Memory plus Meta-Learning

Deepest Season 6

August 31, 2019

정재원

Outline

1. Introduction to Meta-Learning

2. Neural Networks with Memory

   - Neural Turing Machine (NTM) *(arXiv 2014)*

   - Differentiable Neural Dictionary (DND) *(ICML 2017)*

3. Memory + Meta-Learning

   - One-Shot Learning with Memory Augmented Neural Networks *(arXiv 2016)*

   - Been There, Done That: Meta-Learning with Episodic Recall *(ICML 2018)*

   - Rapid Adaptation with Conditionally Shifted Neurons *(ICML 2018)*

Outline

~~1. Introduction to Meta-Learning~~

2. Neural Networks with Memory

    - Neural Turing Machine (NTM) *(arXiv 2014)*

    - Differentiable Neural Dictionary (DND) *(ICML 2017)*

3. Memory + Meta-Learning

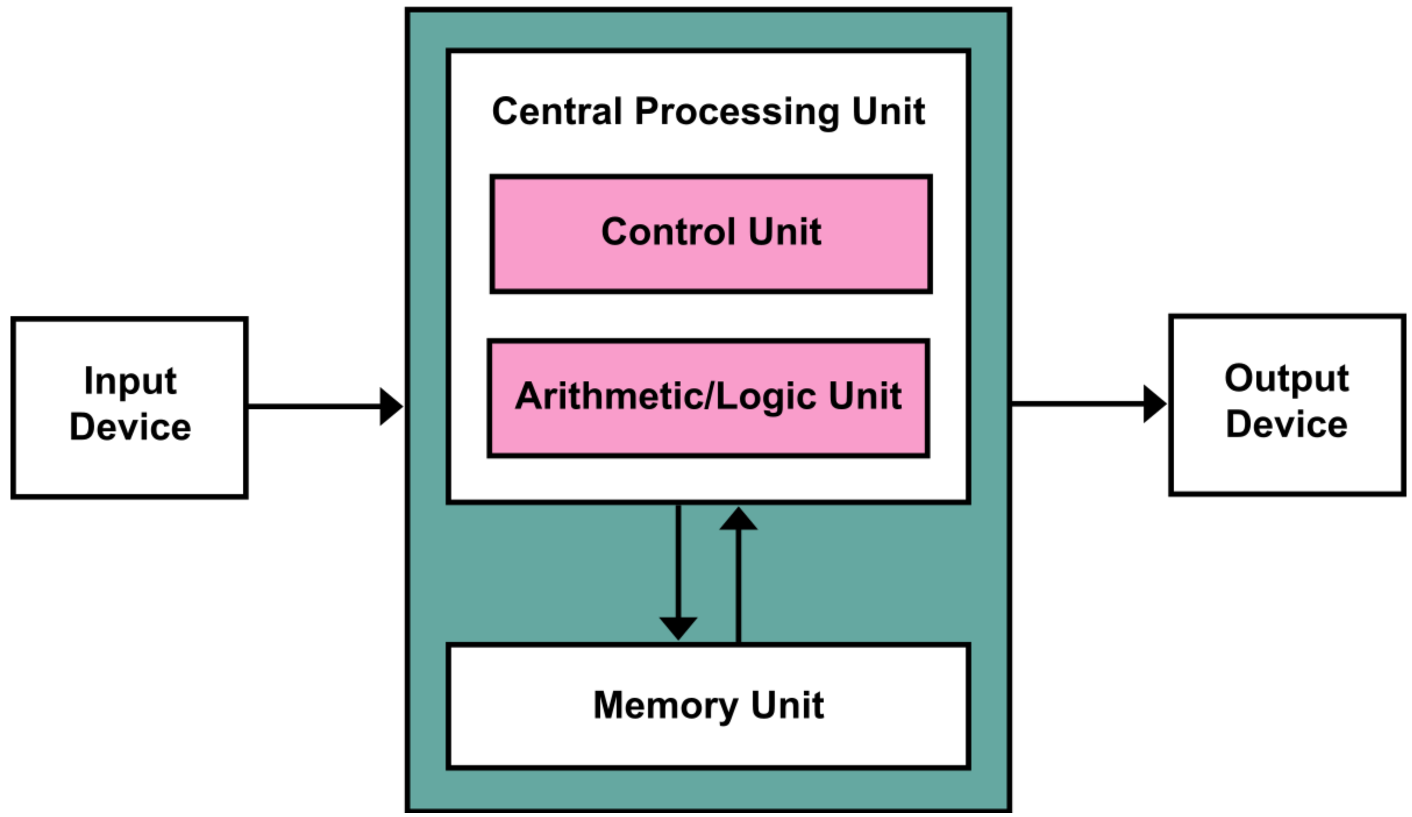
    - One-Shot Learning with Memory Augmented Neural Networks *(arXiv 2016)*

