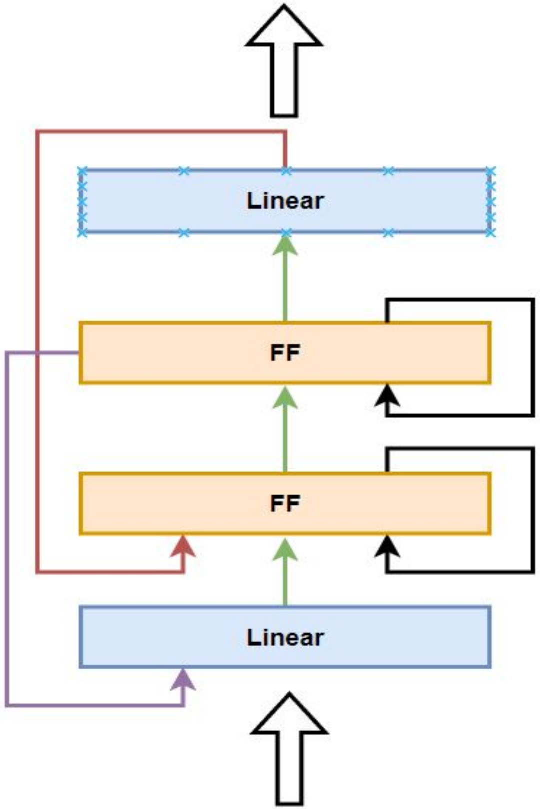
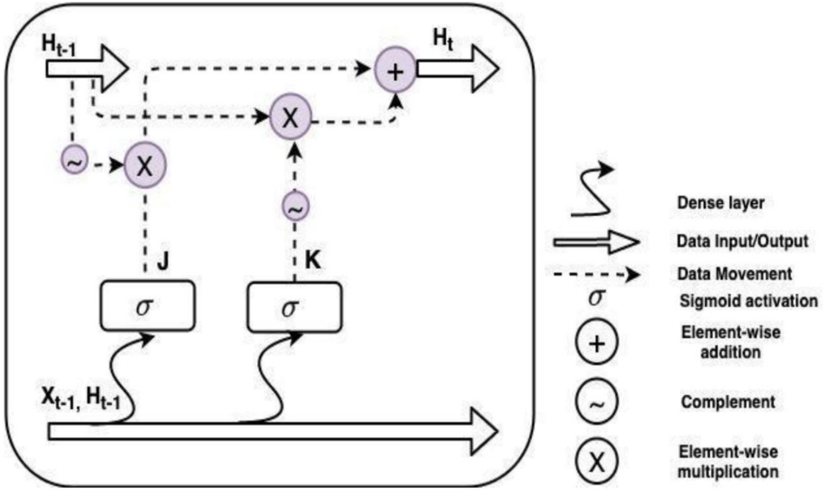
Hippocampus Modeling



Model Architecture



$$\mathbf{J} = \sigma\left(\mathbf{W}_{JX}\mathbf{X}_t + \mathbf{W}_{JH_{t-1}}\mathbf{H}_{t-1}\right)$$

$$\mathbf{K} = \sigma\left(\mathbf{W}_{KX}\mathbf{X}_t + \mathbf{W}_{KH_{t-1}}\mathbf{H}_{t-1}\right)$$

$$\mathbf{H}_t = \mathbf{J}(1-\mathbf{H}_{t-1}) + (1-\mathbf{K})\mathbf{H}_{t-1}$$

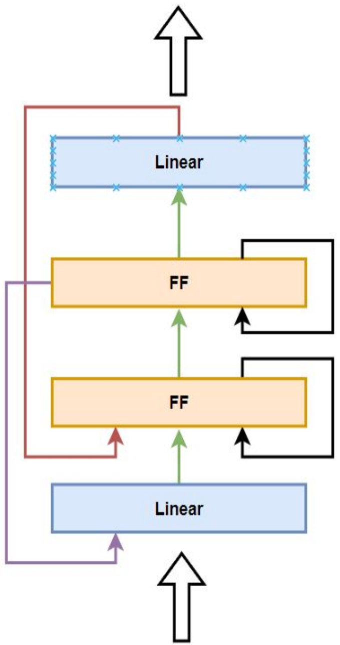
X = Input, H = Hidden state of flip-flop

