Reasoning, Attention and Memory Based Machine Learning Models

Prepared By:

Dhruv Kohli, Dept. of Mathematics

Content

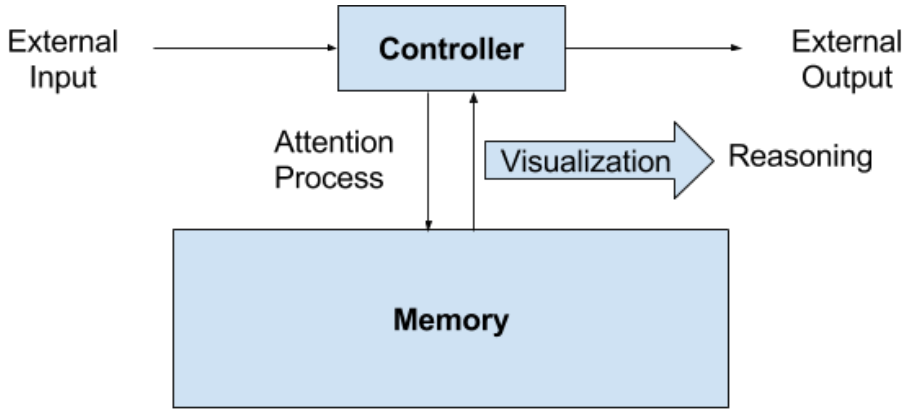
- Introduction
- Neural Turing Machine
- End to End Memory Networks
- Similarity between NTM and E2EMemNN
- Game Playing Agent with RAM

Task
Model
Training
Results

Aim
Issue with state-of-the-art
Resolution with RAM

Introduction

- The models discussed comprises of a memory component and a controller, such as an artificial neural network, which
  - Takes inputs from external world
  - Stores a representation or encoding of those inputs into the **memory**
  - Interacts with the memory using a defined mechanism called **attention process**
  - produces outputs for the external world with **reasoning** where the reasoning follows from the graphical visualization of the interaction between controller and memory.



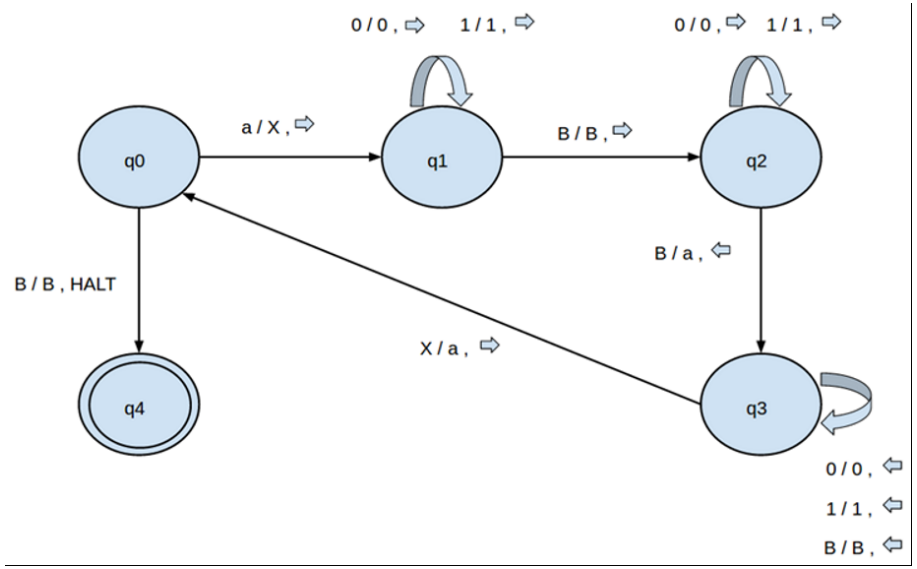
Introduction

- In this thesis, I focus on applying RAM based machine learning models in the domain of sequence to sequence learning, question answering task and building game playing agents.

Neural Turing Machine - Task

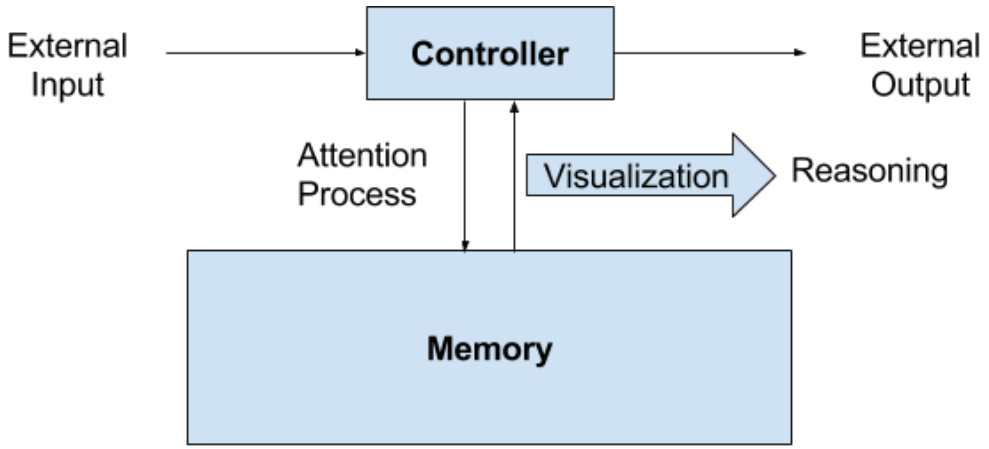
- NTM is used to learn algorithm mapping an sequence of inputs to a sequence of outputs.
- For ex: in Copy Task, where
  - input is B10110B BBBBB
  - output should be BBB……B 10110

Following is the transition diagram for the copy task:



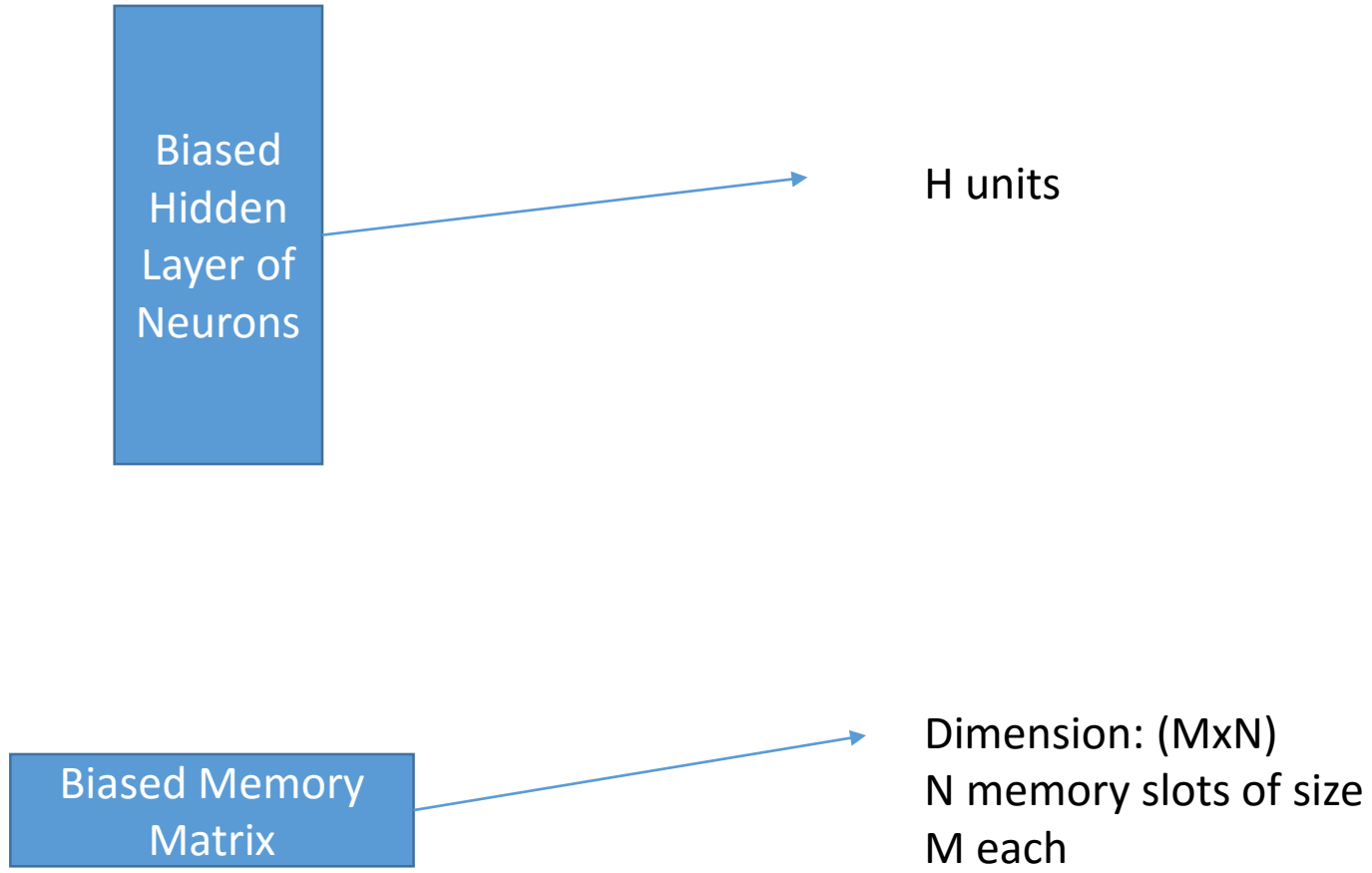
Neural Turing Machine - Model

- Unlike Turing Machine where the controller is defined (in the sense that the controller know the transition function), in NTM, the controller is learnt (in the sense that the transition function is learnt).
- "Learning a function", in statistical terms, is same as "approximating a function", which further reduces to "approximating the parameters of the function" given that the approach being followed for approximating the function is parametric.

