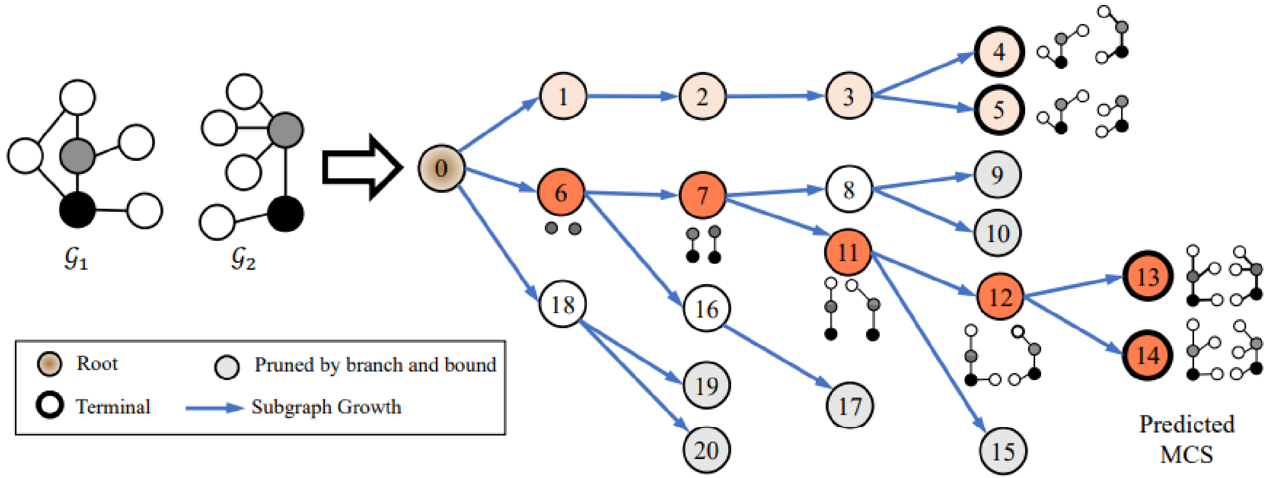
Best: O(m+n)

Abboud, Ralph, Ismail Ceylan, and Thomas Lukasiewicz. "Learning to reason: Leveraging neural networks for approximate DNF counting." In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, pp. 3097-3104. 2020.

Example: GNN trained with reinforcement learning (maximum common subgraph)

- Maximum common subgraph (MCS) is NP-hard
- Search for MCS:
  - BFS then pruning
  - Which node to visit first?
- Cast to RL:
  - State:
    - Current subgraph
    - Node-node mapping
    - Input graph
  - Action: Node pair or edge will be visited
  - Reward: +1 if a node pair is selected
  - Q(s,a)=largest common subgraph size





Bai, Yunsheng, Derek Xu, Alex Wang, Ken Gu, Xueqing Wu, Agustin Marinovic, Christopher Ro, Yizhou Sun, and Wei Wang. "Fast detection of maximum common subgraph via deep q-learning." *arXiv preprint arXiv:2002.03129* (2020).

