Memory networks

Zhirong Wu
Feb 9th, 2015

Outline

motivation

Most machine learning algorithms try to learn a static mapping, and it has been elusive to incorporate memory in the learning.

“ Despite its wide-ranging success in modelling complicated data, modern machine learning has largely neglected the use of logical flow control and external memory. “

“ Most machine learning models lack an easy way to read and write to part of a (potentially very large) long-term memory component, and to combine this seamlessly with inference.”

— quoted from today’s papers

Outline

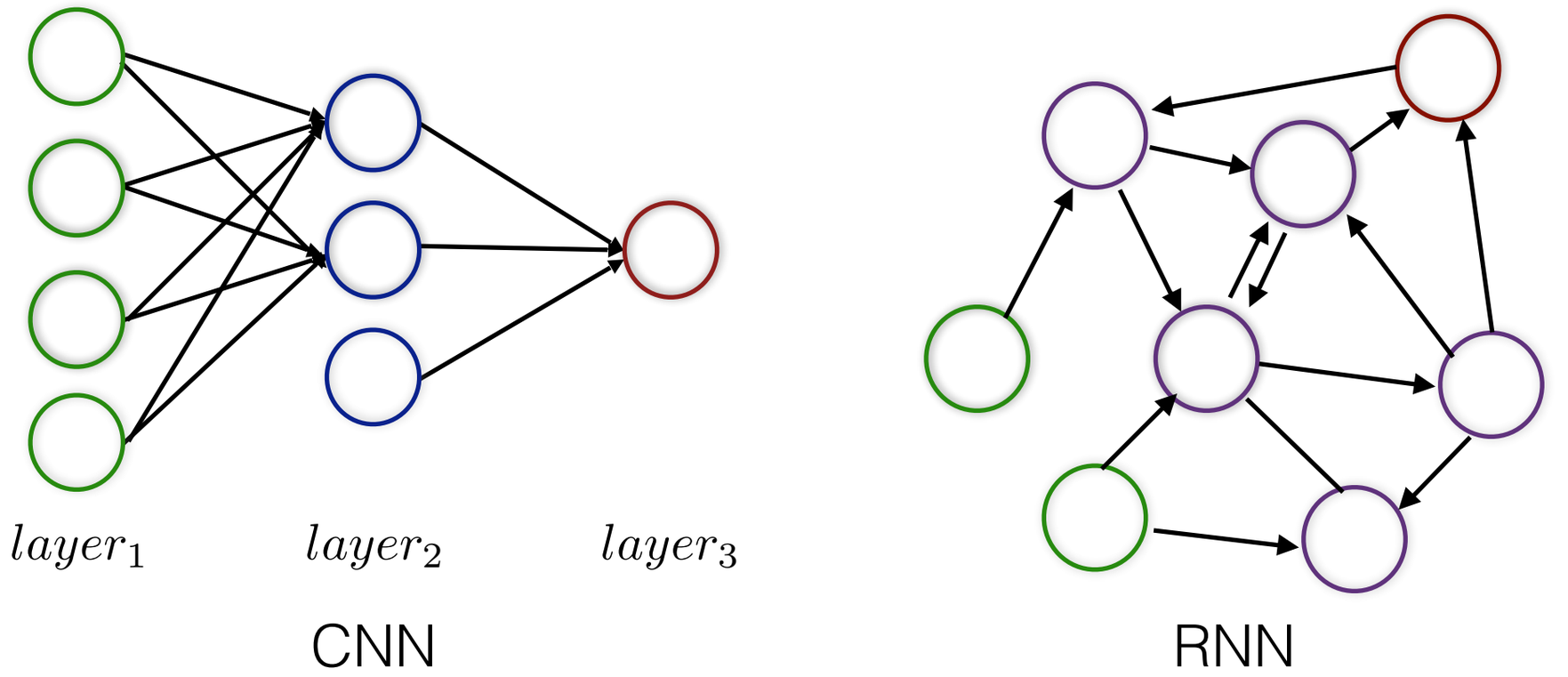
3 papers:

- Learning to execute:
  a direct application of RNN.
- QA memory network:
  explicitly models hardware memory.
- neural turing machine:
  also formulate addressing mechanism.

end to end machine learning

Learning to execute

Recap RNN:



- similar to CNN, RNN has input, hidden, and output units.
- unlike CNN, the output is not only a function of the new input, but also relies on the hidden state of previous time.
- LSTM is a special case of RNN, where it is made to store long term memory easily.

Learning to execute

Can LSTM learn to execute python code?

Input:

```
j=8584
for x in range(8):
  j+=920
b=(1500+j)
print((b+7567))
```

Target: 25011.

Input:

```
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else 5308)
```

Target: 12184.

LSTM reads the entire input one character at a time and produces the output one character at a time.

Learning to execute

experiment settings

operators:

addition, subtraction, multiplication, variable assignments, if statements, and for loops, but not double loops.

length parameter:

constrain the integer in a maximum length.

nesting parameter:

constrain the number of times to combine operations.

Input:

```
i=8827
c=(i-5347)
print((c+8704) if 2641<8500 else 5308)
```

Target: 12184.

an example of length = 4, nesting = 3

Learning to execute

curriculum learning

A trick for learning that gradually increase the difficulties of training examples.

baseline:	training examples with length = a, nesting = b.
naive:	start with length = 1, nesting = 1 and gradually increase until length = a, nesting = b.
mix:	to generate a example, first pick a random length from [1, a], and a random nesting from [1, b].
combined:	a combination of naive and mix.