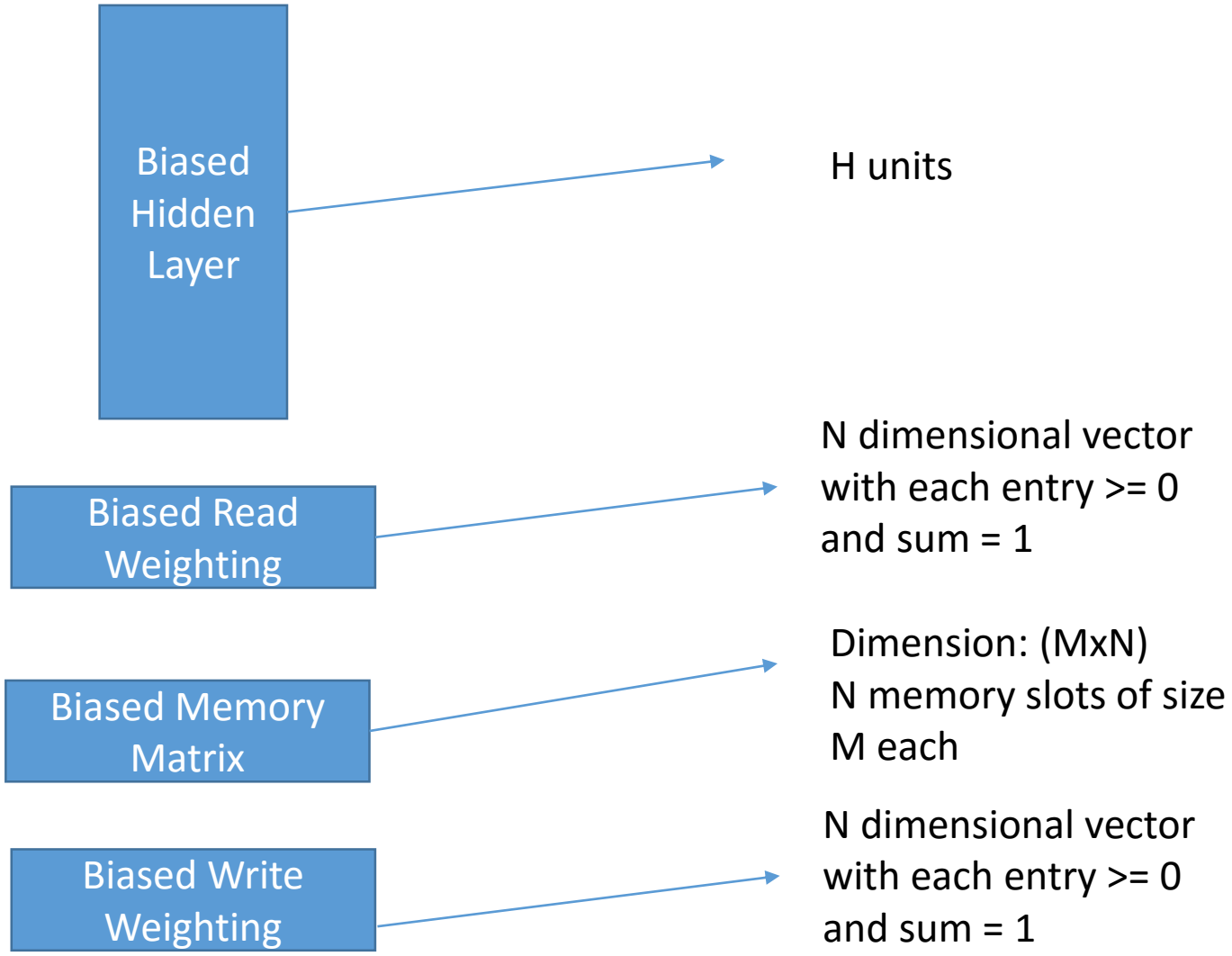


Architecture of NTM.