Learning state representation with GNN

Bidomain representation

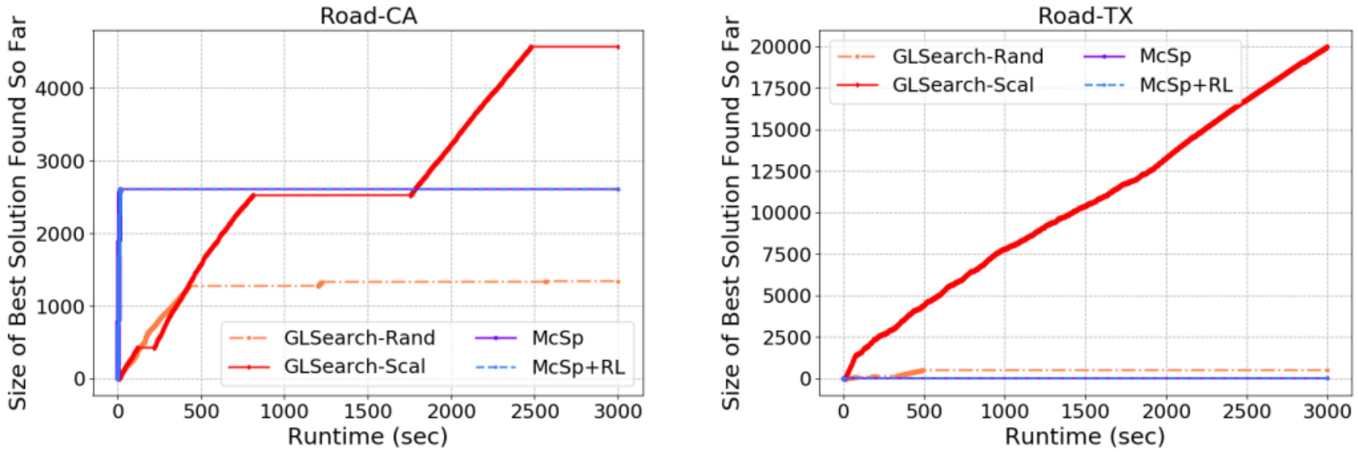
$$Q^*(s_t, a_t), \text{ as } r_t + \gamma V^*(s_{t+1})$$

$$Q(s_t,a_t) = 1+\gamma \text{MLP}\Big(\text{CONCAT}\big(\text{INTERACT}(\boldsymbol{h}_{\mathcal{G}_1}, \boldsymbol{h}_{\mathcal{G}_2}), \text{INTERACT}(\boldsymbol{h}_{s1}, \boldsymbol{h}_{s2}), \boldsymbol{h}_{\mathcal{D}c}, \boldsymbol{h}_{D_0}\big)\Big).$$

$$\boldsymbol{h}_{D_k} = \text{INTERACT}\big(\text{READOUT}(\{\boldsymbol{h}_i | i \in \mathcal{V}'_{k1}\}), \text{READOUT}(\{\boldsymbol{h}_j | j \in \mathcal{V}'_{k2}\})\big).$$

$$\text{READOUT}(\{\boldsymbol{h}_{D_k} | k \in \mathcal{D}^{(c)}\}$$

Pretrain with ground-truth Q or expert estimation
Then train as DQN



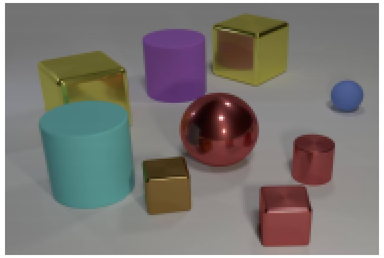
(a) Result on ROAD-CA with 978513 nodes.

(b) Result on ROAD-TX with 1080909 nodes.

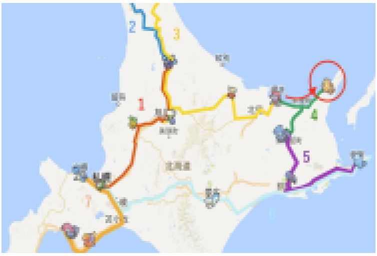
Method	ROAD	DBEN
GLSEARCH (no $\boldsymbol{h}_{\mathcal{G}}$)	0.977	0.878
GLSEARCH (no $\boldsymbol{h}_s$)	**1.000**	0.874
GLSEARCH (no $\boldsymbol{h}_{\mathcal{D}c}$)	0.803	0.780
GLSEARCH (no $\boldsymbol{h}_{\mathcal{D}0}$)	0.576	0.856
GLSEARCH (SUM interact)	0.902	0.913
GLSEARCH (unfactored)	0.447	0.807
GLSEARCH (unfactored-i)	0.500	0.789
GLSEARCH	0.992	**1.000**
BEST SOLUTION SIZE	132	508

Neural networks and algorithms alignment

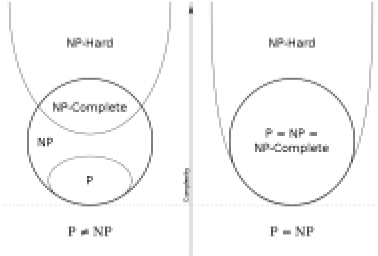
Summary statistics
What is the maximum value difference among treasures?