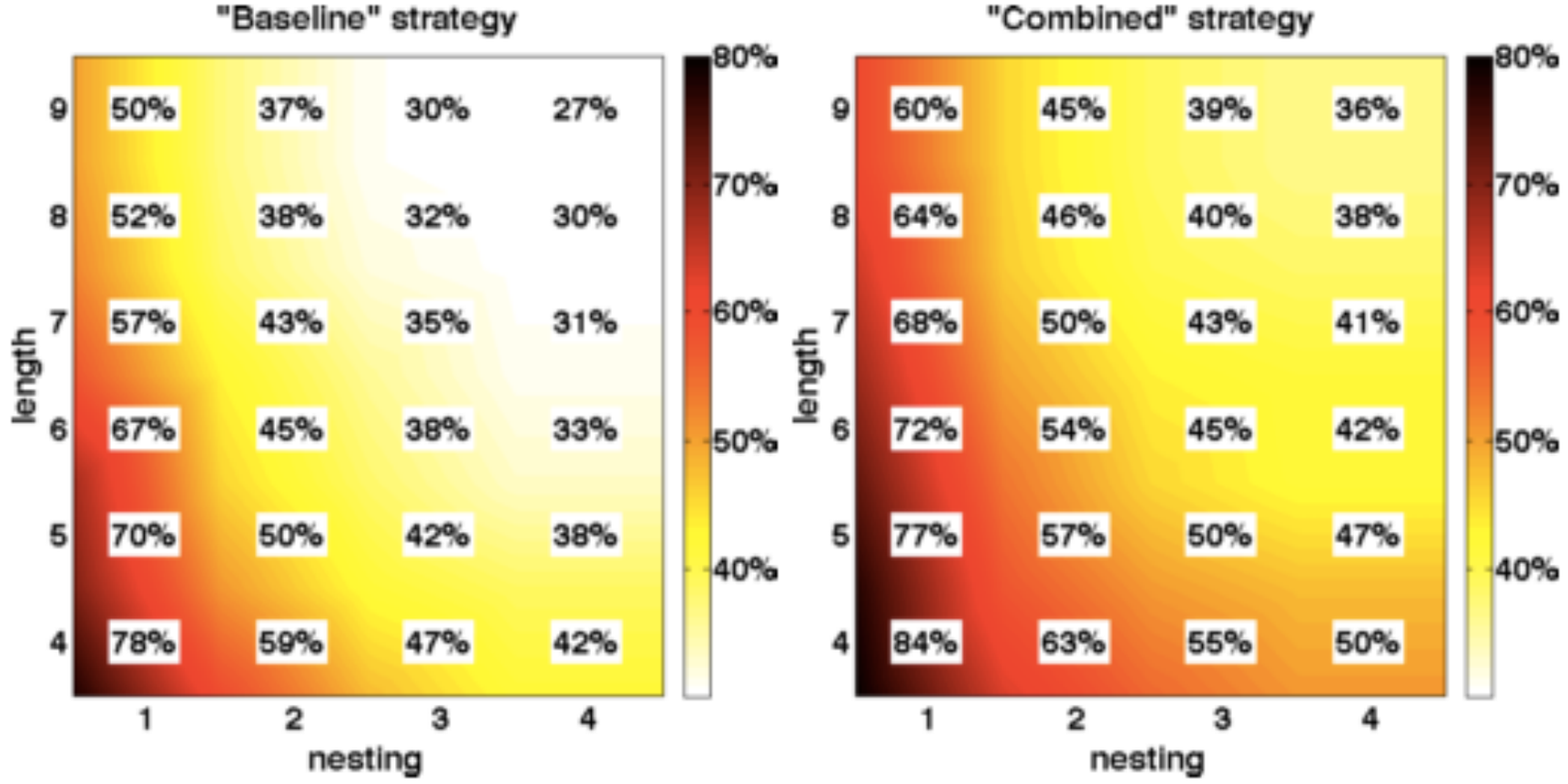
Learning to execute

evaluation

use teacher forcing

when predicting the i-th digit of the target, the LSTM is provided with the correct first i-1 digits.

results



Learning to execute

torch code available:

https://github.com/wojciechz/learning_to_execute

QA memory networks

The hidden state of RNN is very hard to understand. Plus the long term memory training is still very difficult.

Instead of using a recurrent matrix to retain information through time, why not build a memory directly?

The model is then trained to learn how to operate effectively with the memory component. A new kind of learning.

QA memory networks

a general framework, 4 components:

I: (input feature map) – converts the incoming input to the internal feature representation.

G: (generalization) – updates old memories given the new input.

O: (output feature map) – produces a new output, given the new input and the current memory state.

R: (response) – converts the output into the response format desired. For example, a textual response or an action.

QA memory networks

a simple implementation for text

I: (input feature map) – converts the incoming input to the internal feature representation.

I(x) = x: raw text

QA memory networks

a simple implementation for text

I: (input feature map) – converts the incoming input to the internal feature representation.

I(x) = x: raw text

G: (generalization) – updates old memories given the new input.

$$m_{S(x)} = I(x)$$

S(x) is the function to select memory location.

the simplest solution is to return the next empty slot.

QA memory networks

a simple implementation for text

O: (output feature map) – produces a new output, given the new input and the current memory state.

$$o_1 = O_1(x, m) = \text{argmax}_{i=1}^{N} s_O(x, m_i)$$

$$o_2 = O_2(x, m) = \text{argmax}_{i=1}^{N} s_O([x, m_{o_1}], m_i)$$

output: $[x, m_{o_1}, m_{o_2}]$

QA memory networks

a simple implementation for text

O: (output feature map) – produces a new output, given the new input and the current memory state.

$$o_1 = O_1(x, m) = \text{argmax}_{i=1}^{N} s_O(x, m_i)$$

$$o_2 = O_2(x, m) = \text{argmax}_{i=1}^{N} s_O([x, m_{o_1}], m_i)$$

output: $[x, m_{o_1}, m_{o_2}]$

R: (response) – converts the output into the response format desired. For example, a textual response or an action.

assume just output one word w:

$$r = \text{argmax}_{w \in W} s_R([x, m_{o_1}, m_{o_2}], w)$$

QA memory networks

example

question: x = “where is the milk now?”

supporting sentence m1 = “Joe left the milk”

supporting sentence m2 = “Joe travelled to the office”

output r = “office”

QA memory networks

scoring function

$$S(x, y) = \Phi(x)^T U^T U \Phi(y)$$

$\Phi(x)$ is bag of words representation.

learning

given questions, answers, as well as supporting sentences.
minimize over parameters U_O, U_R

$$\sum_{\bar{f} \neq f_1} \max(0, \gamma - s_O(x, f_1) + s_O(x, \bar{f})) +$$
$$\sum_{\bar{f}' \neq f_2} \max(0, \gamma - s_O([x, \mathbf{m}_{o_1}], f_2]) + s_O([x, \mathbf{m}_{o_1}], \bar{f}'])) +$$
$$\sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], r) + s_R([x, \mathbf{m}_{o_1}, \mathbf{m}_{o_2}], \bar{r}]))$$

QA memory networks

experiments

Joe went to the garden then Fred picked up the milk; Joe moved to the bathroom and Fred dropped the milk, and then Dan moved to the living_room.
Where is Dan? A: living room I believe
Where is Joe? A: the bathroom
Fred moved to the bedroom and Joe went to the kitchen then Joe took the milk there and Dan journeyed to the bedroom; Joe discarded the milk.
Where is the milk now ? A: the milk is in the kitchen
Where is Dan now? A: I think he is in the bedroom
Joe took the milk there, after that Mike travelled to the office, then Joe went to the living_room, next Dan went back to the kitchen and Joe travelled to the office.
Where is Joe now? A: I think Joe is in the office