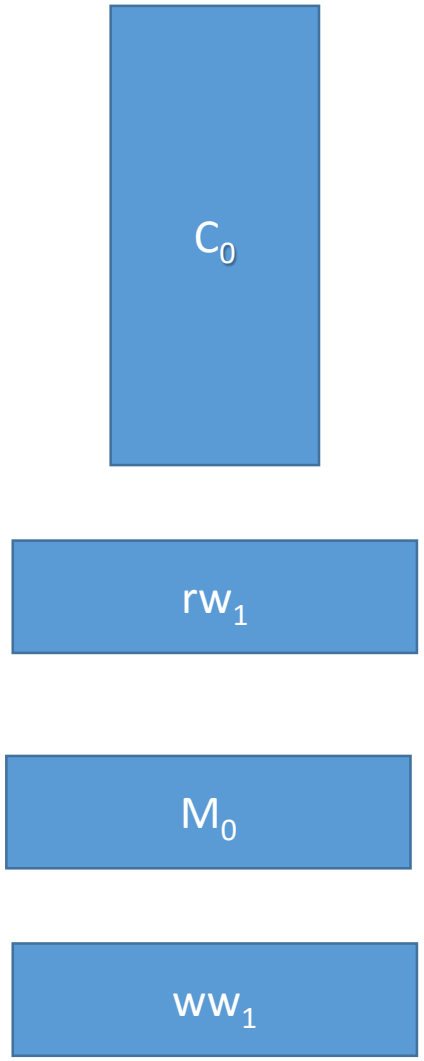
Neural Turing Machine - Model



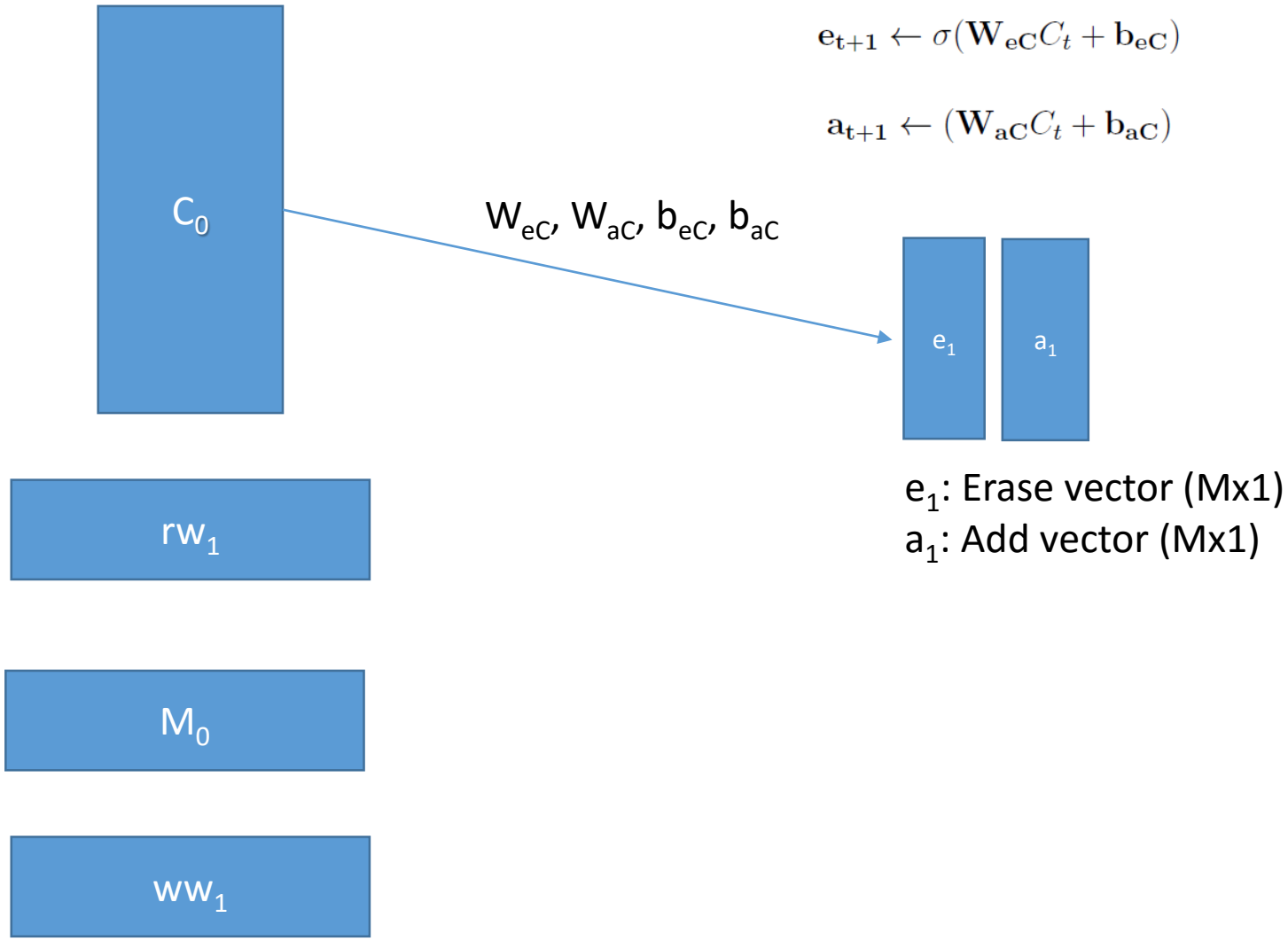
Neural Turing Machine - Model



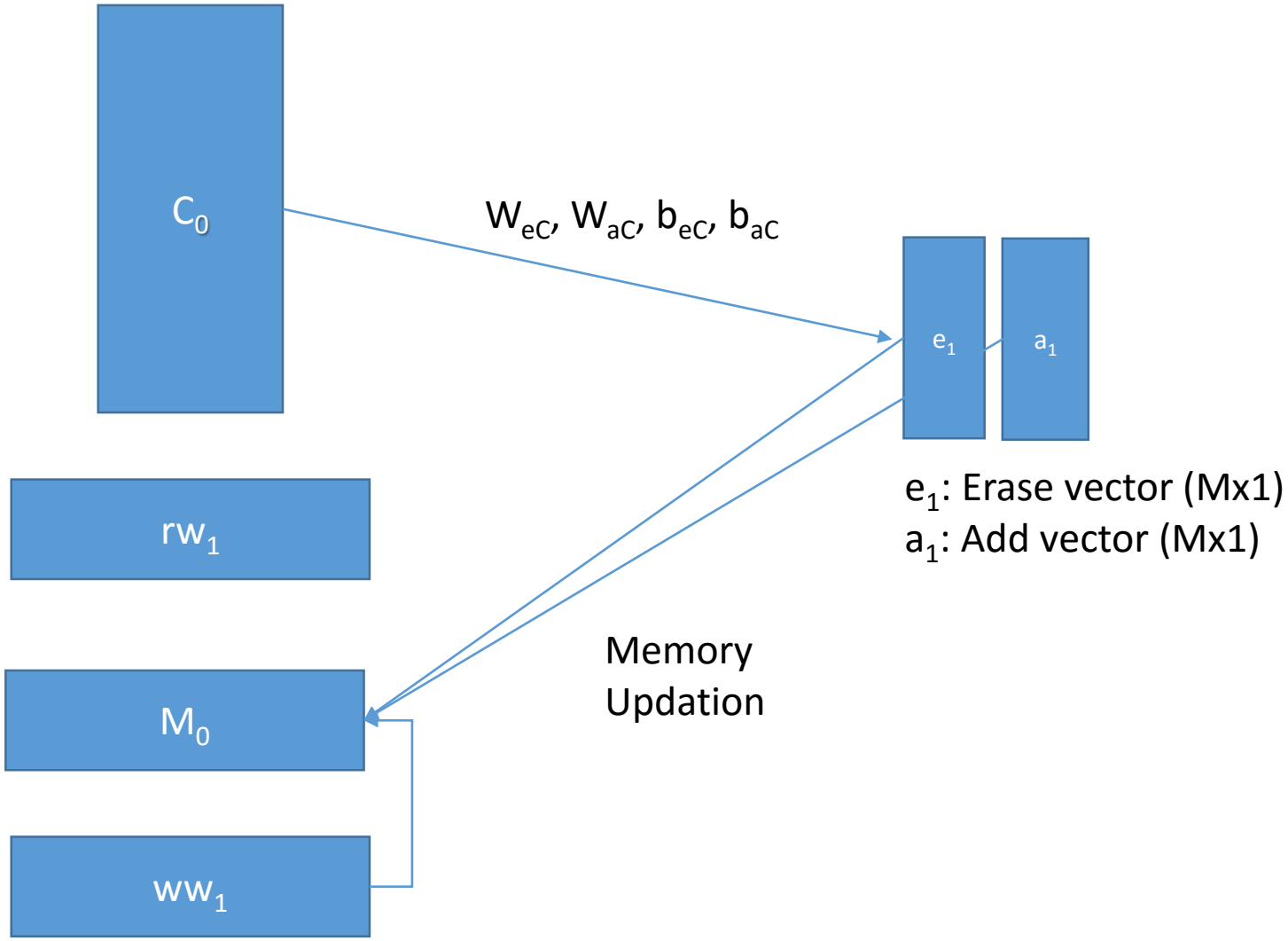
Neural Turing Machine - Model



Neural Turing Machine - Model



Neural Turing Machine - Model

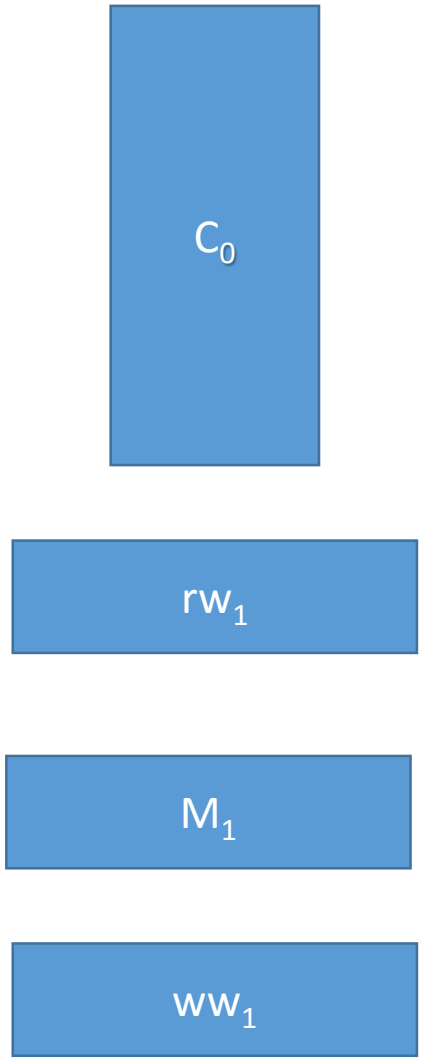


Neural Turing Machine – Model (Memory Updation)

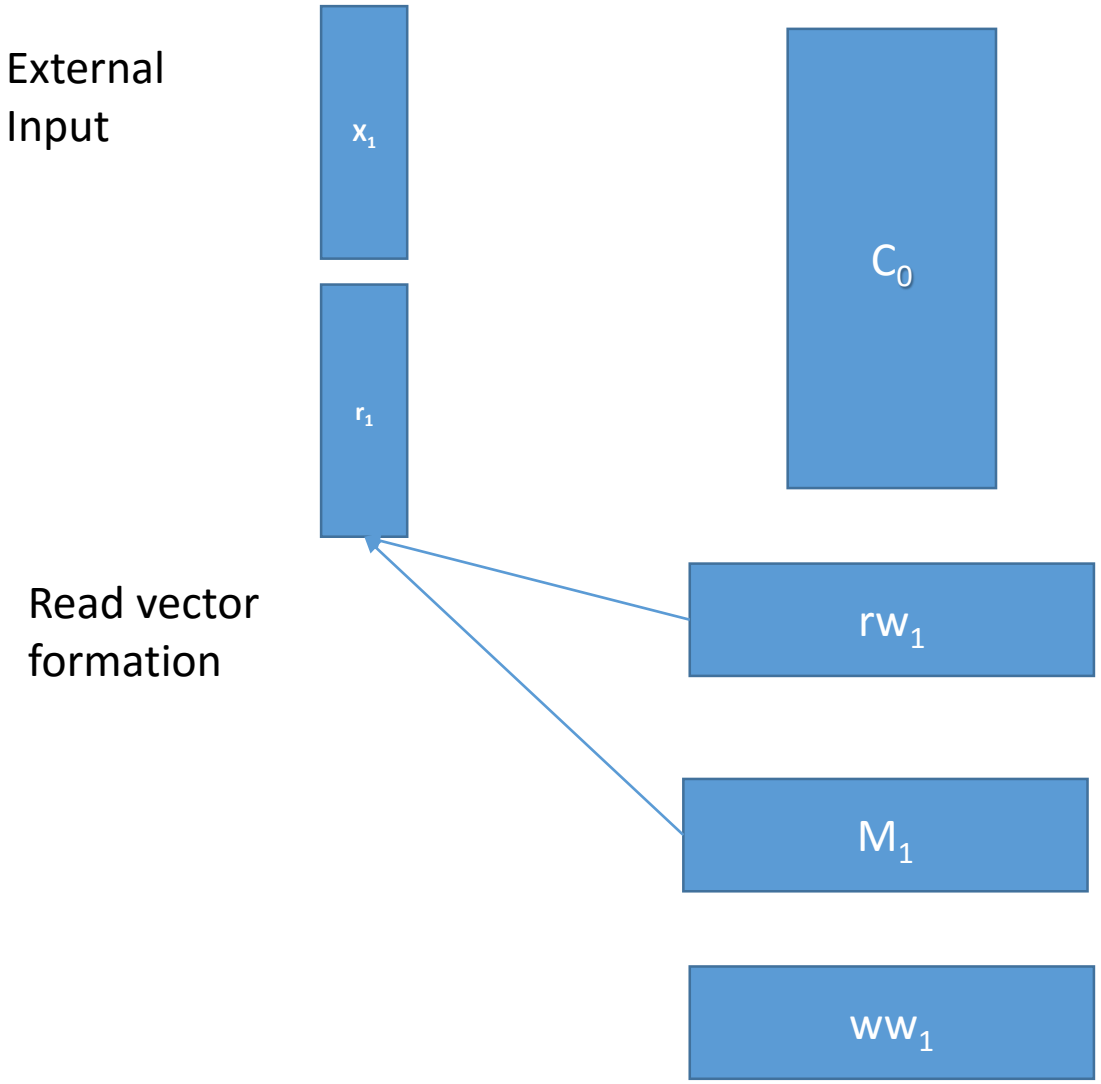
$$\widetilde{\mathbf{M}}_t(i) \leftarrow \mathbf{M}_{t-1}(i)[1 - ww_t(i)\mathbf{e}_t]$$

$$\mathbf{M}_t(i) \leftarrow \widetilde{\mathbf{M}}_t(i) + ww_t(i)\mathbf{a}_t$$

Neural Turing Machine - Model



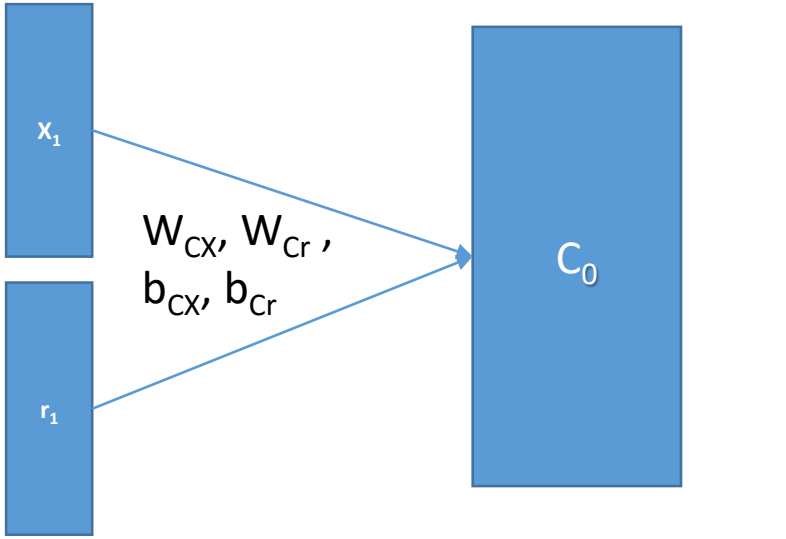
Neural Turing Machine - Model



Neural Turing Machine – Model (Read Vector Formation)

$$\mathbf{r}_t \leftarrow \sum_i rw_t(i)\mathbf{M}_t(i)$$

Neural Turing Machine - Model



$$C_t \leftarrow relu(\mathbf{W_{CX}}X_t + \mathbf{b_{CX}} + \mathbf{W_{Cr}}\mathbf{r_t} + \mathbf{b_{Cr}})$$

rw_1

M_1

ww_1

Neural Turing Machine - Model

$$P_t \leftarrow \sigma(\mathbf{W_{PC}}C_t + \mathbf{b_{PC}})$$

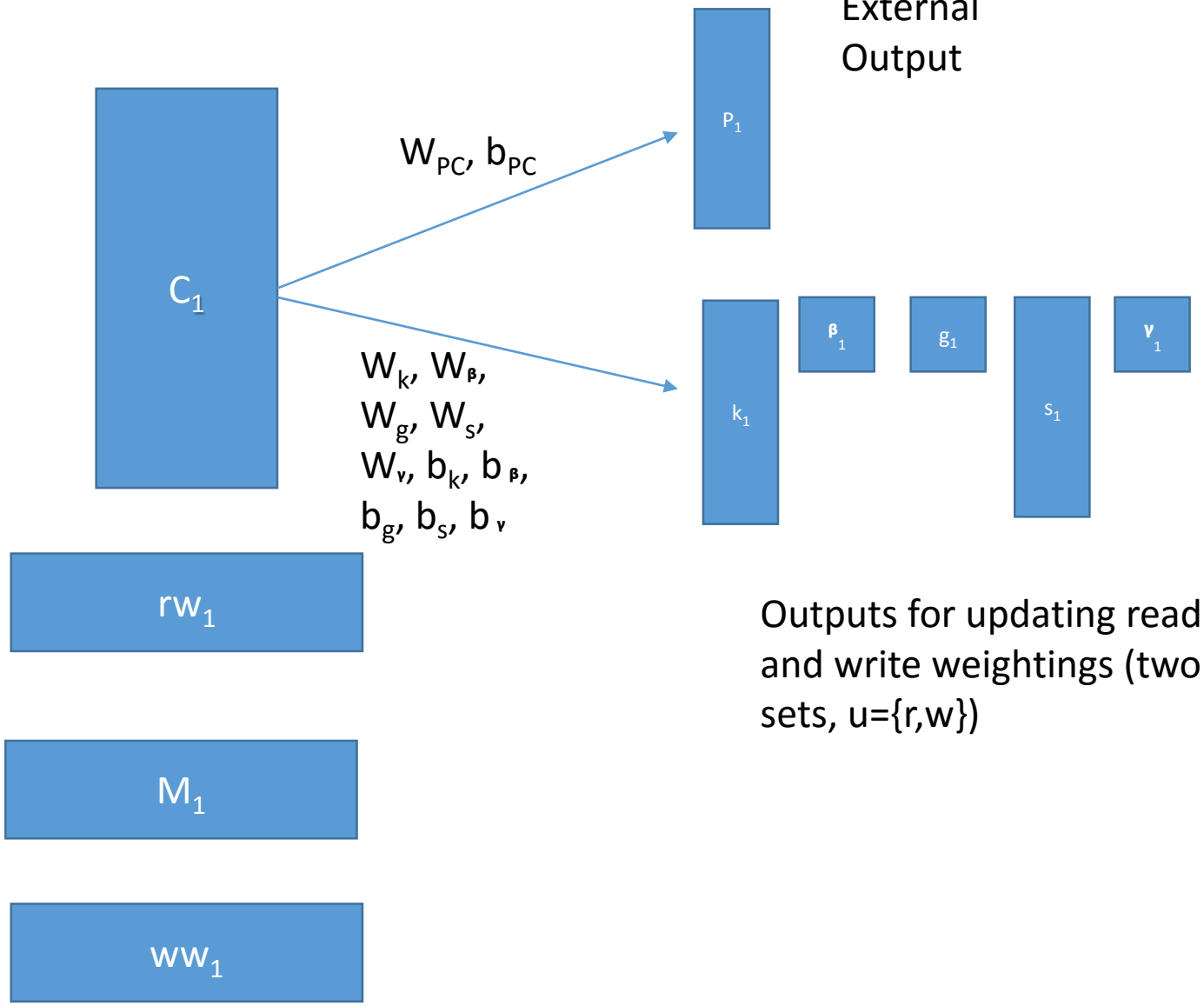
$$\mathbf{k_{u,t}} \leftarrow \mathbf{W_{k_uC}}C_t + \mathbf{b_{k_uC}}$$