    - Been There, Done That: Meta-Learning with Episodic Recall *(ICML 2018)*

    - Rapid Adaptation with Conditionally Shifted Neurons *(ICML 2018)*

Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*



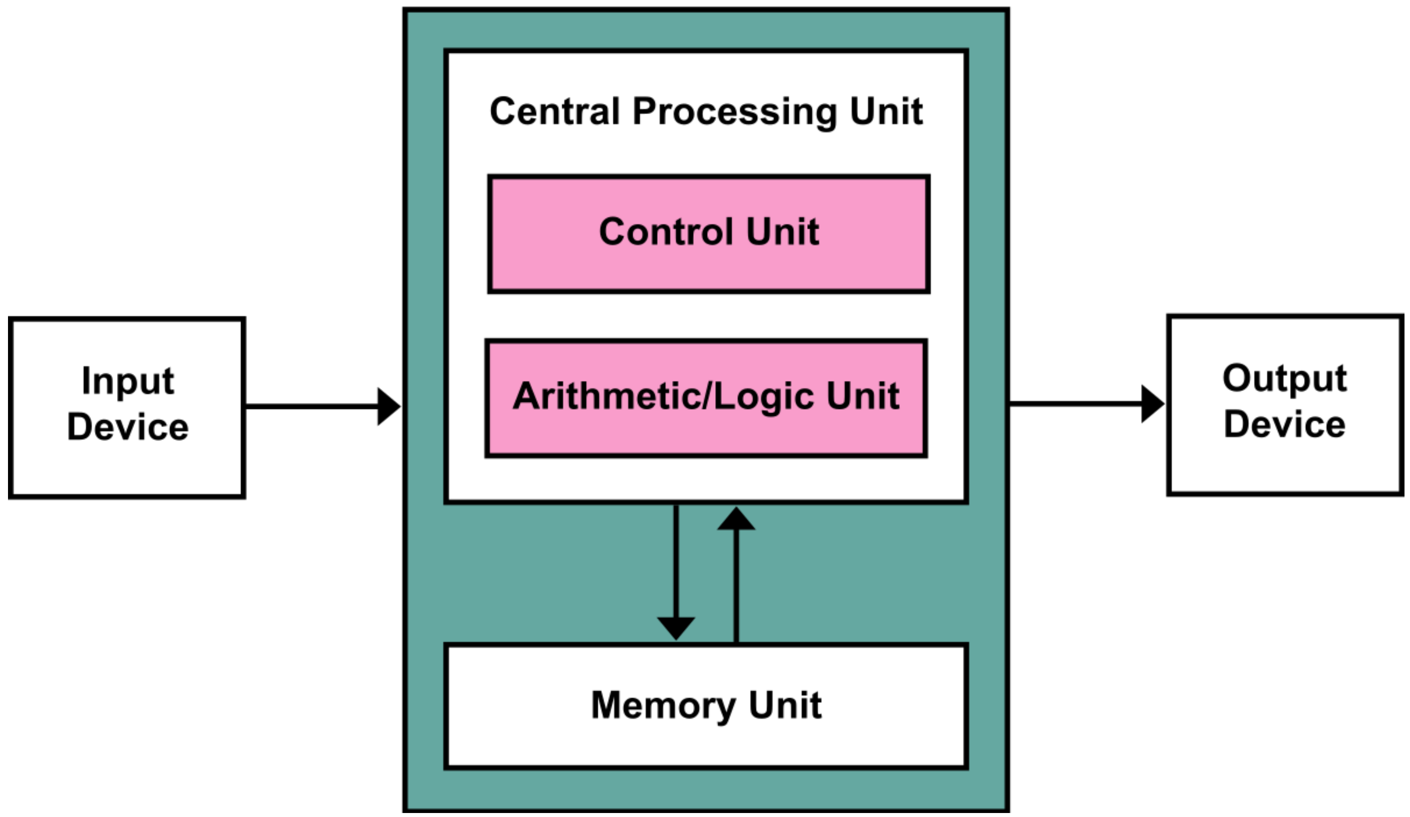
Von Neumann Architecture

1. **Load** program and data from memory unit
2. Perform arithmetic and logical operations
3. **Store** results back into memory unit