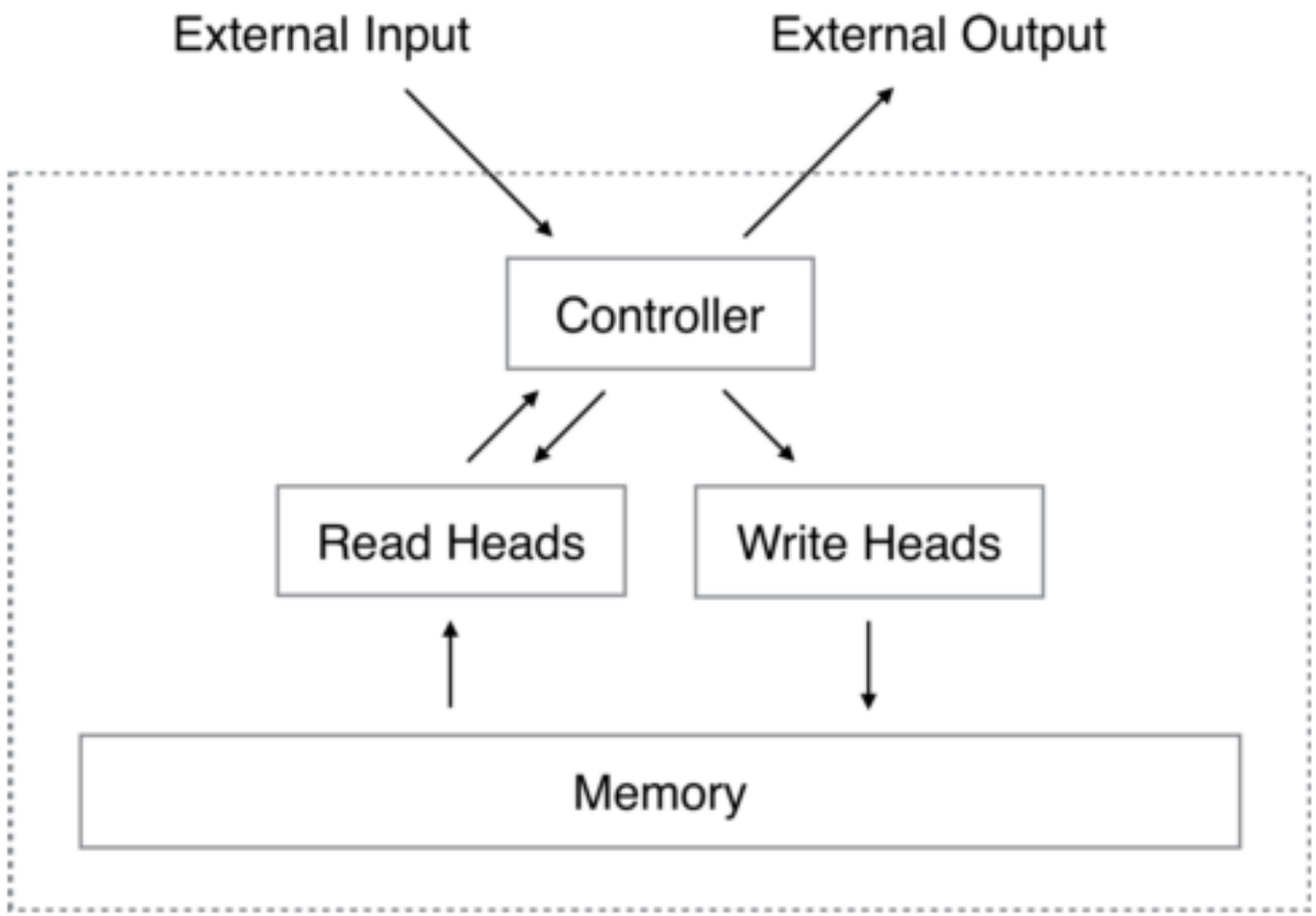
neural turing machine

In QA memory network, memory is mainly used for a knowledge database. Interaction between computation resources and memory is very limited.

neural turing machine proposes an addressing mechanism as well as coupled reading & writing operations.

neural turing machine

machine architecture



neural turing machine

Let M_t be the memory matrix of size NxM, where N is the number of memory locations, and M is the vector size at each location.

Read:

$$\sum_i w_t(i) = 1, \quad 0 \leq w_t(i) \leq 1$$

$$r_t \leftarrow \sum_i w_t(i) M_t(i)$$

Write:

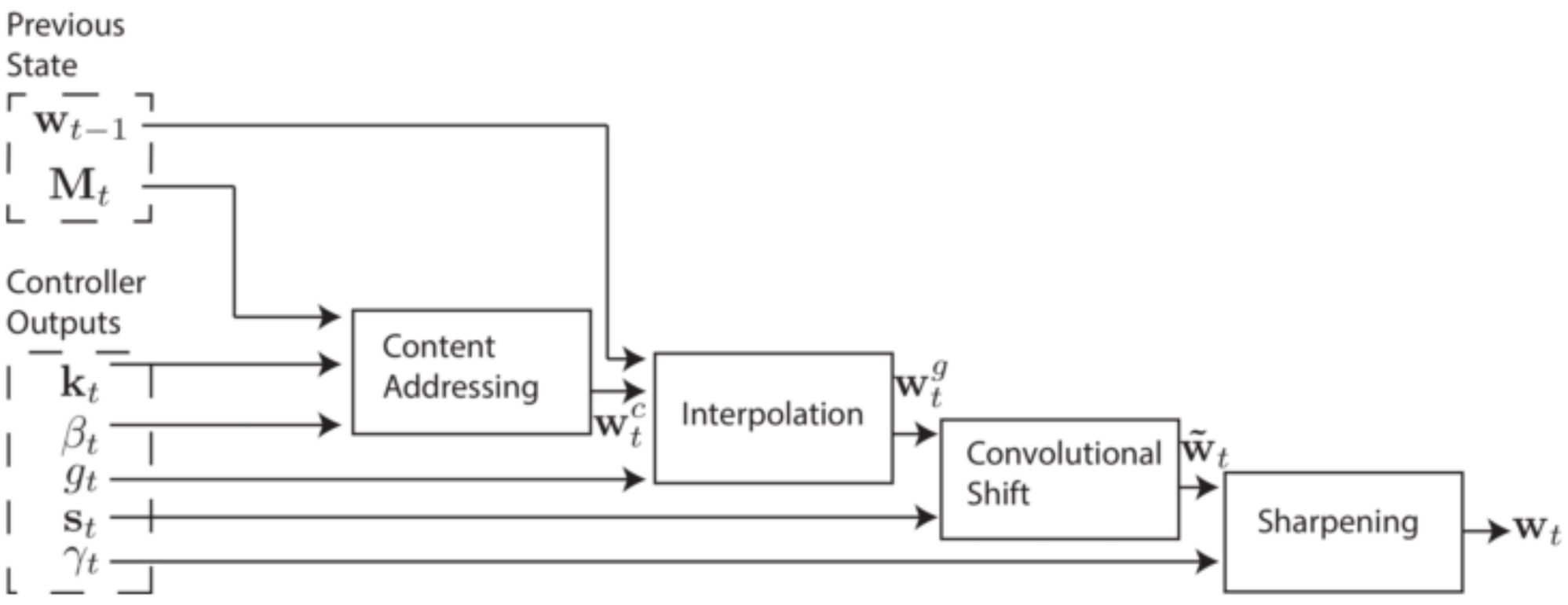
erase: $\tilde{M}_t(i) \leftarrow M_{t-1}(i)[1 - w_t(i)e_t]$