$$\beta_{u,t} \leftarrow relu(\mathbf{W}_{\beta_u\mathbf{C}}C_t + \mathbf{b}_{\beta_u\mathbf{C}})$$

$$g_{u,t} \leftarrow \sigma(\mathbf{W_{g_uC}}C_t + \mathbf{b_{g_uC}})$$

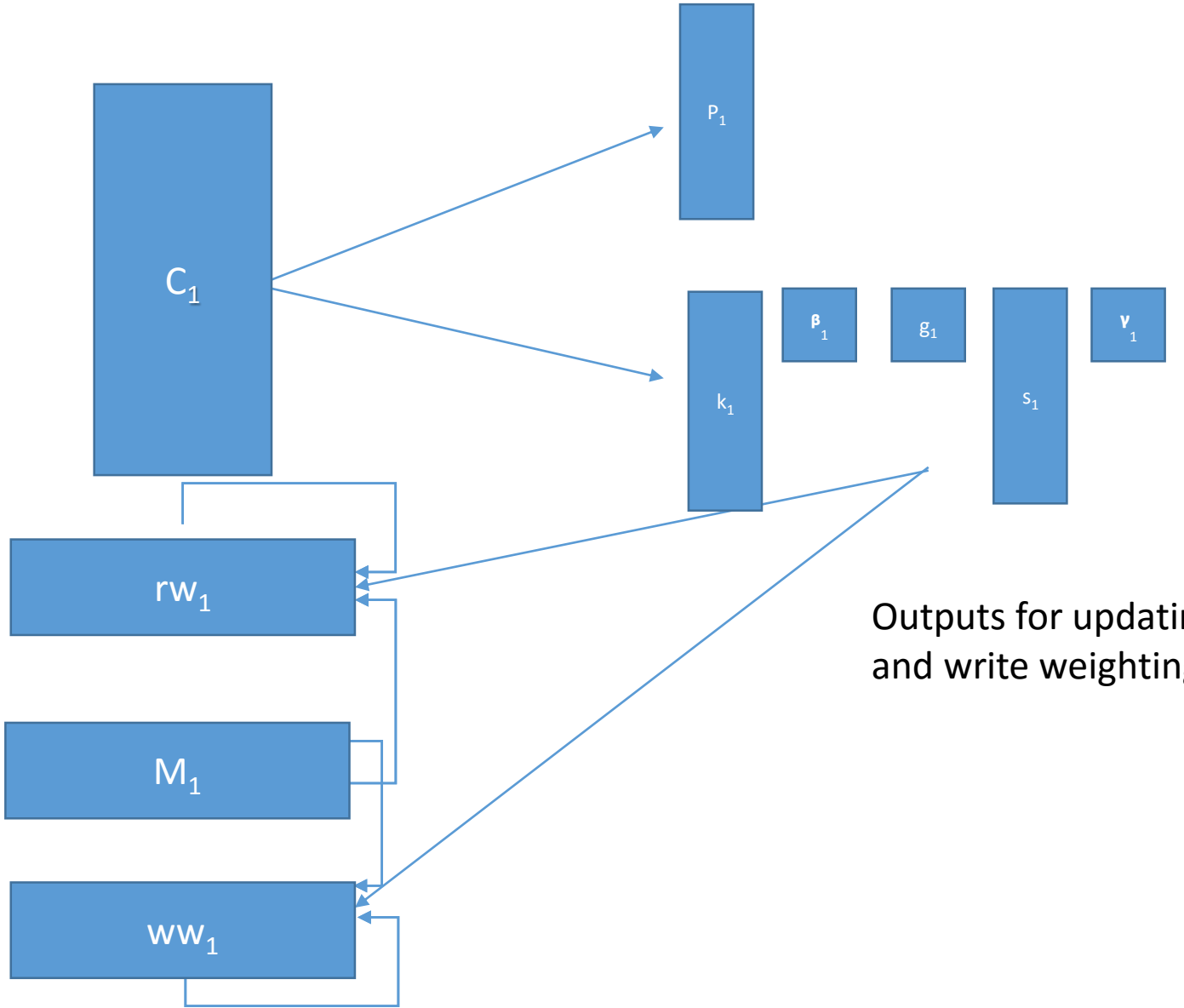
$$\mathbf{s_{u,t}} \leftarrow \sigma(\mathbf{W_{s_uC}}C_t + \mathbf{b_{s_uC}})$$

$$\gamma_{u,t} \leftarrow relu(\mathbf{W}_{\gamma_u\mathbf{C}}C_t + \mathbf{b}_{\gamma_u\mathbf{C}})$$



Outputs for updating read and write weightings (two sets, u={r,w})

Neural Turing Machine - Model



Neural Turing Machine – Model (Weight Updation)

Updation outputs, $\mathbf{k_t}$, $\boldsymbol{\beta_t}$, $\mathbf{g_t}$, $\mathbf{s_t}$ and $\boldsymbol{\gamma_t}$ are used.

$\mathbf{k_t}$ = Key vector (Mx1)

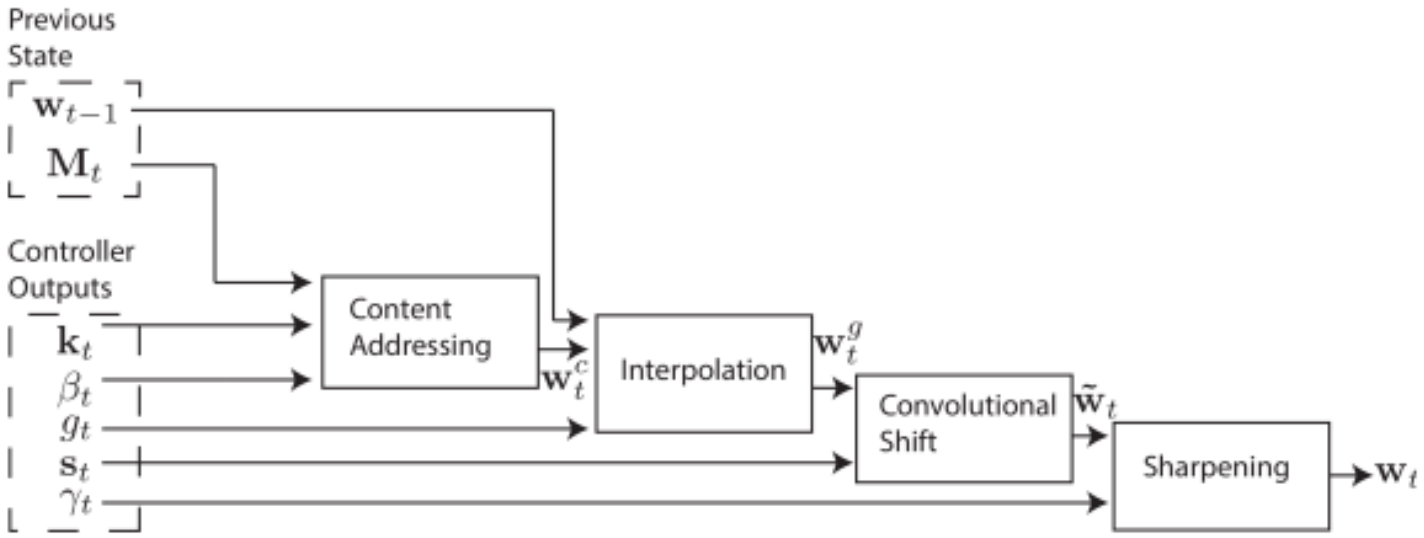
$\boldsymbol{\beta_t}$ = Key strength scalar

$\mathbf{g_t}$ = Gate scalar for interpolation

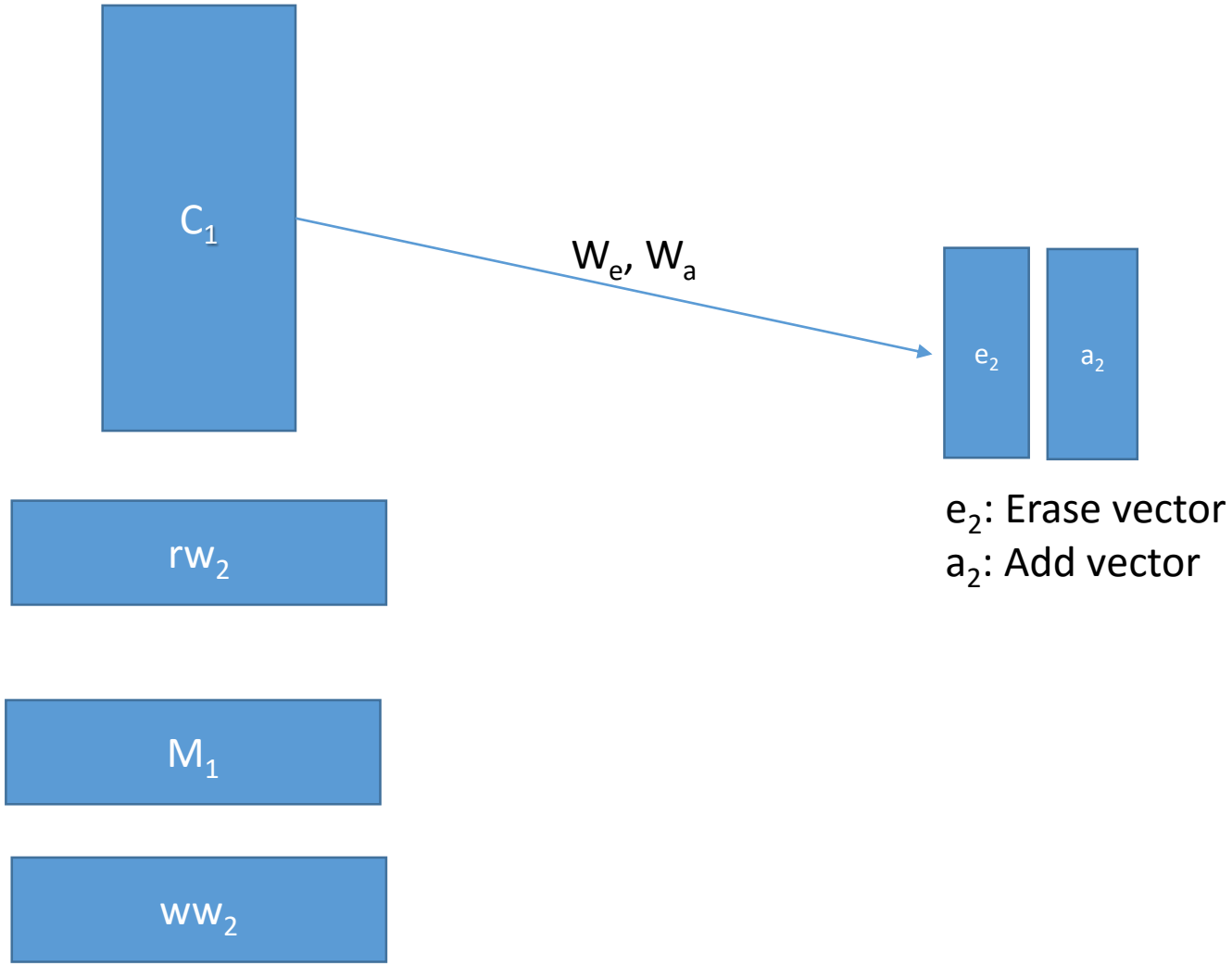
$\mathbf{s_t}$ = Shift weighting vector (number-of-allowed-shiftx1)

$\boldsymbol{\gamma_t}$ = sharpening scalar

Hyper-parameter



Neural Turing Machine - Model



e_2: Erase vector
a_2: Add vector

Neural Turing Machine - Model

And the whole process repeats until the external input is exhausted.