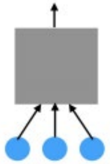
Relational argmax
What are the colors of the furthest pair of objects?



Dynamic programming
What is the cost to defeat monster X by following the optimal path?

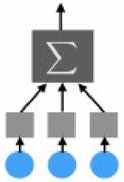


NP-hard problem
Subset sum: Is there a subset that sums to 0?



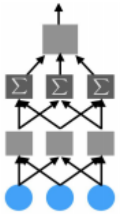
MLPs
~ feature extraction

$$y = \mathrm{MLP}(\|_{s\in S} X_s)$$

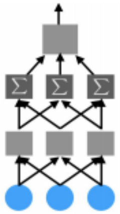


Deep Sets (Zaheer *et al.*,
~ summary statistics

$$y = \mathrm{MLP}_2\left(\sum_{s\in S} \mathrm{MLP}_1(X_s)\right)$$



GNNs
~ (pairwise) relations



GNNs
~ (pairwise) relations

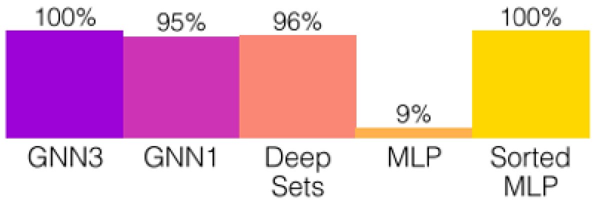
$$h_s^{(k)} = \sum_{t\in S} \mathrm{MLP}_1^{(k)}\left(h_s^{(k-1)}, h_t^{(k-1)}\right)$$
$$y = \mathrm{MLP}_2\left(\sum_{s\in S} h_s^{(K)}\right)$$

Neural exhaustive search

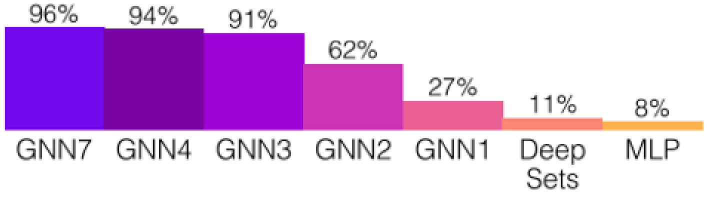
https://petar-v.com/talks/Algo-WWW.pdf

Xu, Keylu, Jingling Li, Mozhi Zhang, Simon S. Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. "What Can Neural Networks Reason About?." *ICLR 2020* (2020).

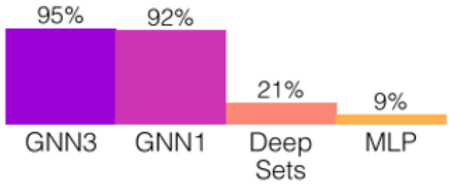
GNN is aligned with Dynamic Programming (DP)



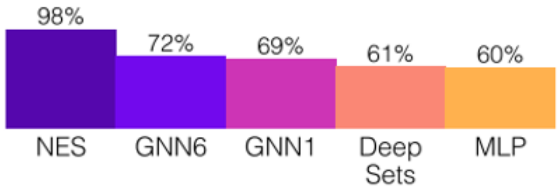
(a) Maximum value difference.



(c) Monster trainer.



(b) Furthest pair.



(d) Subset sum. Random guessing yields 50%.