add: $M_t(i) \leftarrow \tilde{M}_t(i) + w_t(i)a_t$

neural turing machine

addressing mechanisms

content-based and location-based addressing



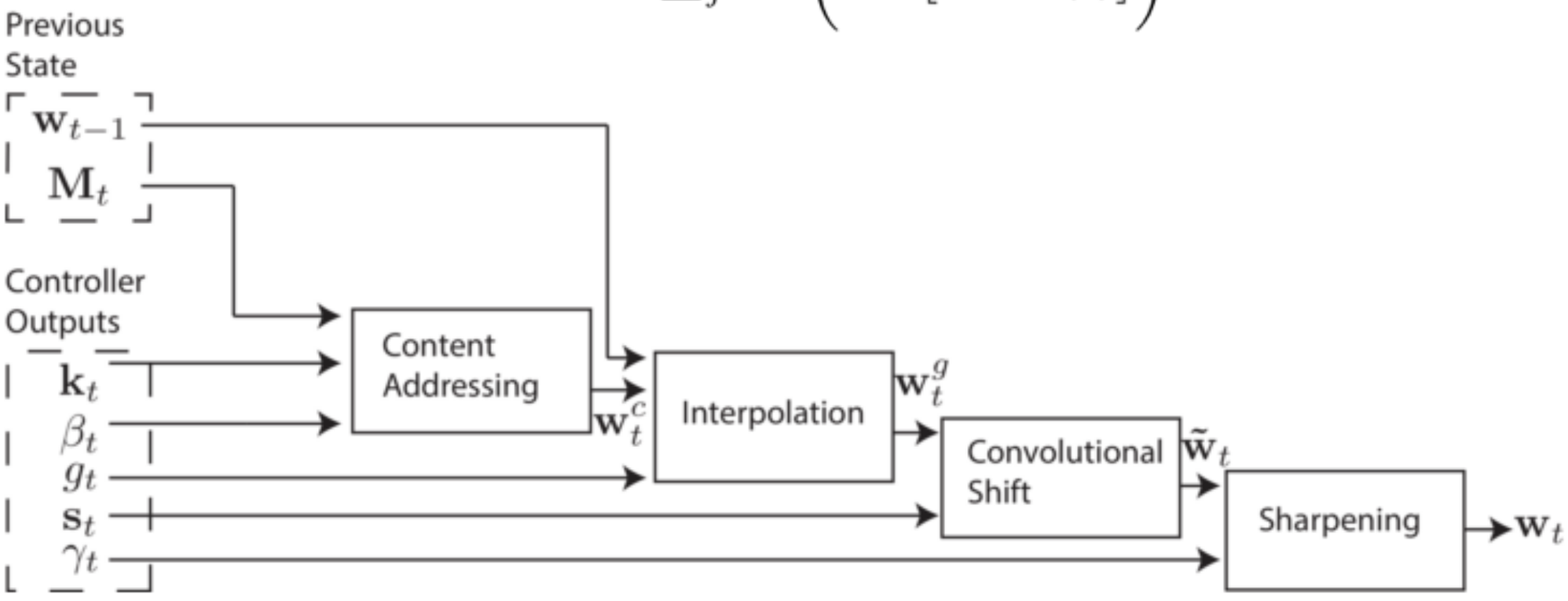
neural turing machine

addressing mechanisms

1. content-based

k_t key vector. β_t key strength.

$$w_t^c(i) \leftarrow \frac{\exp\left(\beta_t K\left[\mathbf{k}_t, \mathbf{M}_t(i)\right]\right)}{\sum_j \exp\left(\beta_t K\left[\mathbf{k}_t, \mathbf{M}_t(j)\right]\right)}$$