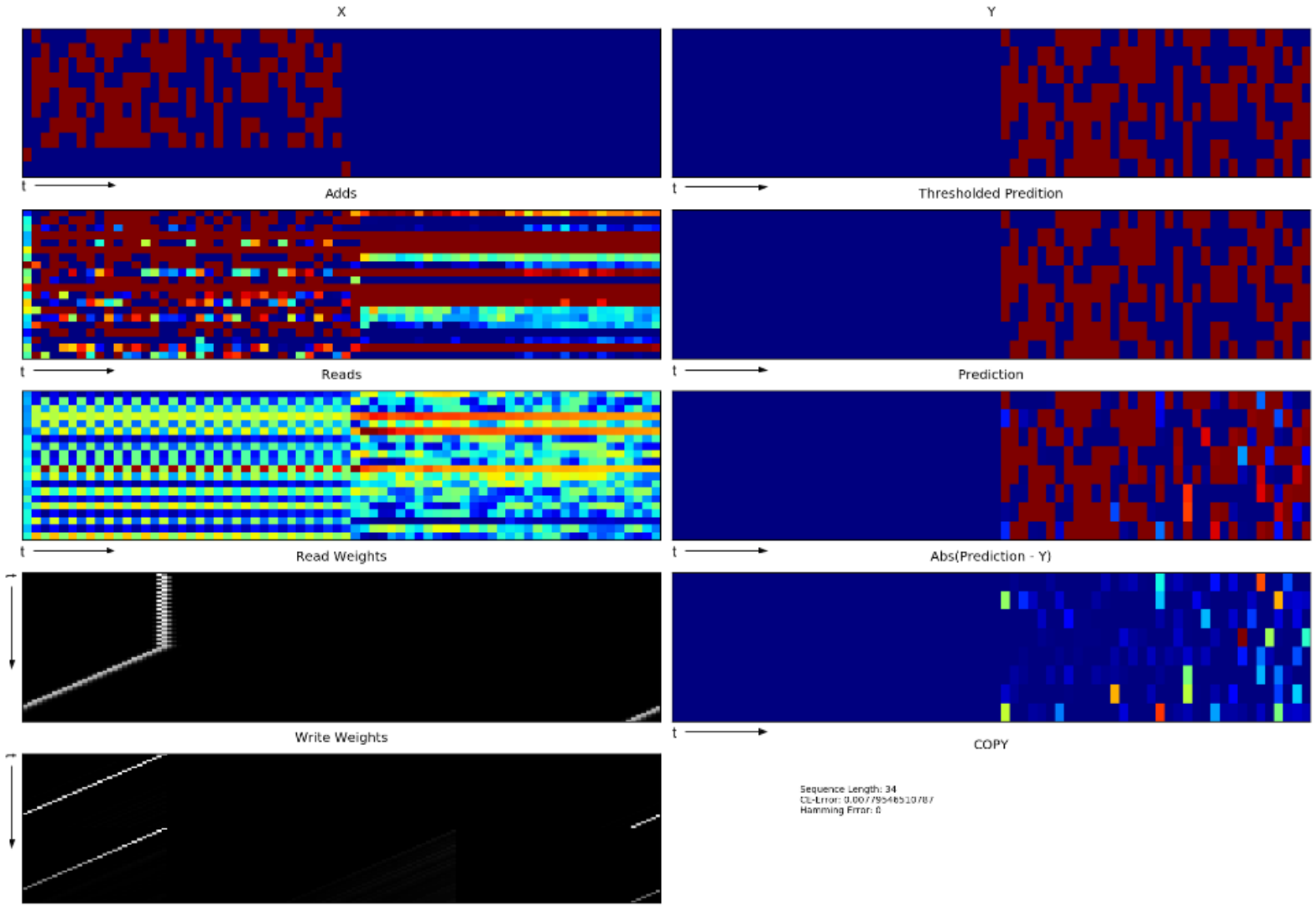
Neural Turing Machine - Model

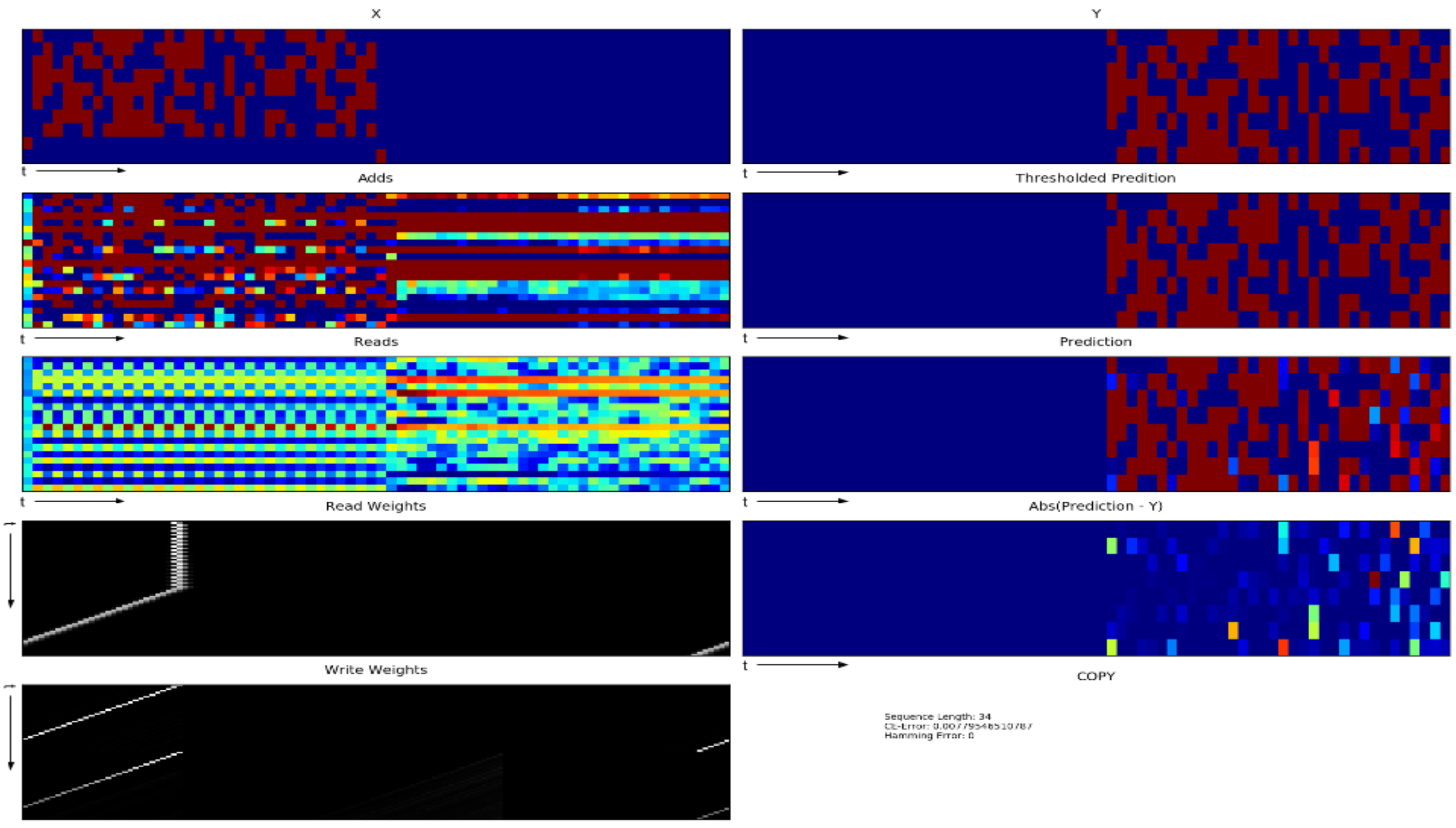
- Now we have a sequence of predictions and a sequence of external outputs (target sequence).
- The error between the prediction sequence and the target sequence is computed and propagated back.
- Then, using these errors the parameters of the model are updated to a value that reduces the prediction error.