Graph Neural Network	**Bellman-Ford algorithm**
for k = 1 … GNN iter:	for k = 1 … |S| - 1:
for u in S:	for u in S:
$h_u^{(k)} = \Sigma_v \text{ MLP}(h_v^{(k-1)}, h_u^{(k-1)})$	$d[k][u] = \min_v d[k-1][v] + \text{cost}(v, u)$

No need to learn for-loops

Learns a simple reasoning step

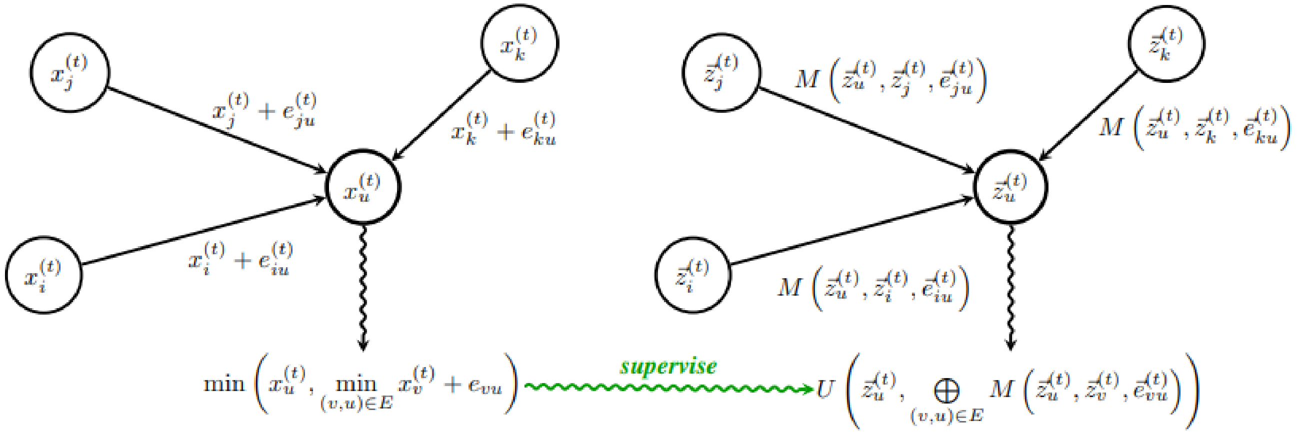
$h_u^{(k)}$	$d[k][u]$
Σ_v	$\min_v$
$\text{MLP}(h_v^{(k-1)}, h_u^{(k-1)})$	$d[k-1][v] + \text{cost}(v, u)$

Neural exhaustive search

$$\text{MLP}_2(\max_{\tau \subseteq S} \text{MLP}_1 \circ \text{LSTM}(X_1, ..., X_{|\tau|} : X_1, ..., X_{|\tau|} \in \tau)).$$

If alignment exists → step-by-step supervision

- Merely simulate the classical graph algorithm, generalizable
- No algorithm discovery



Algorithm	**Inputs**	**Supervision signals**
Breadth-first search	$x_i^{(t)}$: is i reachable from s in $\leq t$ hops?	$x_i^{(t+1)}$, $\tau^{(t)}$: has the algorithm terminated?
Bellman-Ford	$x_i^{(t)}$: shortest distance from s to i (using $\leq t$ hops)	$x_i^{(t+1)}$, $\tau^{(t)}$, $p_i^{(t)}$: predecessor of i in the shortest path tree (in $\leq t$ hops)
Prim's algorithm	$x_i^{(t)}$: is node i in the (partial) MST (built from s after t steps)?	$x_i^{(t+1)}$, $\tau^{(t)}$, $p_i^{(t)}$: predecessor of i in the partial MST

Joint training is encouraged

Veličković, Petar, Rex Ying, Matilde Padovano, Raia Hadsell, and Charles Blundell. "Neural Execution of Graph Algorithms." In *International Conference on Learning Representations*. 2019.