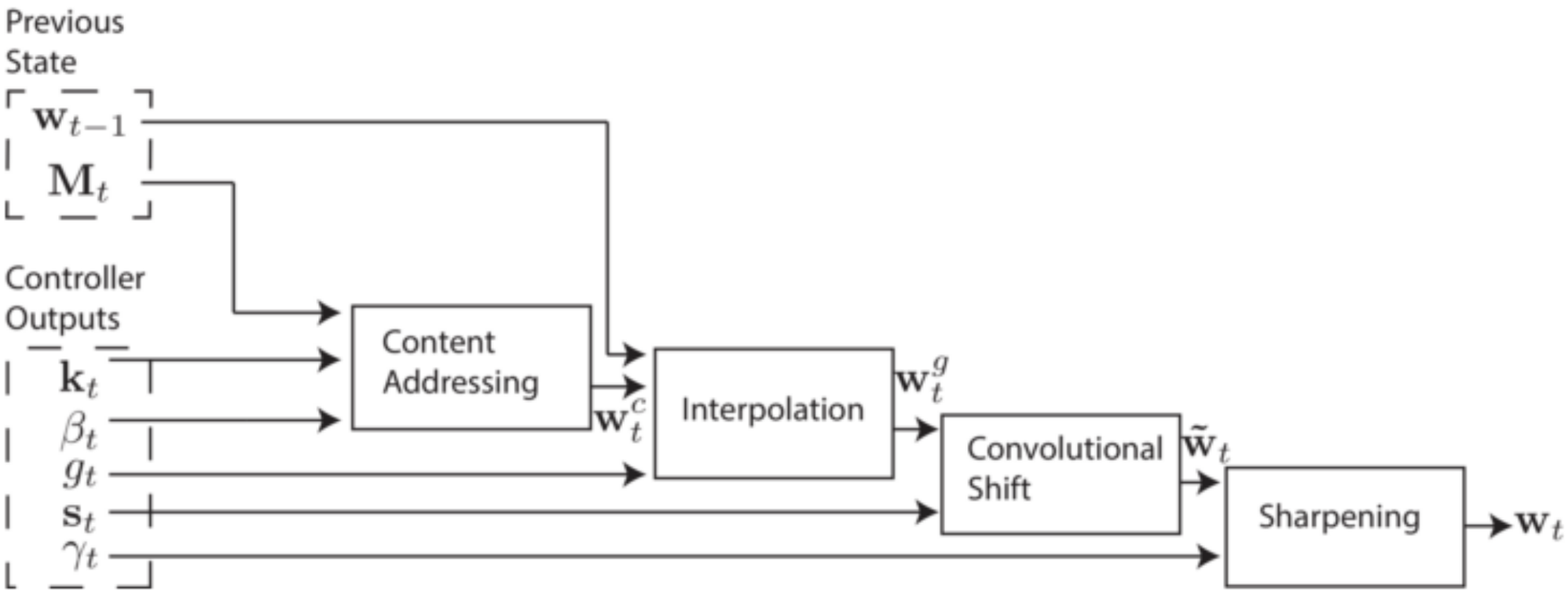
neural turing machine

addressing mechanisms

2. interpolation

g_t interpolation gate

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



neural turing machine

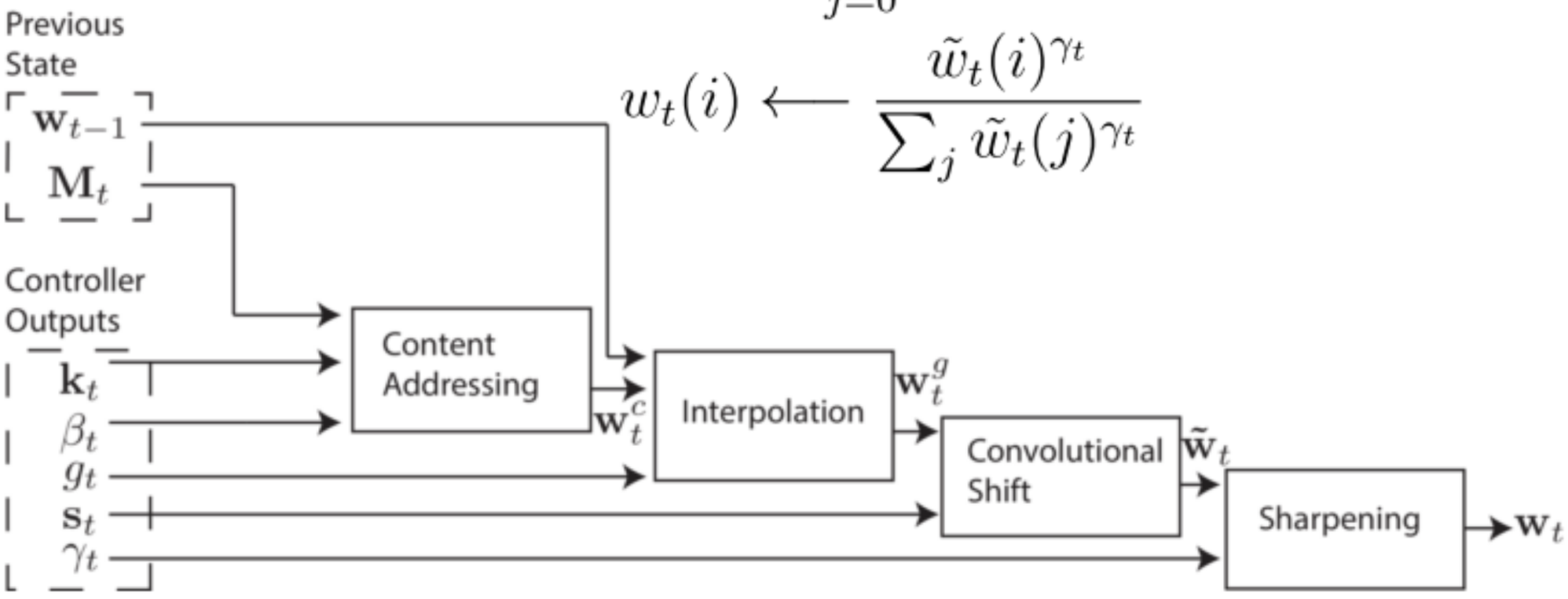
addressing mechanisms

3. shifting and sharpening

s_t shift weighting γ_t sharpening scalar

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

$$w_t(i) \longleftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$



neural turing machine

Addressing Mechanisms

operate in 3 complementary modes:

- weights can be chosen only by the content system without any modification of location system.
- weights from the content system can be chosen and then shifted. Find a contiguous block of data, then assess a particular element.
- weights from previous time step can be rotated without any input from the content-based address. Allows iteration.

neural turing machine

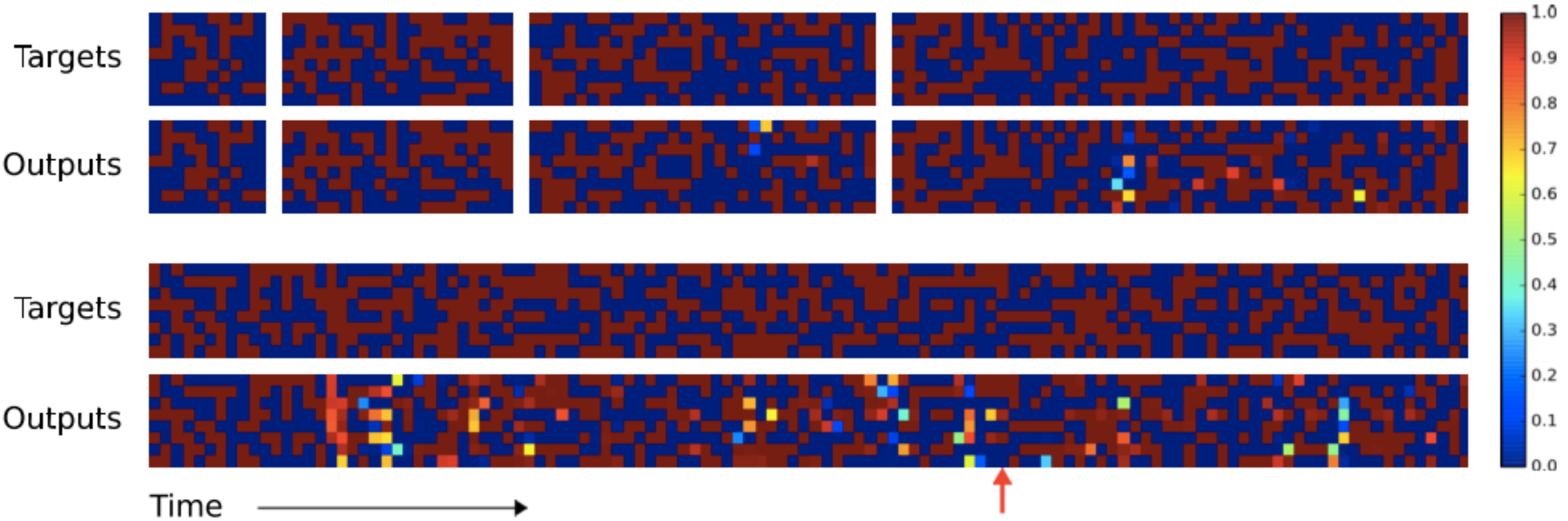
Controller network

Given the input signal, decide the addressing variables.

- a feedforward neural network
- a recurrent neural network
  - allow the controller to mix information across time.
  - If one compares the controller to the CPU in a digital computer, memory unit to RAM, the hidden states of the controller are akin to registers in the CPU.

neural turing machine

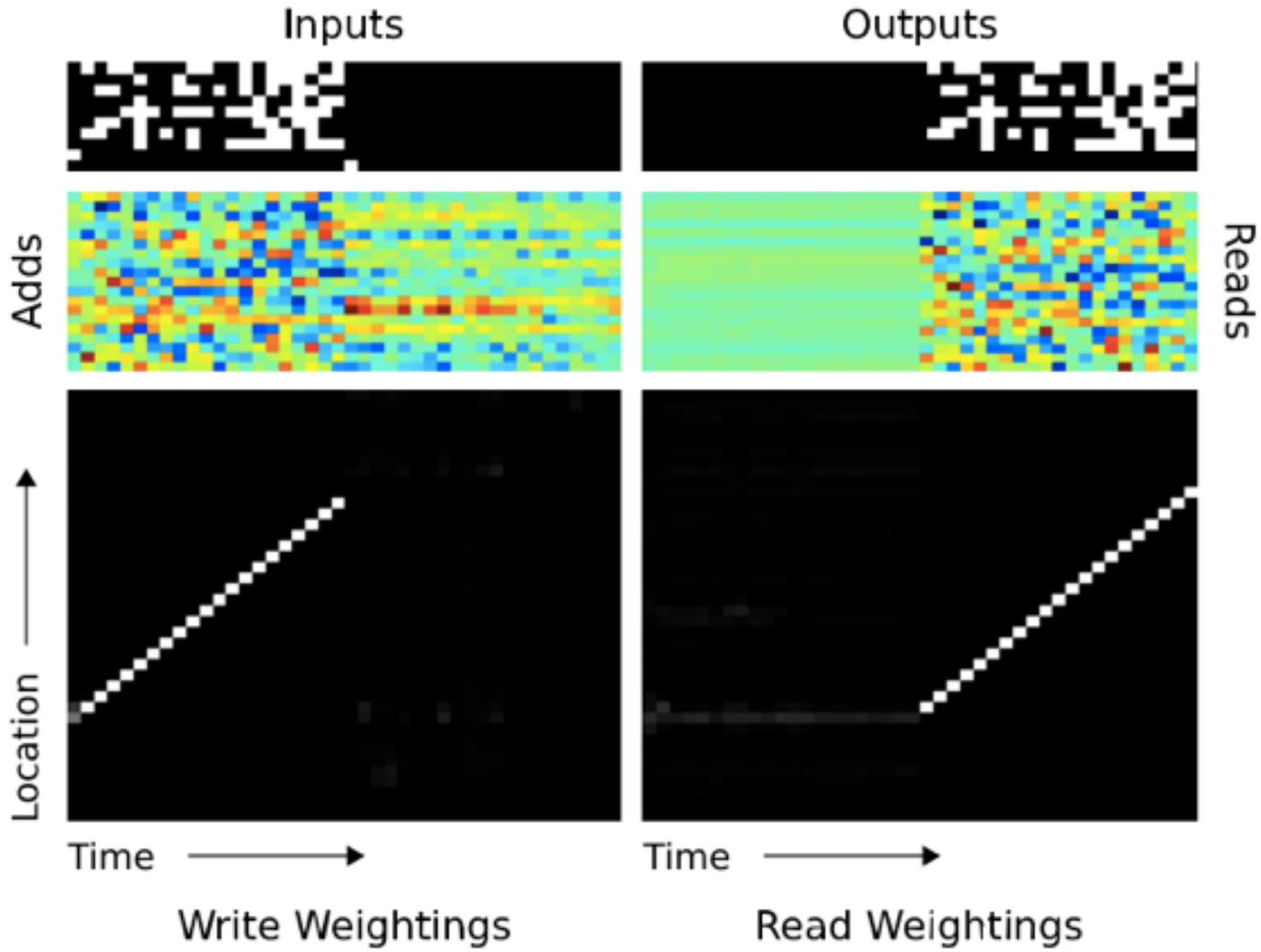
Copy: NTM is presented with an input sequence of random binary vectors, and asked to recall it.



neural turing machine

Copy: intermediate variables suggest the following copy algorithm.