Flip-flop Neuron

Modes of Operation

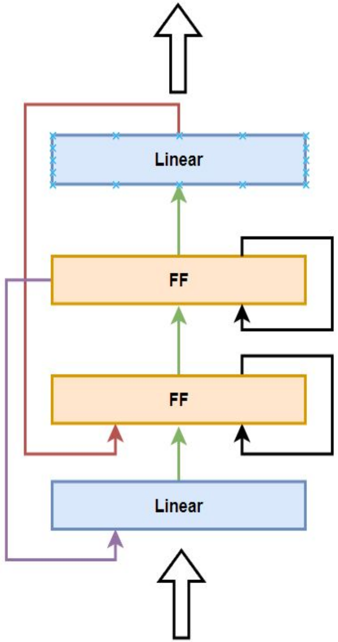
- Listen
- Maintain
- Retrieve

Illustration



Input Item	Instruction	Memory Stack
No Input	-	[]
[1 0 0 1 0 1 1]	Store	[[1 0 0 1 0 1 1]]
[0 1 0 1 1 1 0]	Store	[[1 0 0 1 0 1 1] [0 1 0 1 1 1 0]]
[1 0 1 0 0 1 0]	Store	[[1 0 0 1 0 1 1] [0 1 0 1 1 1 0] [1 0 1 0 0 1 0]]
[1 0 1 1 1 1 0]	Store	[[1 0 0 1 0 1 1] [0 1 0 1 1 1 0] [1 0 1 0 0 1 0] [1 0 1 1 1 1 0]]

Illustration



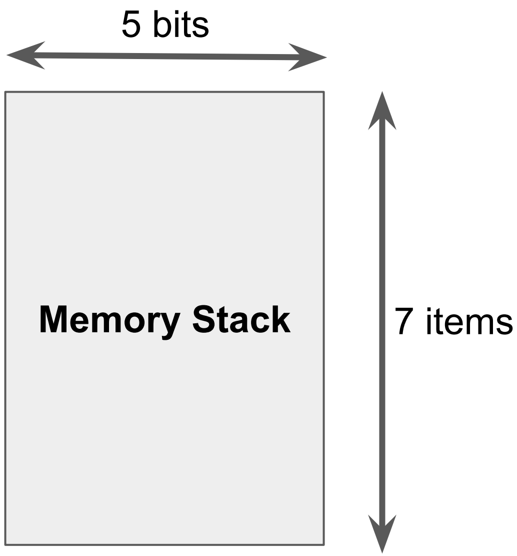
Input Item	Instruction	Memory Stack
[0 0 1 1 0 1 1]	Maintain	[[1 0 0 1 0 1 1] [0 1 0 1 1 1 0] [1 0 1 0 0 1 0] [1 0 1 1 1 1 0]]
[1 1 0 1 0 1 0]	Maintain	[[1 0 0 1 0 1 1] [0 1 0 1 1 1 0] [1 0 1 0 0 1 0] [1 0 1 1 1 1 0]]
[1 1 1 0 1 1 0]	Maintain	[[1 0 0 1 0 1 1] [0 1 0 1 1 1 0] [1 0 1 0 0 1 0] [1 0 1 1 1 1 0]]
[1 0 1 0 1 1 1]	Maintain	[[1 0 0 1 0 1 1] [0 1 0 1 1 1 0] [1 0 1 0 0 1 0] [1 0 1 1 1 1 0]]

Illustration

Input Item	Instruction	Output	Memory Stack
[0 0 1 1 0 1 1]	Retrieve	[1 0 0 1 0 1 1]	[[0 1 0 1 1 1 0] [1 0 1 0 0 1 0] [1 0 1 1 1 1 0]]
[1 1 0 1 0 1 0]	Retrieve	[0 1 0 1 1 1 0]	[[1 0 1 0 0 1 0] [1 0 1 1 1 1 0]]
[1 1 1 0 1 1 0]	Retrieve	[1 0 1 0 0 1 0]	[1 0 1 1 1 1 0]]
[1 0 1 0 1 1 1]	Retrieve	[1 0 1 1 1 1 0]	[]

Modelling Basic Microprocessor Behaviour

Operation bit (5 bits)	Address bit 1 (3 bits)	Address bit 2 (3 bits)	Data bit (5 bits)