Table 1: Accuracy of predicting reachability at different test-set sizes, trained on graphs of 20 nodes. GAT* correspond to the best GAT setup as per Section 3 (GAT-full using the full graph).

Model	**Reachability (mean step accuracy / last-step accuracy)**		
	20 nodes	*50 nodes*	*100 nodes*
LSTM (Hochreiter & Schmidhuber, 1997)	81.97% / 82.29%	88.35% / 91.49%	68.19% / 63.37%
GAT* (Veličković et al., 2018)	93.28% / 99.86%	93.97% / **100.0%**	92.34% / **99.97%**
GAT-full* (Vaswani et al., 2017)	78.40% / 77.86%	85.76% / 91.83%	88.98% / 91.51%
MPNN-mean (Gilmer et al., 2017)	**100.0% / 100.0%**	61.05% / 57.89%	27.17% / 21.40%
MPNN-sum (Gilmer et al., 2017)	99.66% / **100.0%**	94.25% / **100.0%**	94.72% / 98.63%
MPNN-max (Gilmer et al., 2017)	**100.0% / 100.0%**	**100.0% / 100.0%**	**99.92%** / 99.80%

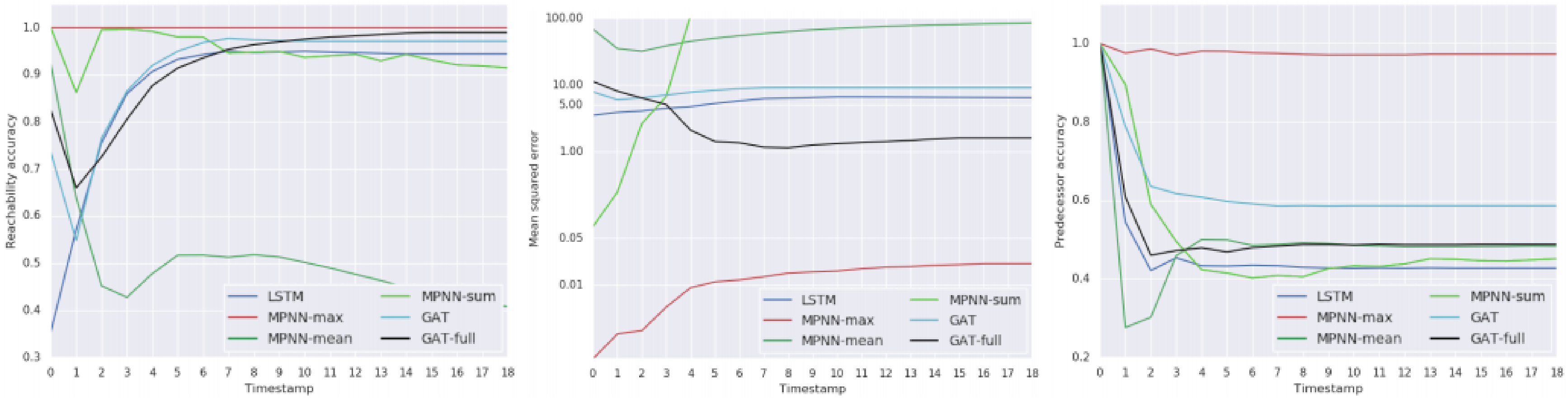
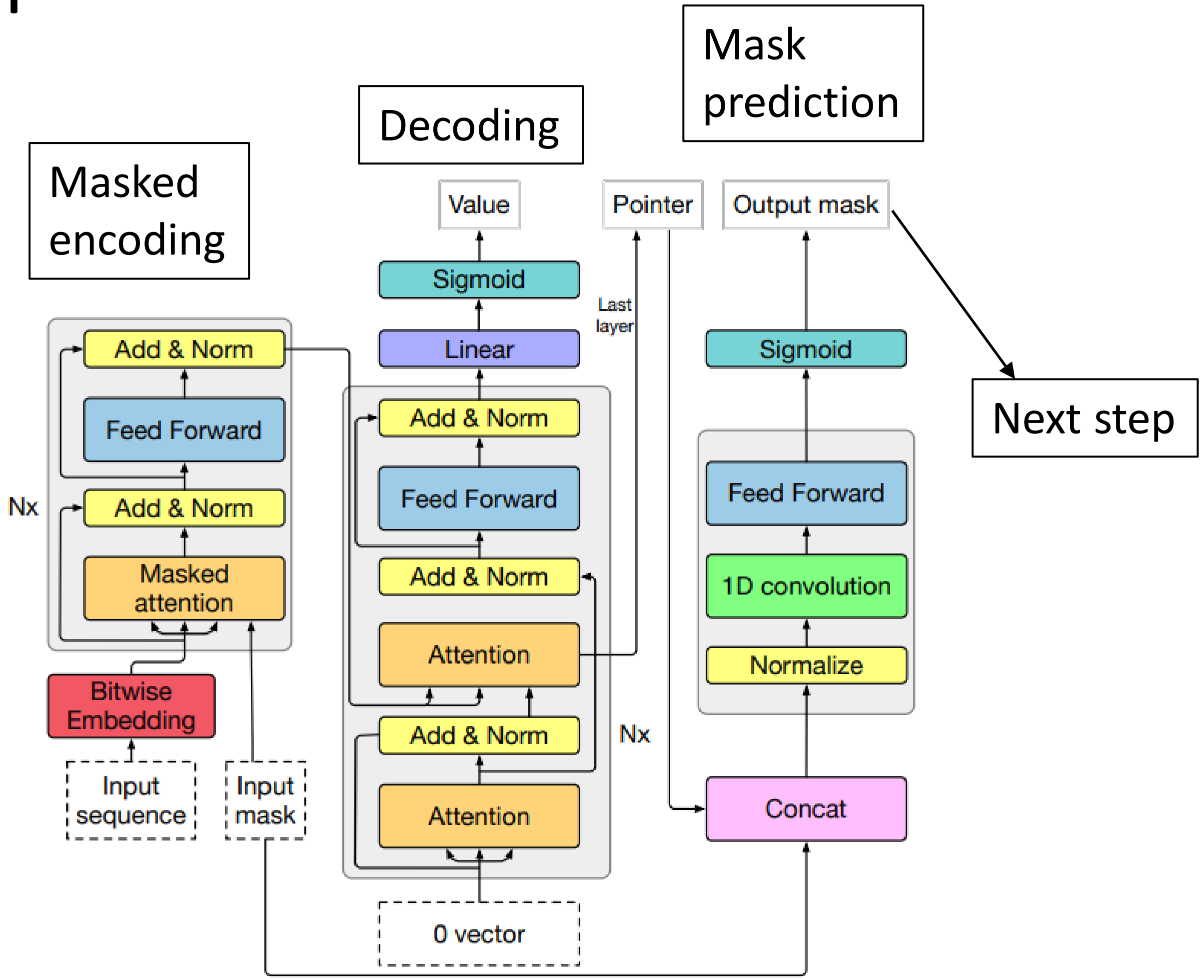


Figure 3: The per-step algorithm execution performances in terms of reachability accuracy (**left**), distance mean-squared error (**middle**) and predecessor accuracy (**right**), tested on 100-node graphs after training on 20-node graphs. Please mind the scale of the MSE plot.