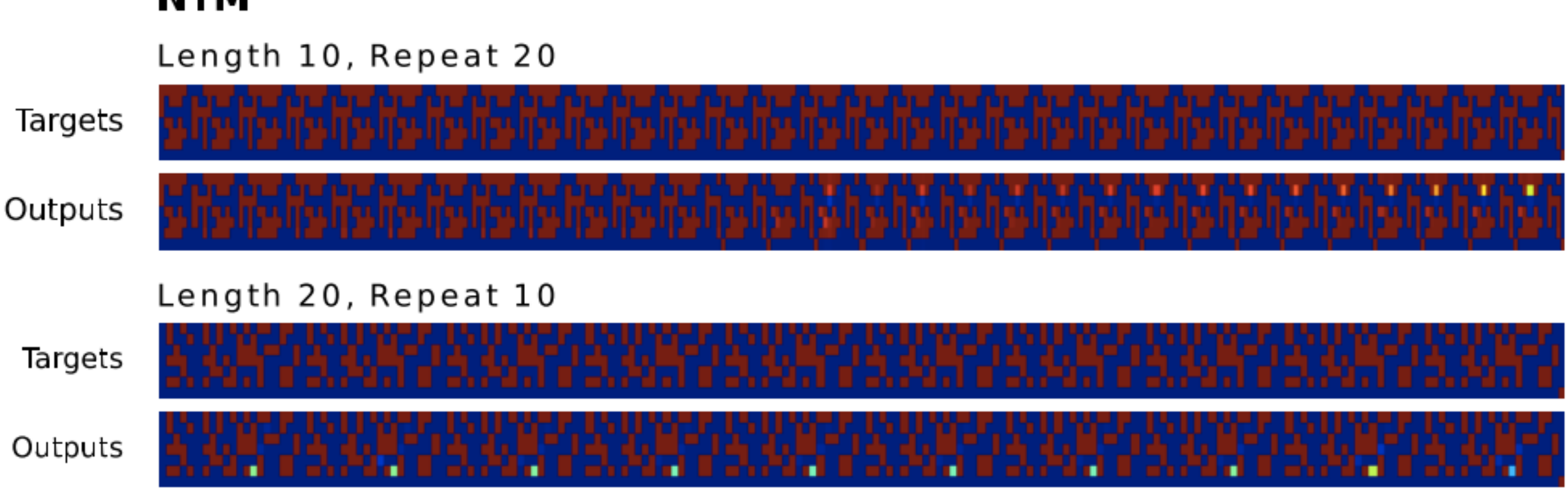
initialise: move head to start location
while input delimiter not seen **do**
    receive input vector
    write input to head location
    increment head location by 1
end while
return head to start location
while true **do**
    read output vector from head location
    emit output
    increment head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



neural turing machine

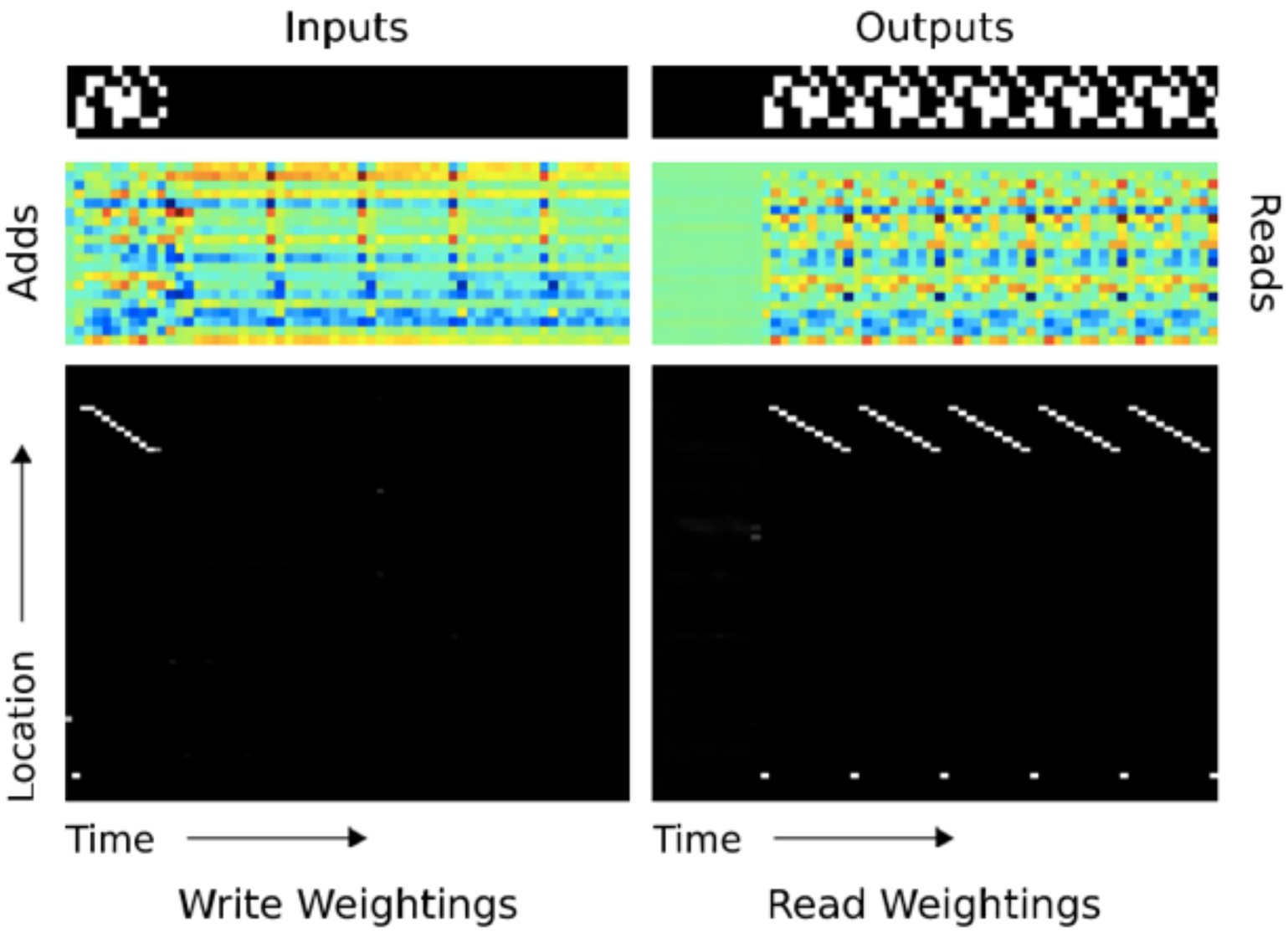
Repeated copy

NTM is presented with an input sequence
and a scalar indicating the number of copies.
To test if NTM can learn simple nested “for loop”