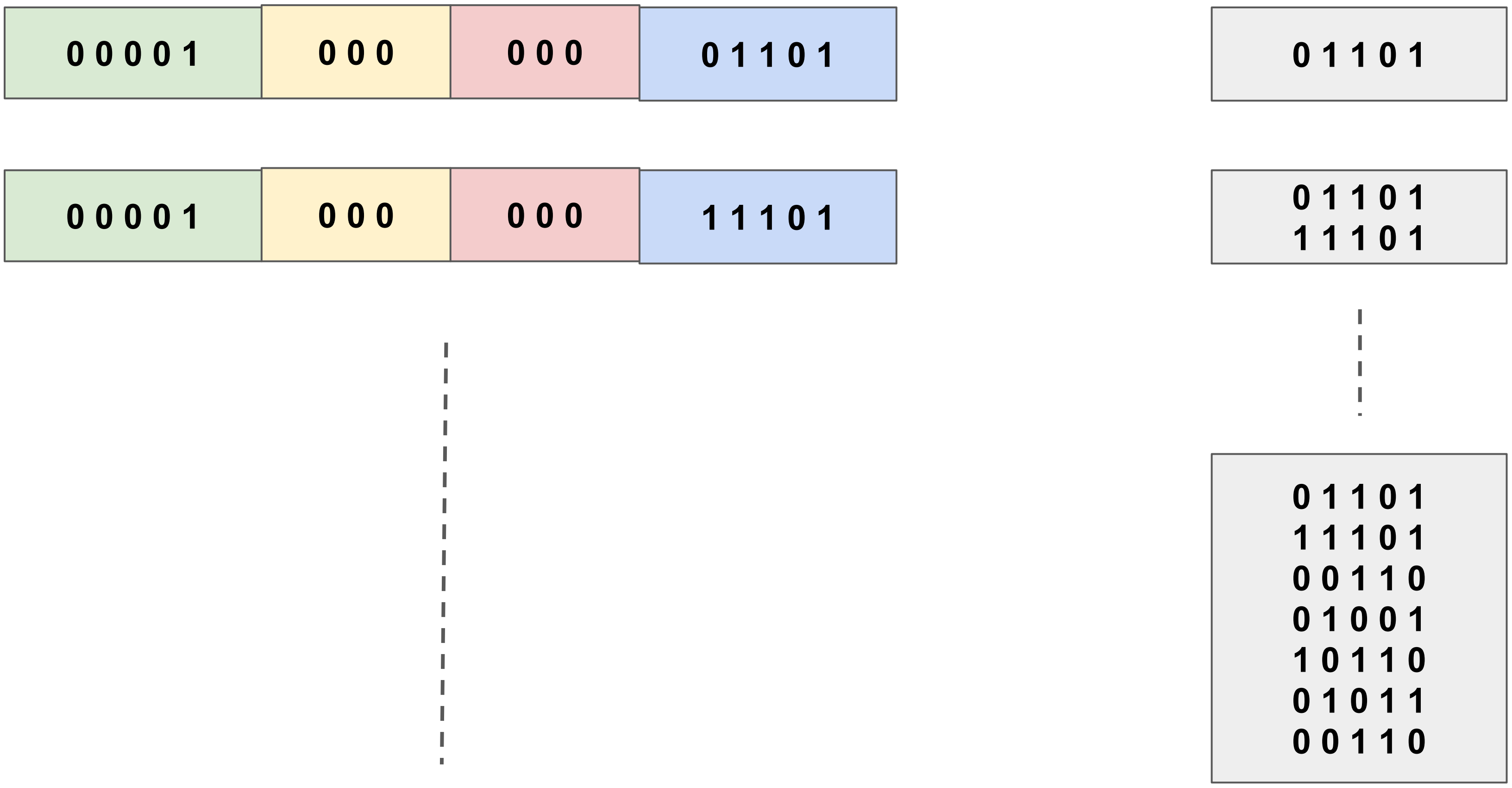
Model training

- Keep pushing items till memory stack is full (7 items)
- Followed by *k* instructions
- Retrieve all modified items from memory