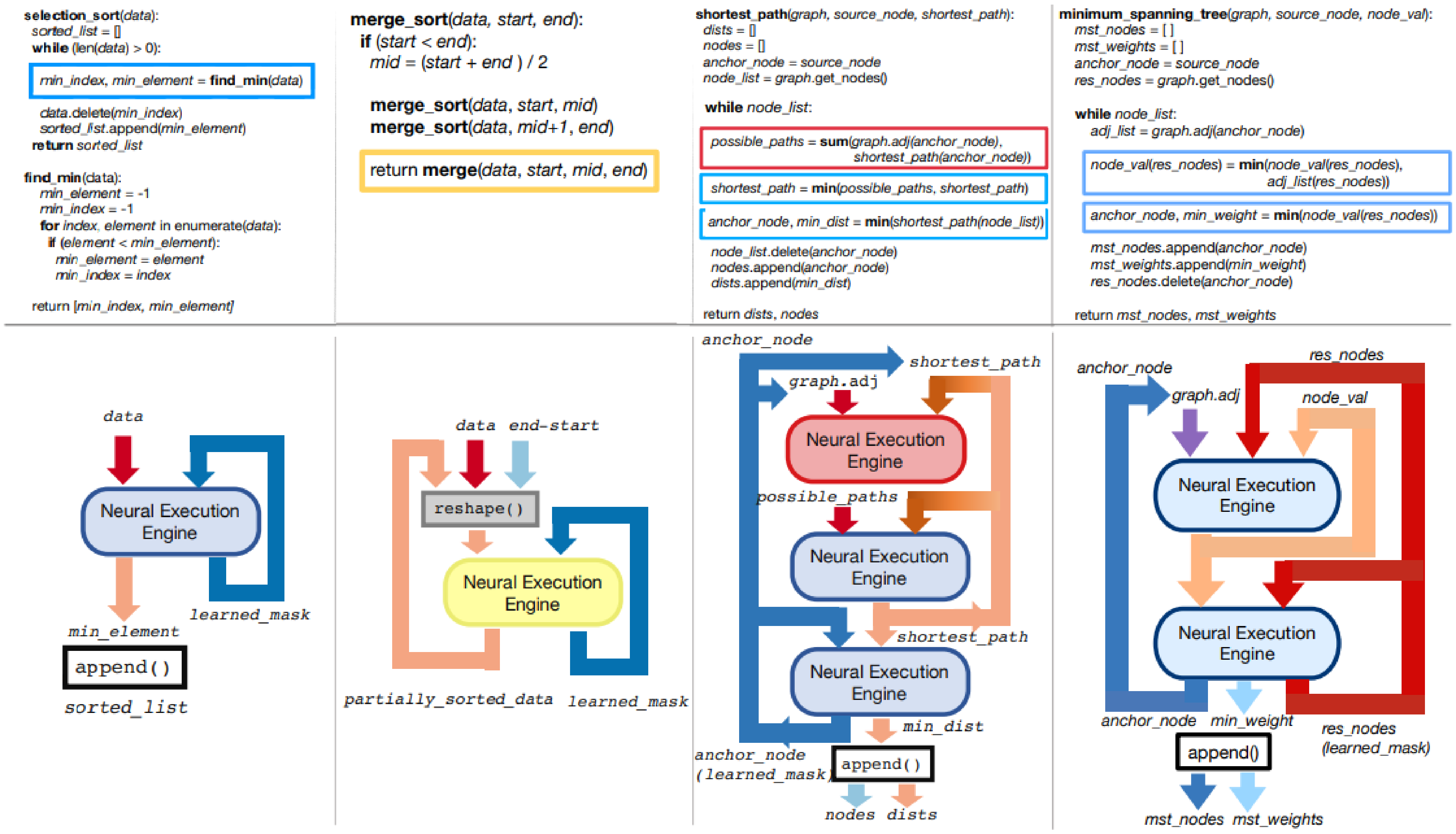
Processor as Transformer

- Back to input sequence (set), but stronger generalization
- Transformer with encoder mask ~ graph attention
- Use Transformer with:
  - Binary representation of numbers
  - Dynamic conditional masking



Yan, Yujun, Kevin Swersky, Danai Koutra, Parthasarathy Ranganathan, and Milad Hashemi.
"Neural Execution Engines: Learning to Execute Subroutines." *Advances in Neural Information Processing Systems* 33 (2020).