neural turing machine

Repeated copy

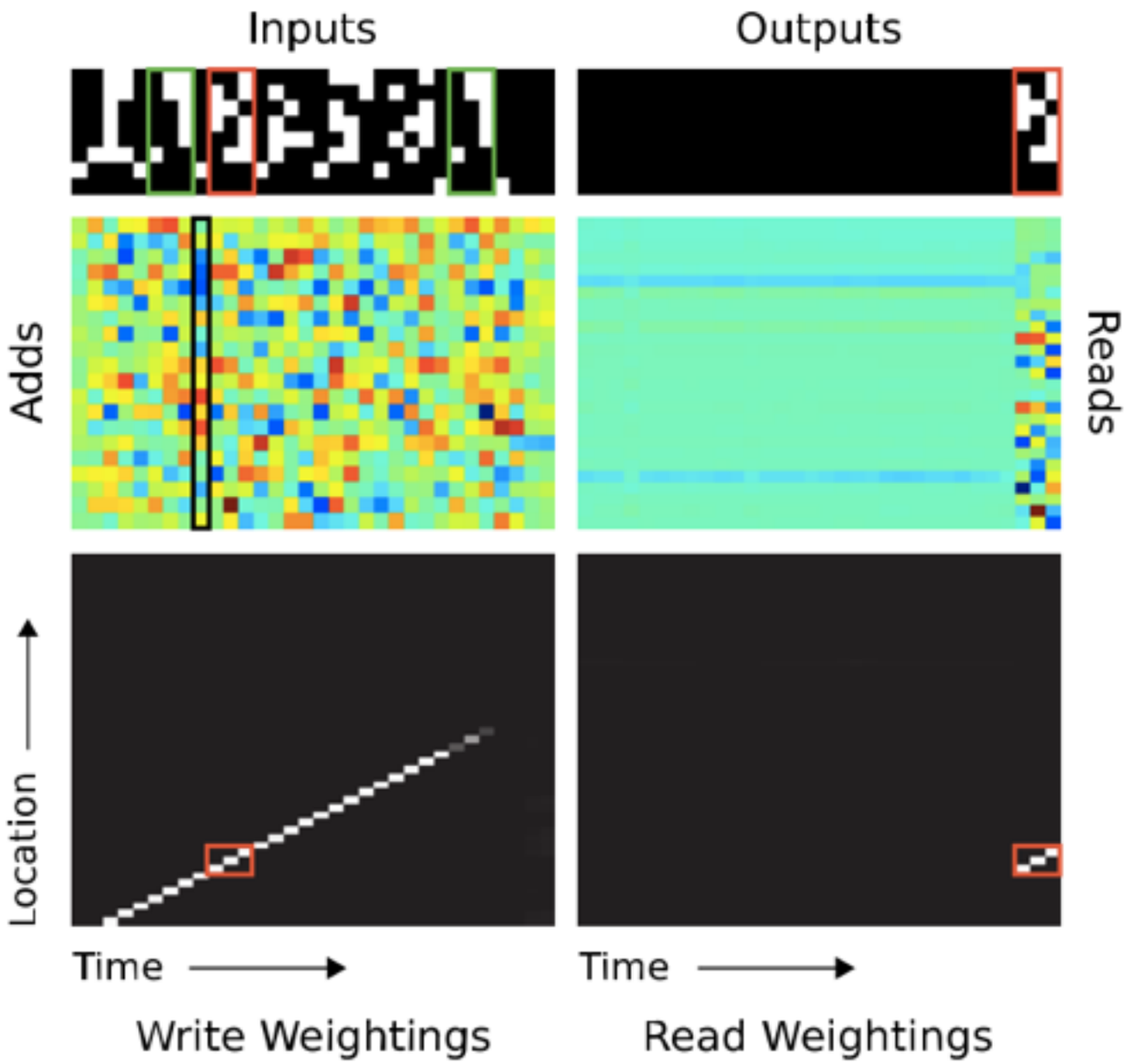


- fails to figure out where to end. Unable to keep count of how many repeats it has completed.
- Use another memory location to help switch back the pointer to the start.

neural turing machine

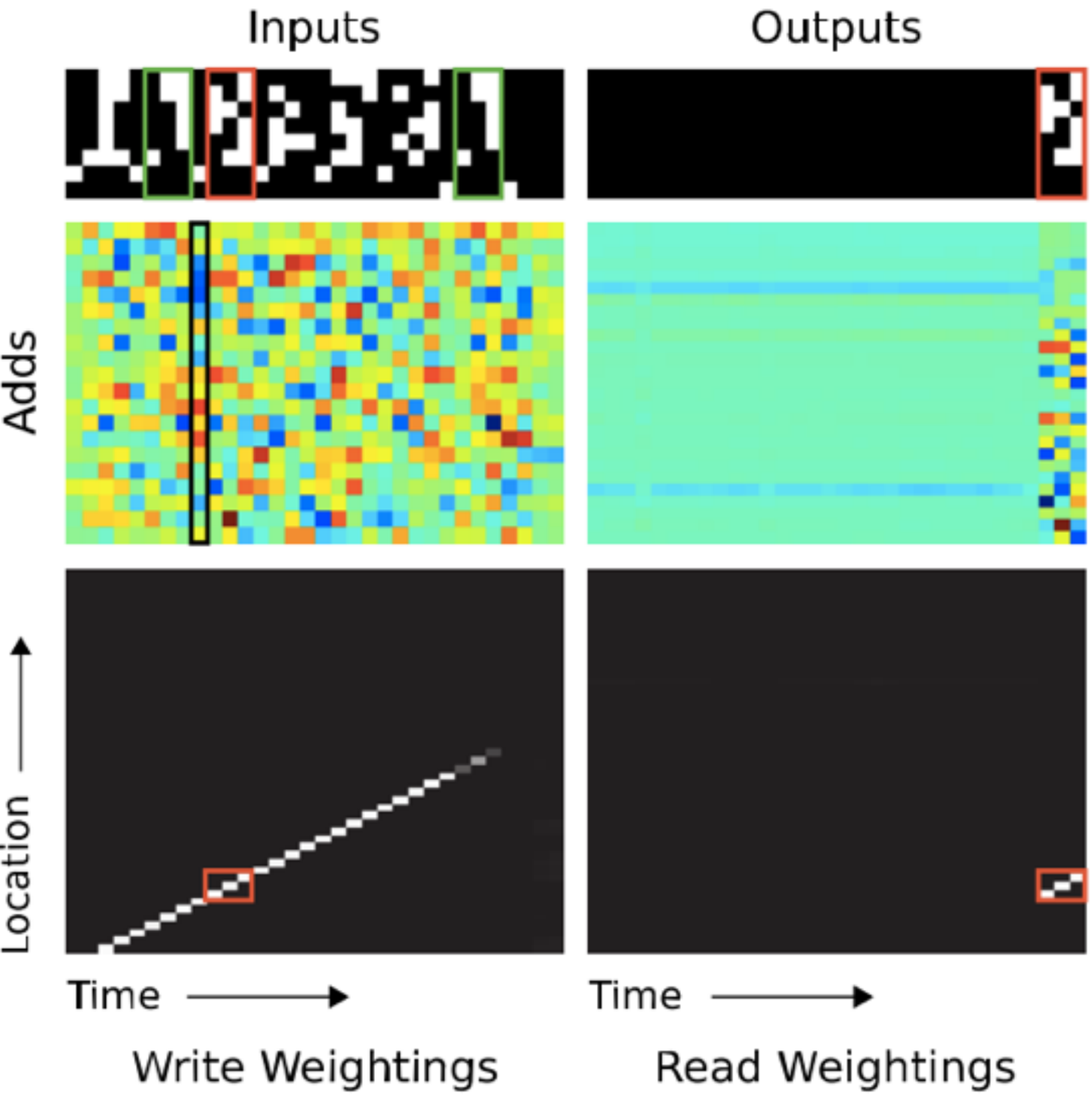
Associative Recall

NTM is presented with a sequence and a query, then it is asked to output datum behind the query.
To test if NTM can apply algorithms to relatively simple, linear data structures.