Neural Turing Machine - Training

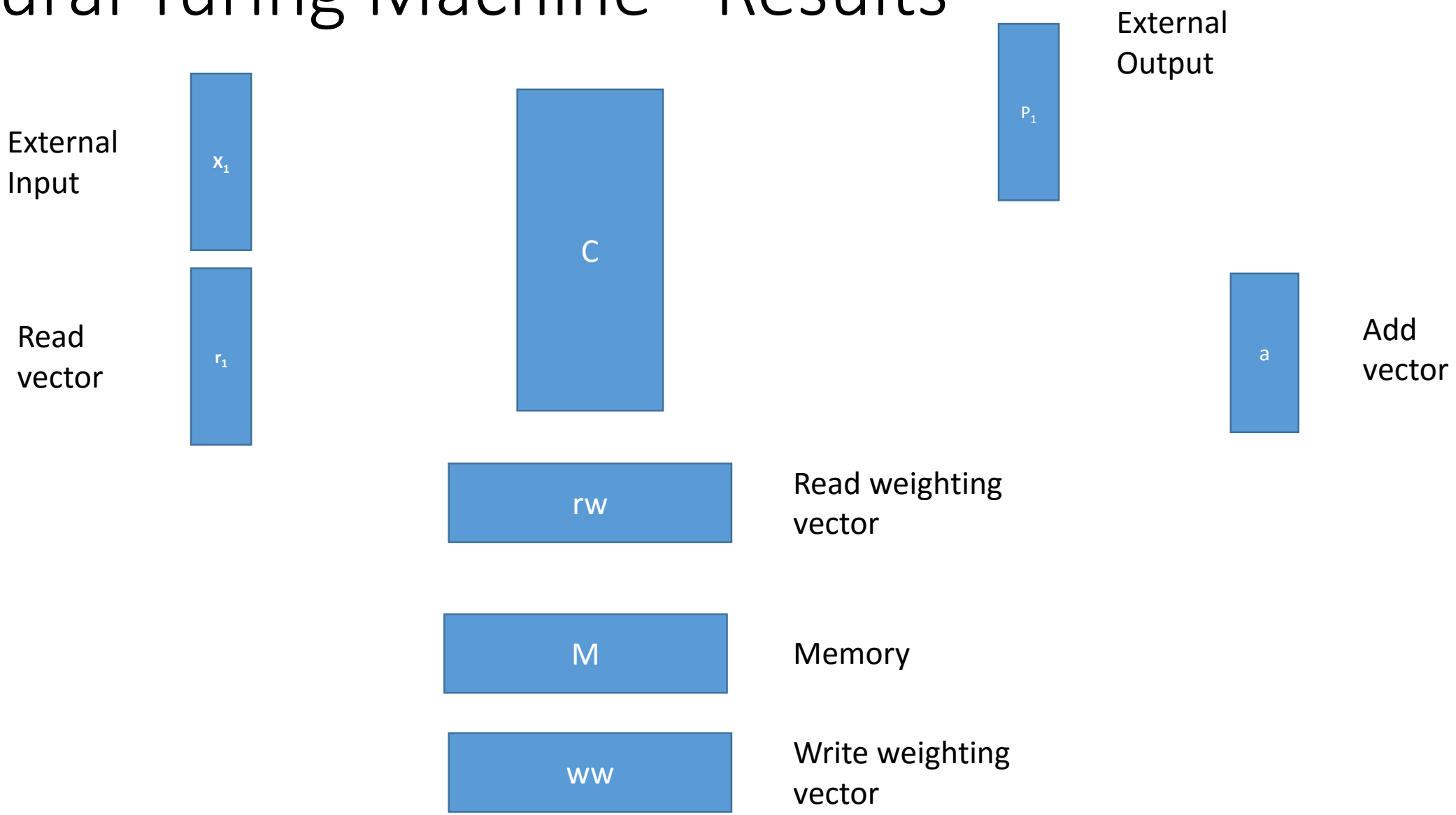
- The model is end to end differentiable with respect to the parameters, so backpropagation algorithm can be used to compute the derivative of loss with respect to the parameters.
- We used RMSProp version of Gradient Descent to update the parameters.
- Previous semester code:
  - CPP, gradient computation from scratch, was not getting good generalization in copy task.
- This semester code:
  - Python-Theano, Theano computes gradients, excellent generalization in copy task.
  - By excellent generalization, we mean that we trained on sequences of length till 20 and got zero hamming distance on sequences of length till 116.

Neural Turing Machine – Results (Copy Task)




X
Y
t
Adds
Thresholded Predition
Reads
Prediction
Read Weights
Abs(Prediction - Y)
Write Weights
COPY
Sequence Length: 34
CE-Error: 0.007795465107817
Hamming Error: 0

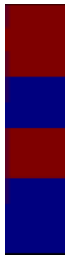
Neural Turing Machine - Results



Neural Turing Machine - Results

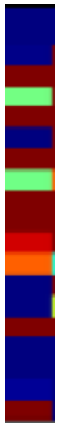
External Input

External Output



C

Read vector



Add vector

Read weighting vector

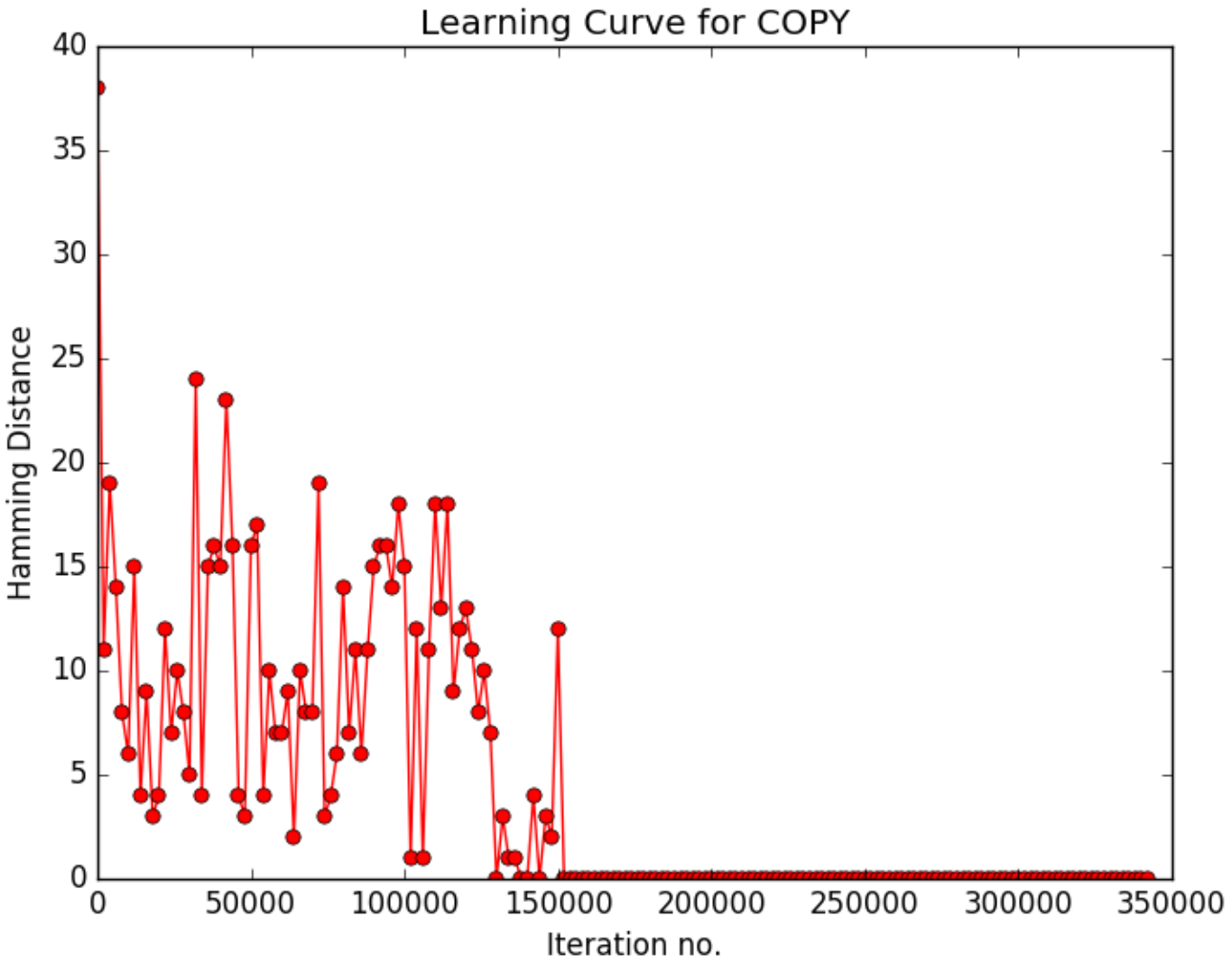
M

Write weighting vector

Neural Turing Machine - Results

- If the NTM has really learnt the transition function of the copy task then
  - Till the second delimiter, the write operation (into memory) should be dominant and hence plotting the add vectors should show some non constant pattern whereas plotting the read vectors should show some constant pattern. Also the write weighting vector should focus on consecutive slots of the memory and the focus by read weighting vector is of no significance till second delimiter.
  - After the second delimiter, the read operation (from memory) becomes dominant and hence plotting the read vectors should show some non constant pattern whereas plotting the add vectors should show some constant pattern. Also the read weighting vector should focus on consecutive slots of the memory in the same order as the write weighting vector before second delimiter and the focus by write weighting vector is of no significance after second delimiter.

Neural Turing Machine – Results (Copy Task)



End to End Memory Networks - Task

- Input:
  - Comprehension:

    1 Mary got the milk there.

    2 John moved to the bedroom.

    3 Sandra went back to the kitchen.

    4 Mary travelled to the hallway.
  - Query:

    5 Where is the milk?
- Output:
  - Answer: hallway
  - Sentences used in computing answer: 1, 4

End to End Memory Networks - Model

- Suppose s_1, s_2, …, s_n be the sentences in our comprehension, q be the query sentence and y be the answer to the query.
- Step 1: Convert s_{ij} (a word) to x_{ij} where x_{ij} = BOW(s_{ij})
- BOW means “Bag Of Words”.
- Let V = size of vocabulary.

End to End Memory Networks - Model

- Step 2: Compute memory vectors m_i from x_i using an embedding, A (dxV).

$$m_i = \sum_j l_j \cdot A x_{ij}$$

where

$$l_{kj} = (1 - j/J) - (k/d)(1 - 2j/J)$$

where J = total number of words in sentence s_i and d is the dimension of the embedding A.

Note: This was something new that I never saw before. The above representation contains the information regarding context as well as order of words (Positional Encoding).

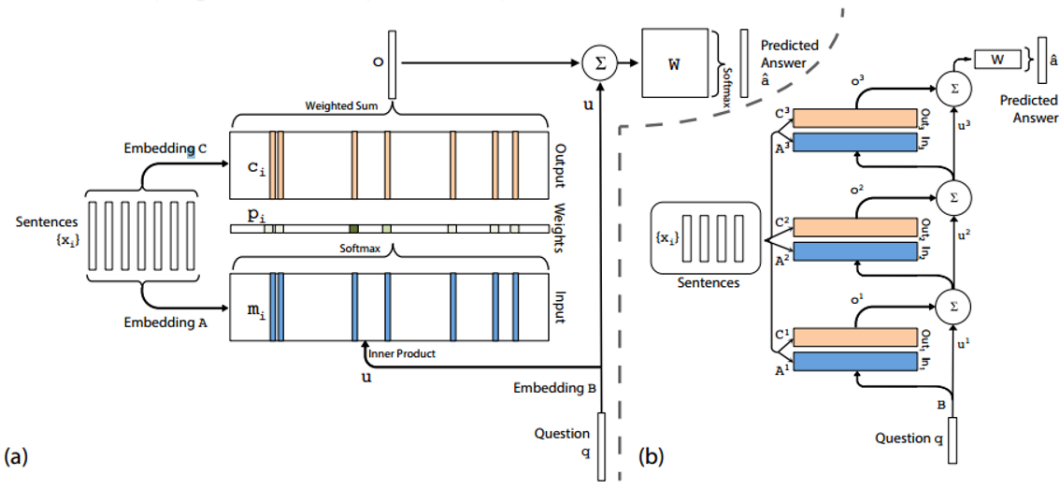


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

- Step 3: In the same manner, output vectors c_i are computed from x_i using an embedding, C and internal representation, u, of query q using an embedding B.
  - Both C and B are of dimension dxV.