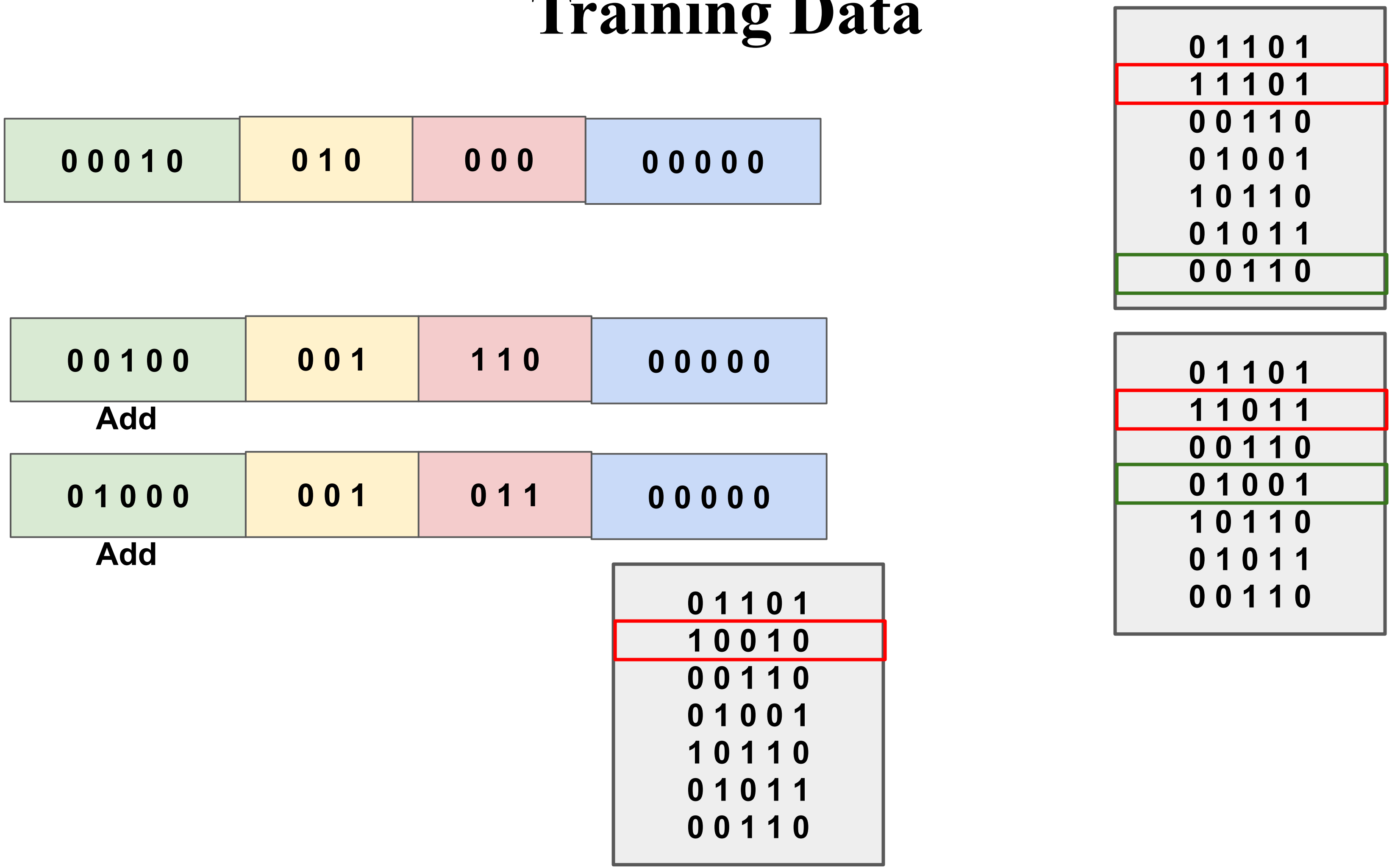
Types of Operation

1. Push
2. Add (Bitwise XOR)
3. Bitwise AND
4. Snooze (Do nothing)
5. Pop (FIFO)

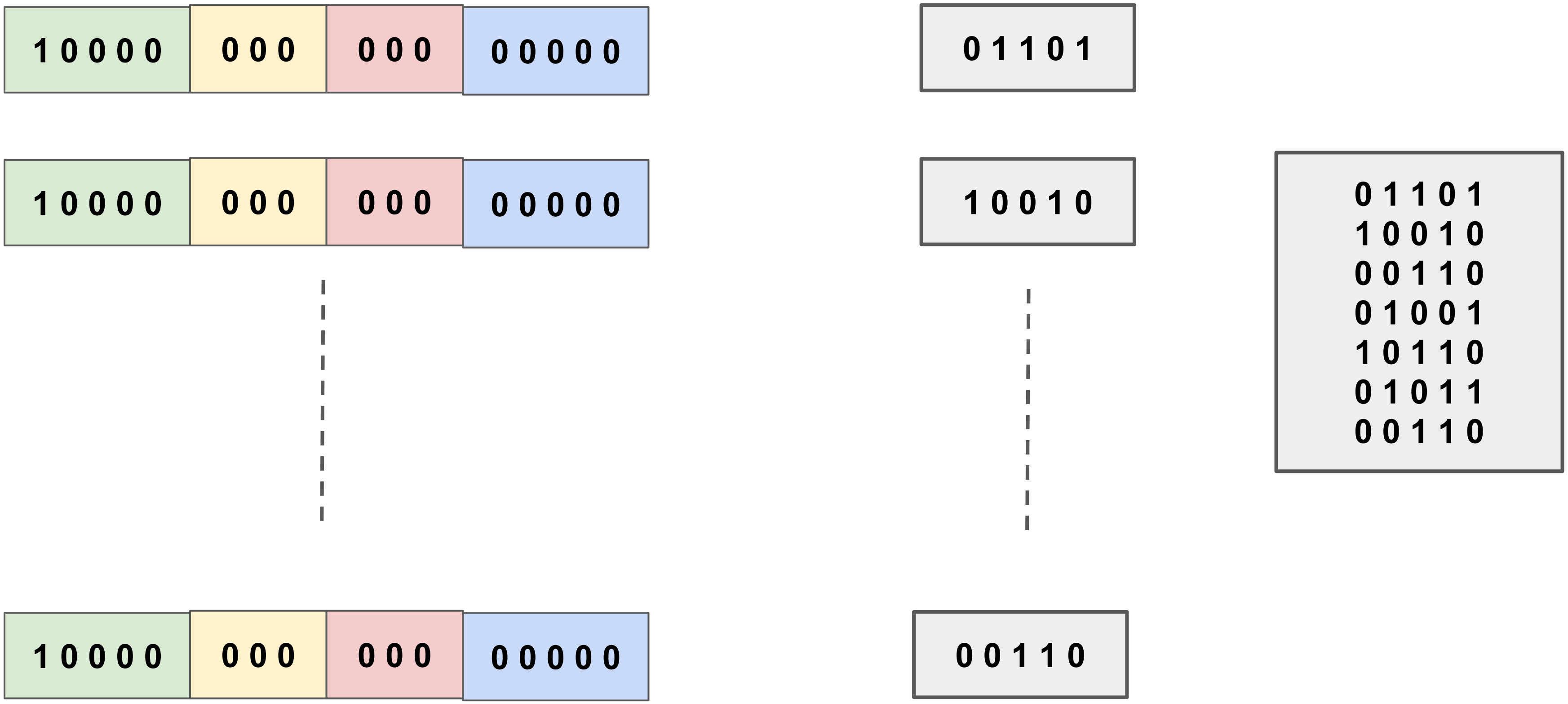
Training Data



Training Data

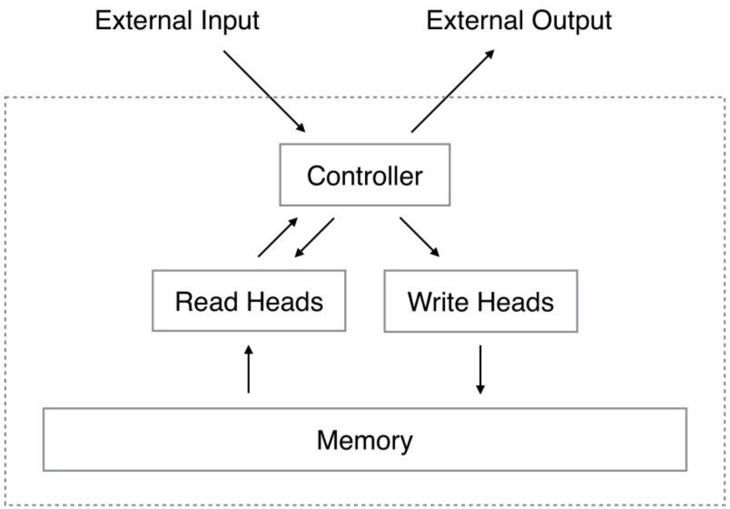


Training Data

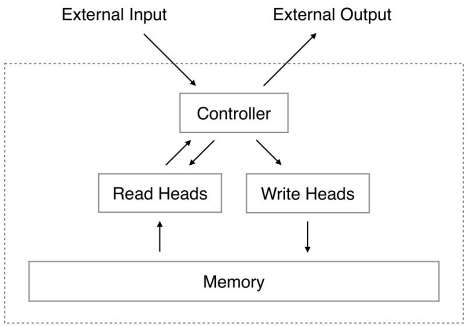


Neural Turing Machines

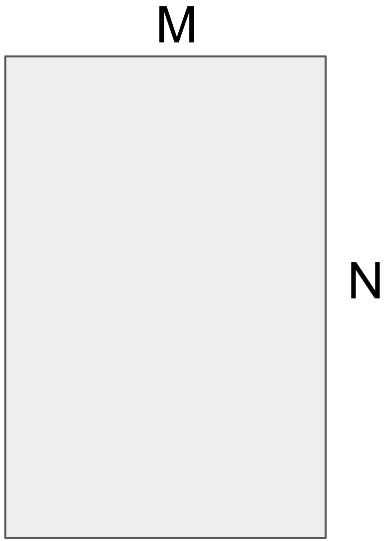
- RNNs are Turing-Complete (Siegelmann and Sontag, 1995) and therefore have the capacity to simulate arbitrary procedures, *if properly wired*
- Yet what is possible in principle is not always what is simple in practice
- We therefore enrich the capabilities of standard recurrent networks with external pool of memory to simplify the solution of algorithmic tasks



NTM - Reading & Writing



- Every component is differentiabel to train with gradient descent
- By an attentional “focus” mechanism each read and write operation interact with a small portion of the memory, while ignoring the rest.