Training with execution trace



The results show strong generalization

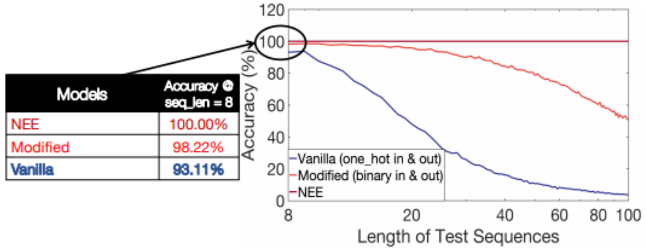


Figure 3: Sorting performance of transformers trained on sequences of up to length 8.

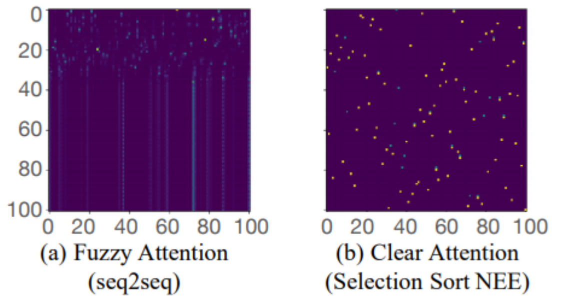


Figure 4: Visualizing decoder attention weights. Attention is over each row. Transformer attention saturates as the output sequence length increases, while NEE maintains sharp attention.

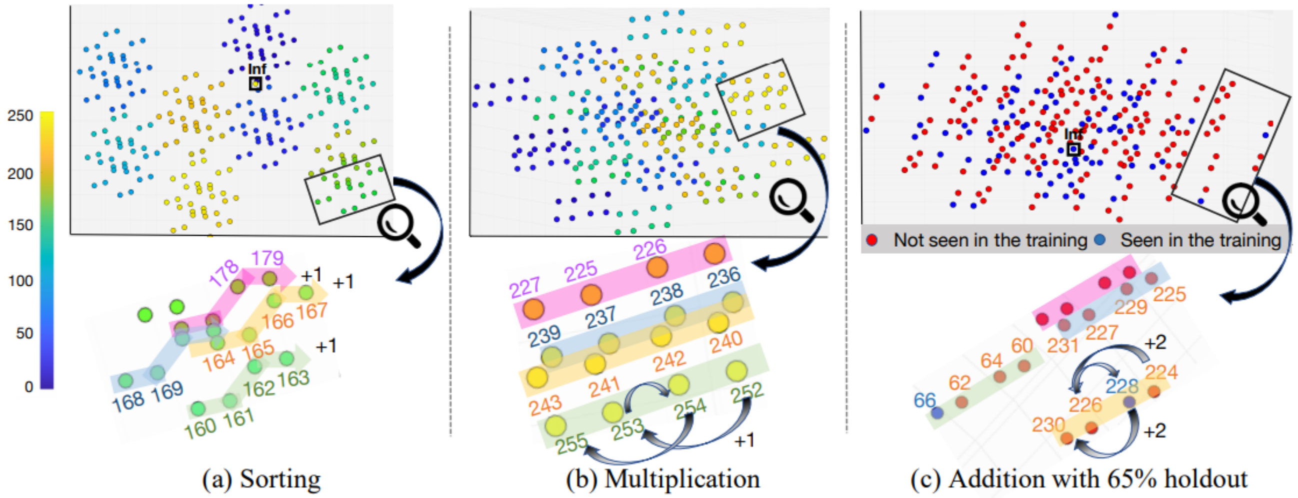


Figure 5: 3D PCA visualization of learned bitwise embeddings for different numeric tasks. The embeddings exhibit regular, task-dependent structure, even when most numbers have not been seen in training (c).

Accuracy \ Sizes	25	50	75	100
Selection sort	100.00	100.00	100.00	100.00
Merge sort	100.00	100.00	100.00	100.00
Shortest path	100.00	100.00	100.00	100.00*
Minimum spanning tree	100.00	100.00	100.00	100.00

QA