neural turing machine

Associative Recall

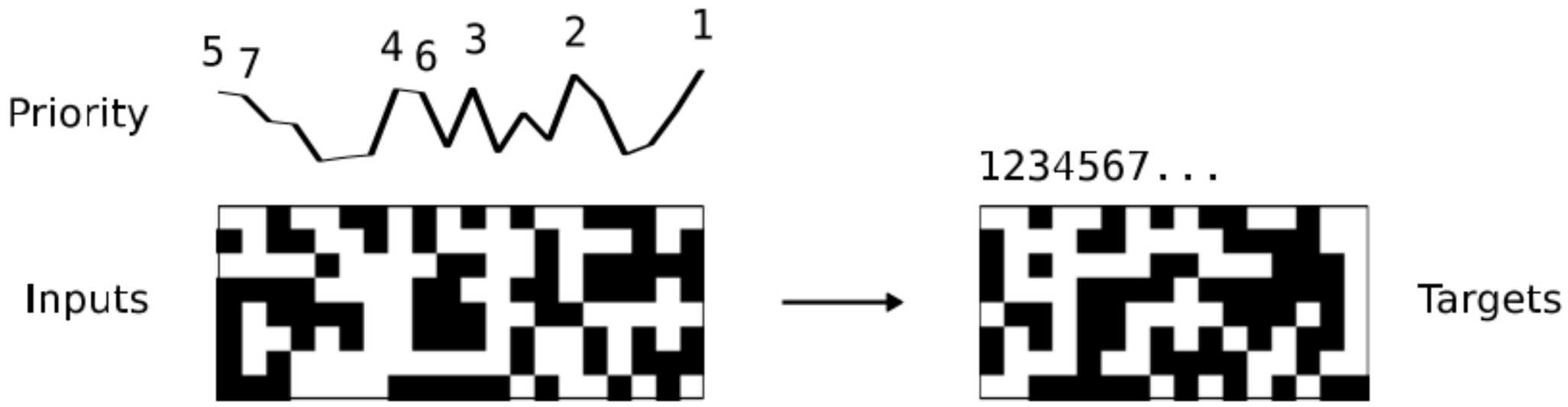


- when each item delimiter is presented, the controller writes a compressed representation of the previous three time slices of the item.
- After the query arrives, the controller recomputes the same compressed representation of the query item, uses a content-based lookup to find the location where it wrote the first representation, and then shifts by one to produce the subsequent item in the sequence

neural turing machine

Priority Sort

A sequence of random binary vectors is input to the network along with a scalar priority rating for each vector.

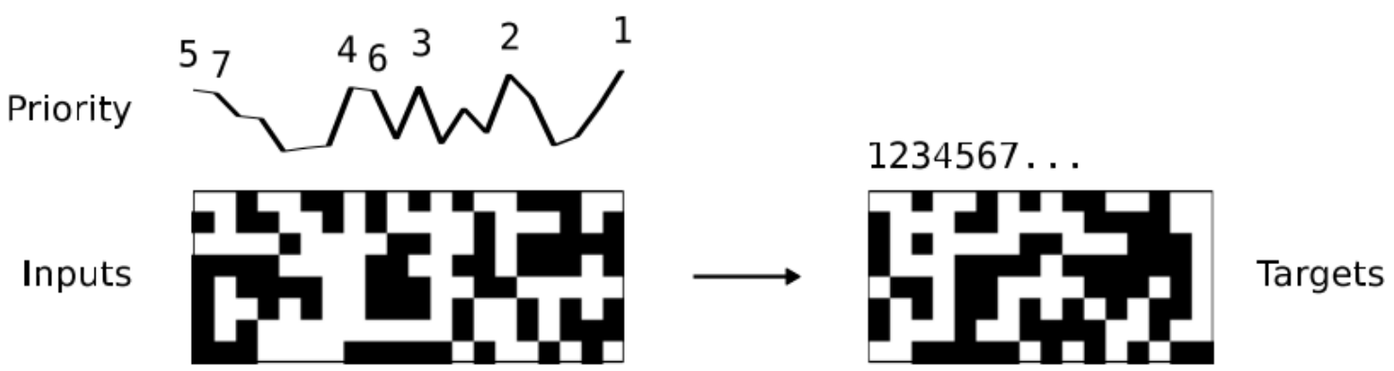


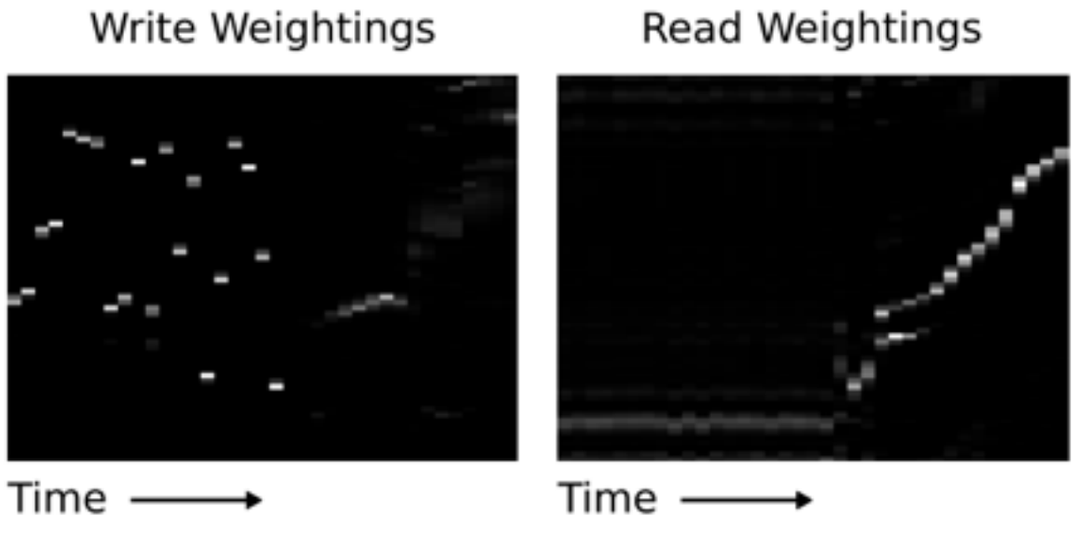
neural turing machine

Priority Sort

hypothesis that NTM uses the priorities to determine the relative location of each write.

The network reads from the memory location in an increasing order.





neural turing machine

theano code available:

https://github.com/shawntan/neural-turing-machines

Thanks!