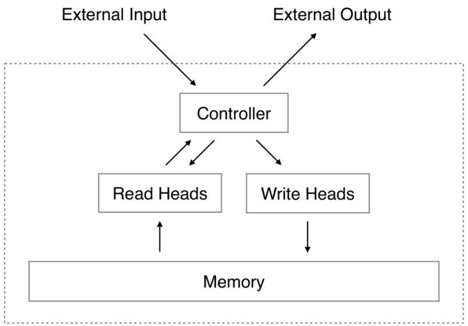
The length M *read vector* $\mathbf{r}_t$ returned by the head is defined as a convex combination of the row-vectors $\mathbf{M}_t(i)$ in memory:

$$\mathbf{r}_t \longleftarrow \sum_i w_t(i)\mathbf{M}_t(i), \tag{2}$$

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

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

NTM - Addressing Mechanisr



- Content-based - focuses attention on location based on similarity of the content (like Hopfield Networks)

- Location-based - focuses attention based on address/location. Used to solve arithmetic problems (like f = x + y)

- Content-based addressing is general than location-based addressing as the content of a memory location could include location information inside it

$$w_t^c(i) \longleftarrow \frac{\exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(i)\big]\Big)}{\sum_j \exp\Big(\beta_t K\big[\mathbf{k}_t, \mathbf{M}_t(j)\big]\Big)}.$$

$$\mathbf{w}_t^g \longleftarrow g_t \mathbf{w}_t^c + (1 - g_t)\mathbf{w}_{t-1}.$$

If we index the N memory locations from 0 to $N-1$, the rotation applied to w_t^g by s_t can be expressed as the following circular convolution:

$$\tilde{w}_t(i) \longleftarrow \sum_{j=0}^{N-1} w_t^g(j)\, s_t(i-j)$$

NTM - Copy & Repeat Task

initialise: move head to start location
while input delimiter not seen **do**
  receive input vector
  write input to head location
  increment head location by 1
end while
return head to start location
while true **do**
  read output vector from head location
  emit output
  increment head location by 1
end while

Repeat Copy

Input: k binary items and i times to repeat

To test the model can learn nested functions (for loop)

NTM continues to copy as the length increases , while LSTM rapidly degrades beyond length 20.