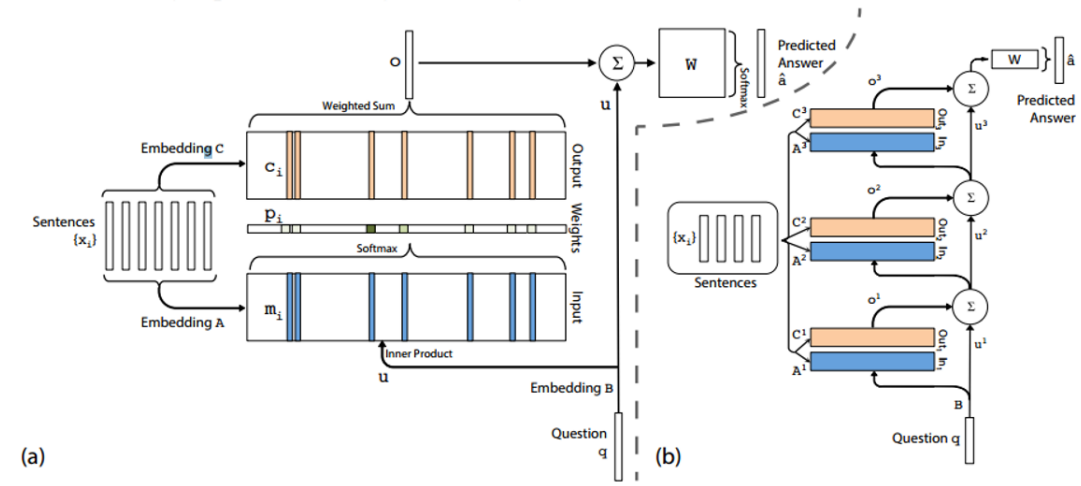


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

- Step 4: Then a weighting, p, over the memory vectors is computed by computing the similarity between the memory vectors and the internal representation of query by softmax.

$$p_i = \text{Softmax}(u^T m_i).$$

  - One can easily observe that, the higher the value of p_i, more relevant is the memory vector m_i in answering the query.

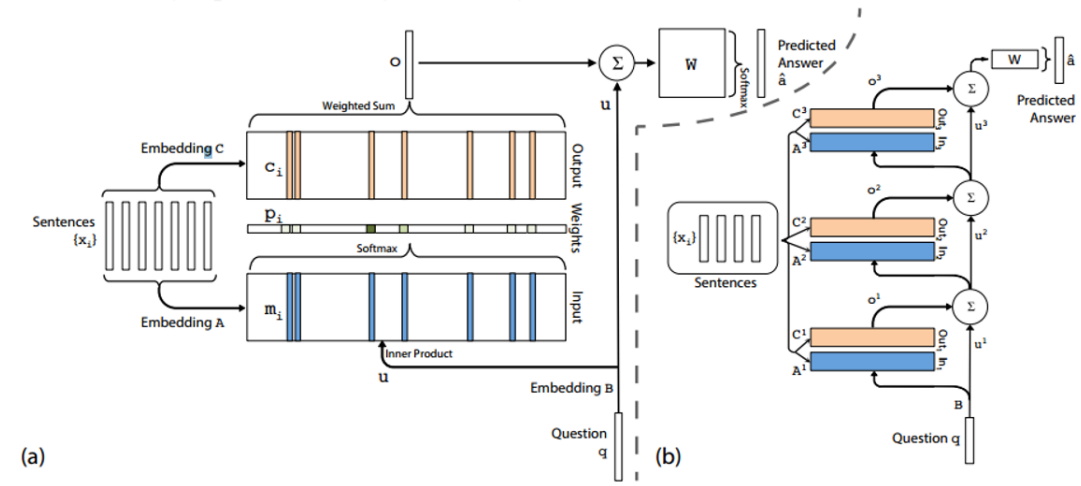


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

- Step 5: An intermediate response, o, is generated by weighted sum of the output vectors c_i with p_i as weight corresponding to c_i.

$$o = \sum_i p_i c_i.$$

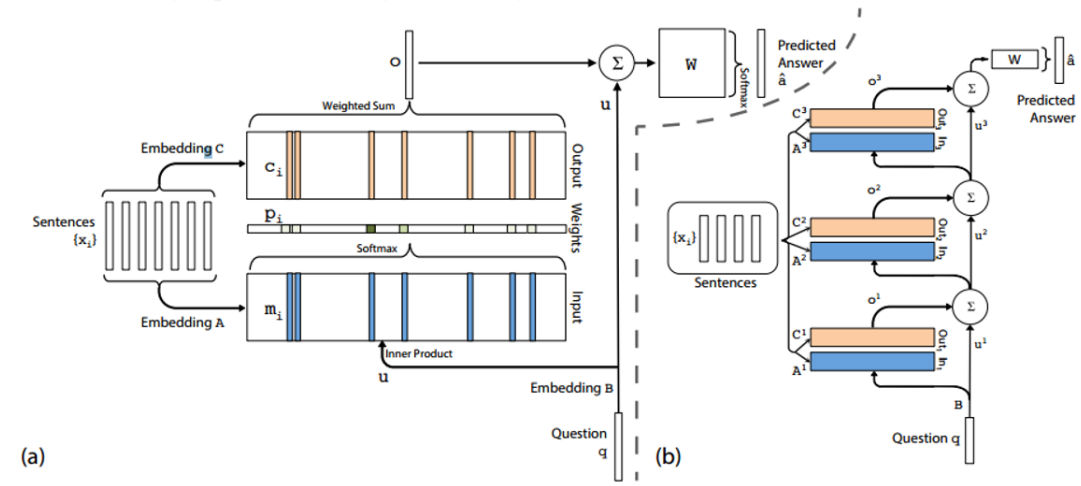


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

- Step 6: This intermediate response is o, is summed with the internal representation of query u.

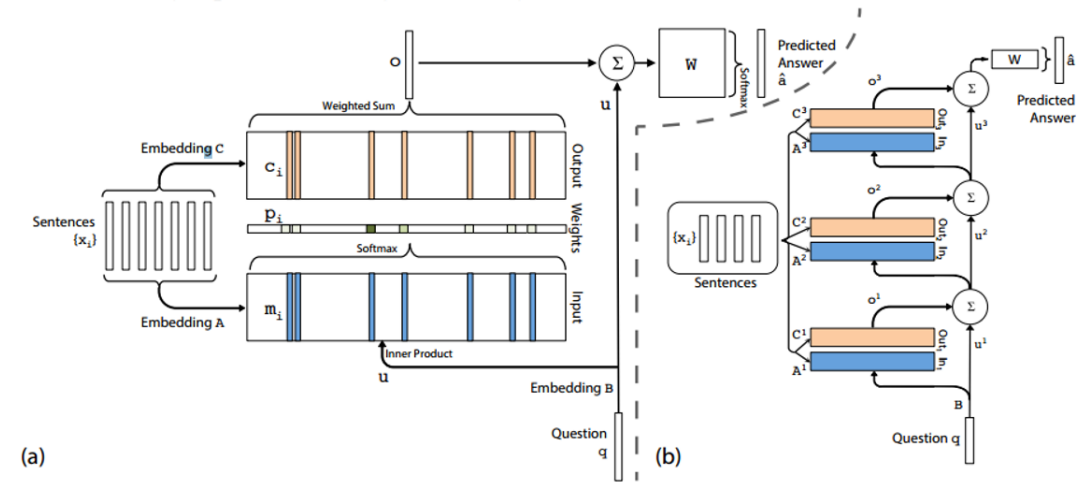


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

- Step 7: The summed vector is transformed using a decoding matrix W of dimension Vxd and softmax-ed to a V dimensional final response â.

$$\hat{a} = \text{Softmax}(W(o + u))$$

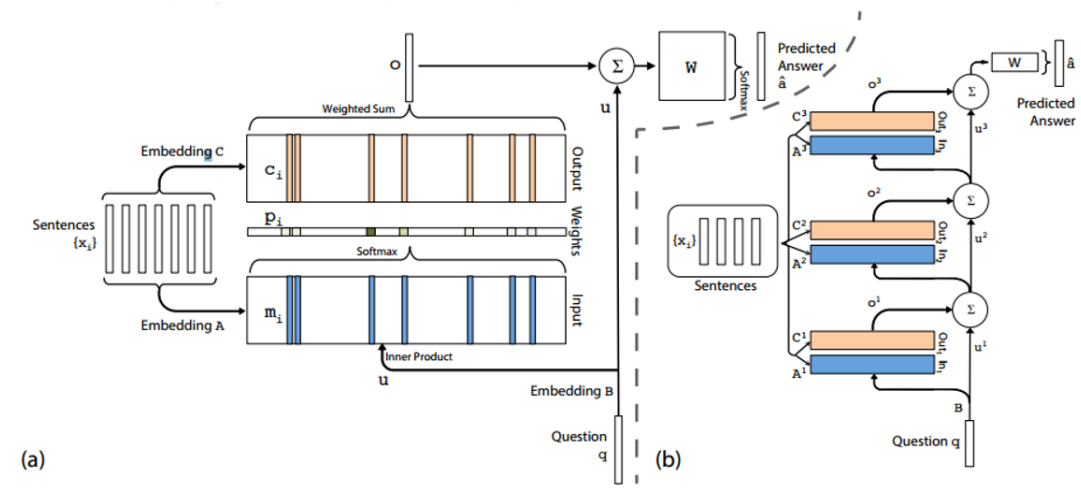


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

- Loss: Binary cross-entropy error between â and BOW(y) where y is the target one-word answer.

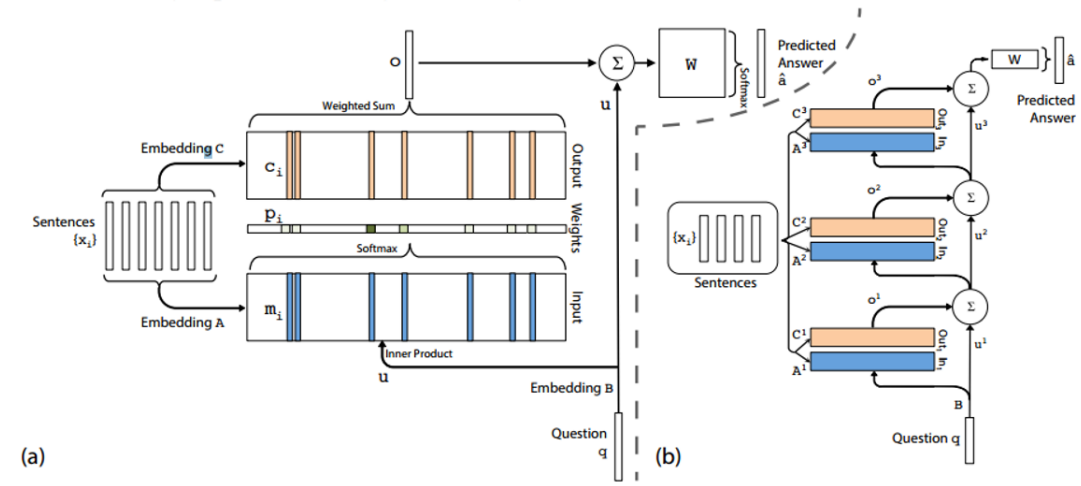


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

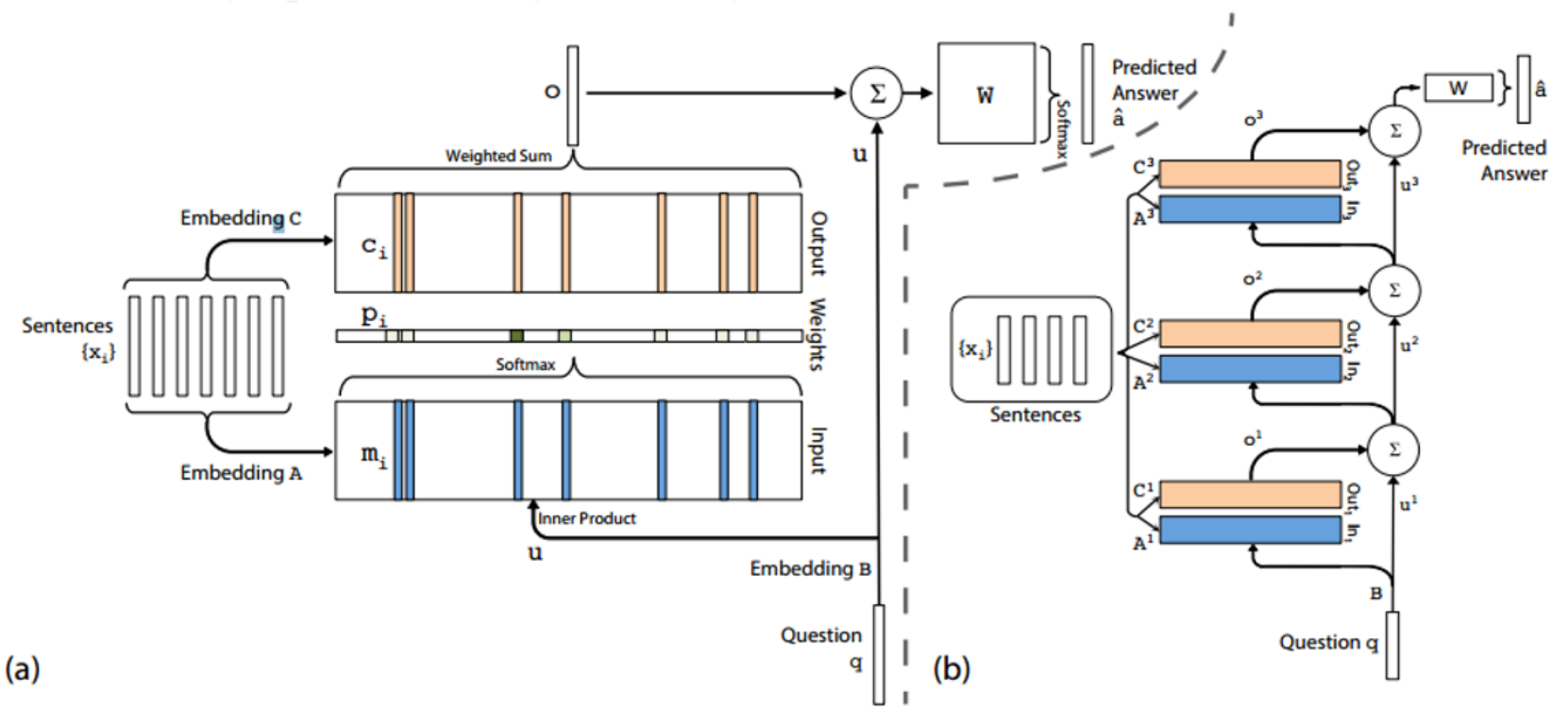


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Model

- Observe that the parameter space is too high, B, W, A_i, C_i, i=1..number-of-hops/layers.
- To tackle this, two types of parameter tying is used:
  - RNN like: $A_1 = A_2 = \ldots = A_n = A$ and $C_1 = C_2 = \ldots = C_n$
  - Adjacent: $C_{k+1} = A_k$, $B = A_1$ and $W^T = C_N$ (We used this in our model)

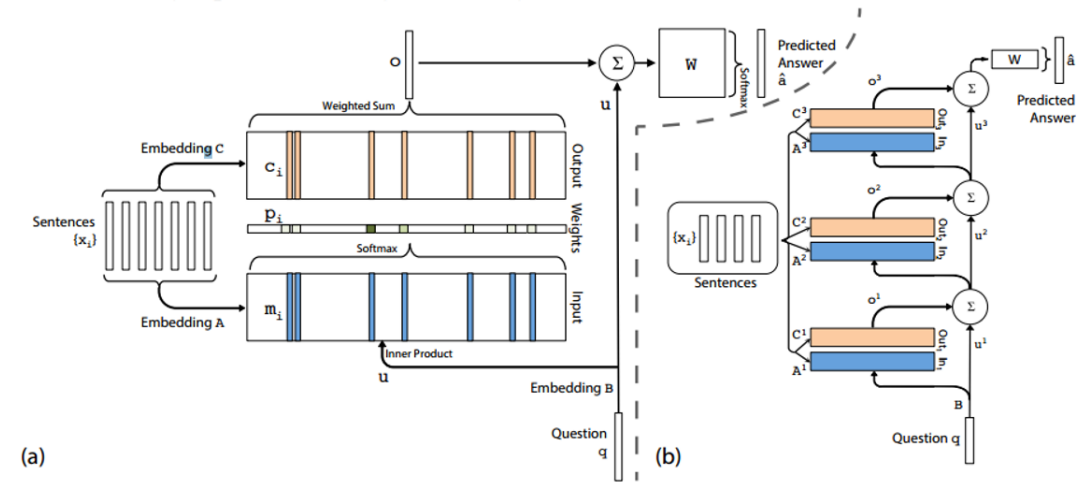


Figure 1: [5](a) Single layer of End to End Memory Networks (b) 3-layered End to End Memory Networks

End to End Memory Networks - Training

- Since the model is end to end differentiable with respect to the parameters, one can easily use backpropagation algorithm for computing the derivative of loss with respect to the parameters.
- Stochastic Gradient Descent is used to update the parameters.

End to End Memory Networks - Results

- We used Babi Project Dataset from FB-AI research.
- To test whether, the network is able to reproduce the results in toy tasks, we tested with 20 toy tasks
- With Single supporting fact toy task we got following results:

	C1	C2	C3	C4	C5	C6
C1	148	0	0	0	0	0
C2	0	168	0	2	0	0
C3	1	0	180	1	1	1
C4	0	0	0	153	0	0
C5	0	0	0	0	156	0
C6	1	0	0	0	0	180

Confusion Matrix

Precision, recall and f1-score calculated using this matrix are consistent with the results shown by FB-AI.

Single Supporting Fact
Comprehension:
1 John travelled to the hallway.
2 Mary journeyed to the bathroom.
Query:
3 Where is John?
Answer:
hallway
Sentence used in inference: 1

Similarity between Neural Turing Machine and End to End Memory Network

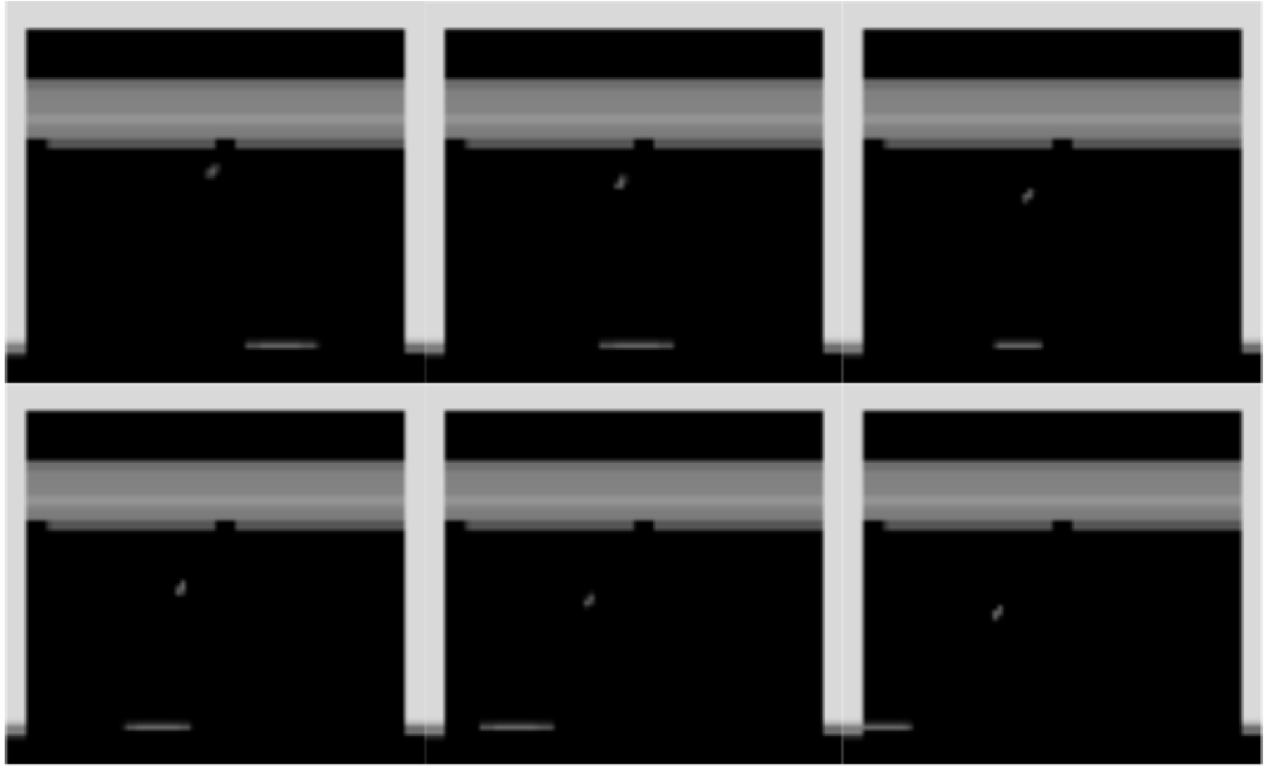
- The content addressing module of NTM is similar to the similarity finding between memory vectors and internal representation of query.
- The location based addressing module of NTM seems to be implementing the multiple hops concept of end to end memory networks.

Game Playing Agent with RAM – Aim

- To build a model
  - that can learn to play any game from the sensory input only
  - in an optimal manner (better than humans)
- Using
  - Reinforcement Learning
    - Markov Decision Process
    - Q-Learning
  - Deep Learning
    - Convolution Neural Networks

Game Playing Agent with RAM – Problem with state-of-the-art

- Problem with current state of the art?
- Consider the game *Breakout* and the following sequence of frames observed by an agent.



Game Playing Agent with RAM – Problem with state-of-the-art

- State-of-the-art agent (based on deep learning) moves to the left.
- But why?
- A human agent might answer that since the ball is moving towards left the action taken is to move left.
- This is the part that the state-of-the-art agent doesn't answer. It has just learnt some set of parameters that once fitted in the neural network predicts optimal action to take.
- But hopefully the RAM-based agent will be able to reason that ”since the ball is moving towards left, I should also move towards left to prevent death”.

Game Playing Agent with RAM – Resolution with RAM (Ideas)

- Weighting over memory slots as well as over the information (bits) in a slot itself.
  - Hope is that: the ball and the slider in the frame will be focused by modified NTM.
  - That'll provide reasoning to some extent that the decision (choice of action) by modified NTM is taken based on the position of ball and the slider.
- Increasing the number of allowed shifts in NTM to be equal to the number of memory slots so that the focus can shift from one memory slot to arbitrarily any slot.