https://en.wikipedia.org/wiki/Von_Neumann_architecture

Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*



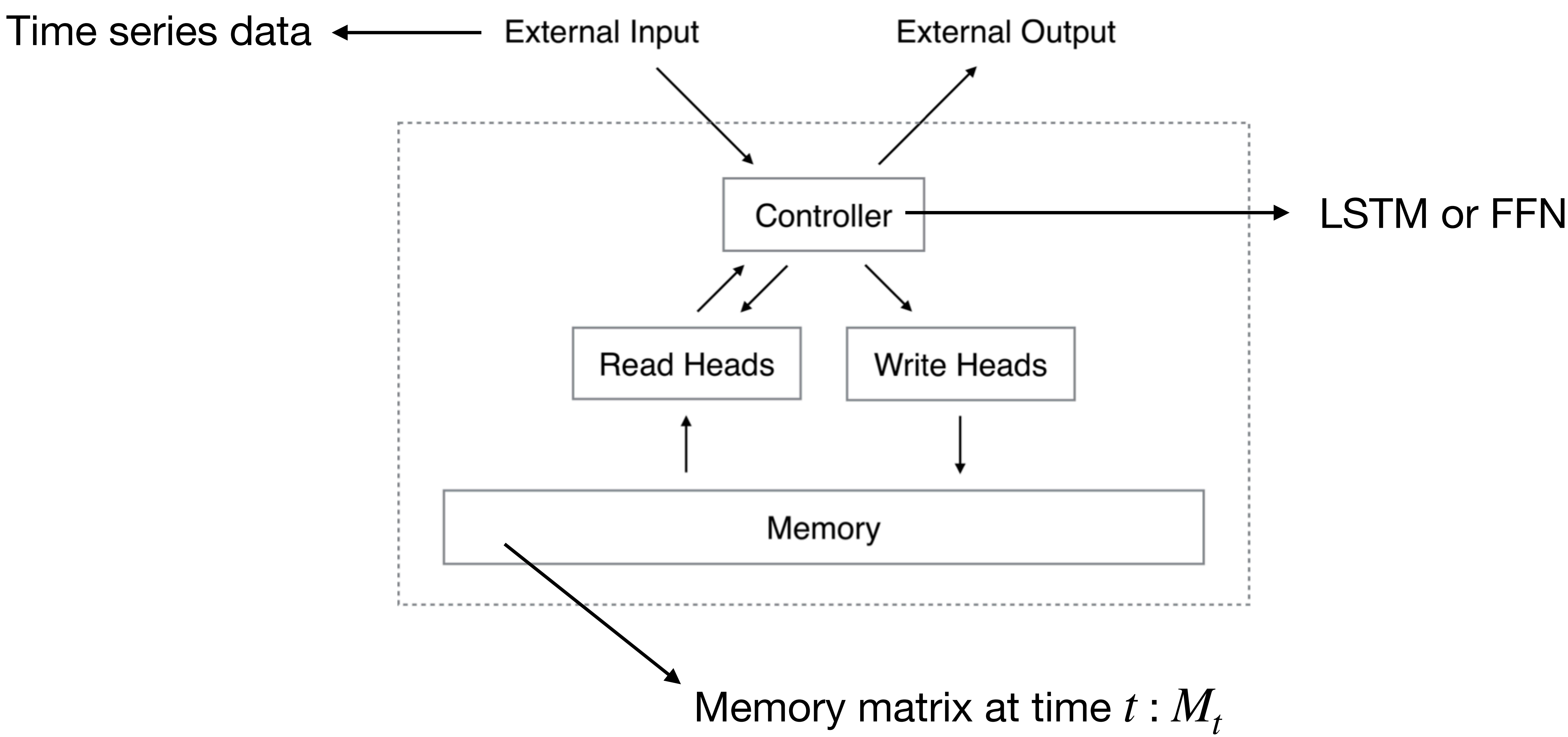
Three **fundamental mechanisms** of computer programs:

1. Elementary operations (e.g. arithmetic operations)
2. Logical flow control (branching)
3. External memory

https://en.wikipedia.org/wiki/Von_Neumann_architecture

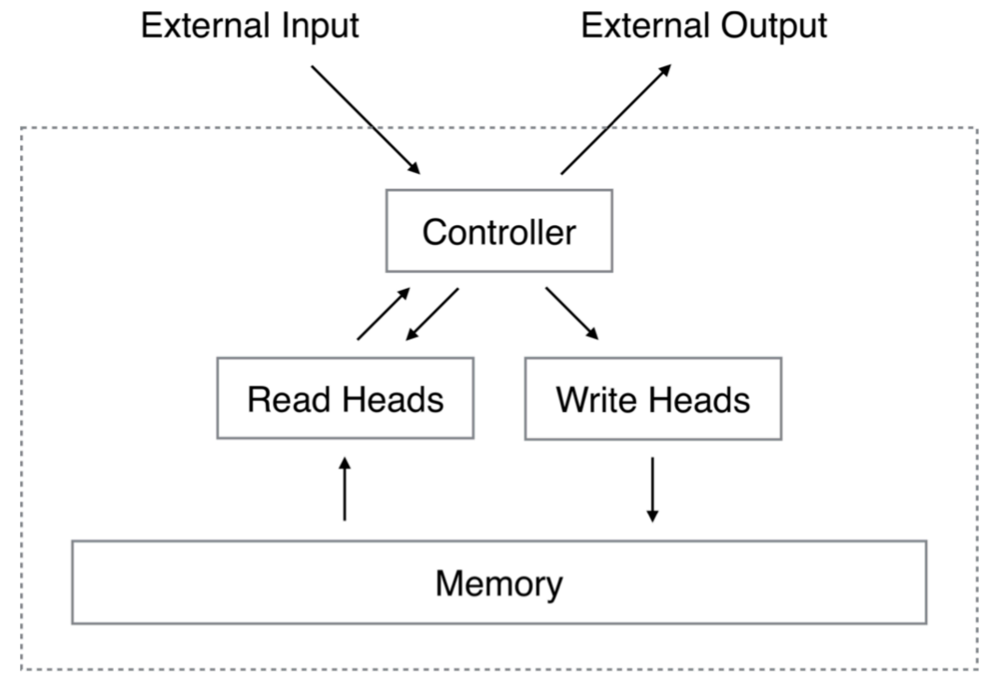
Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*



Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*



Reading from memory

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

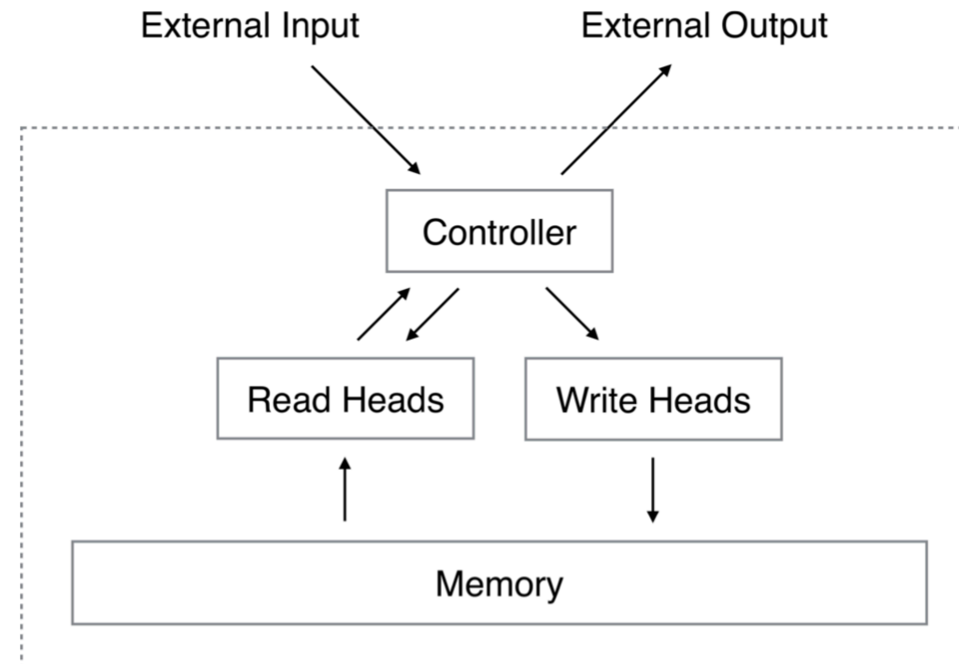
Writing to memory

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

erase add

Neural Networks with Memory

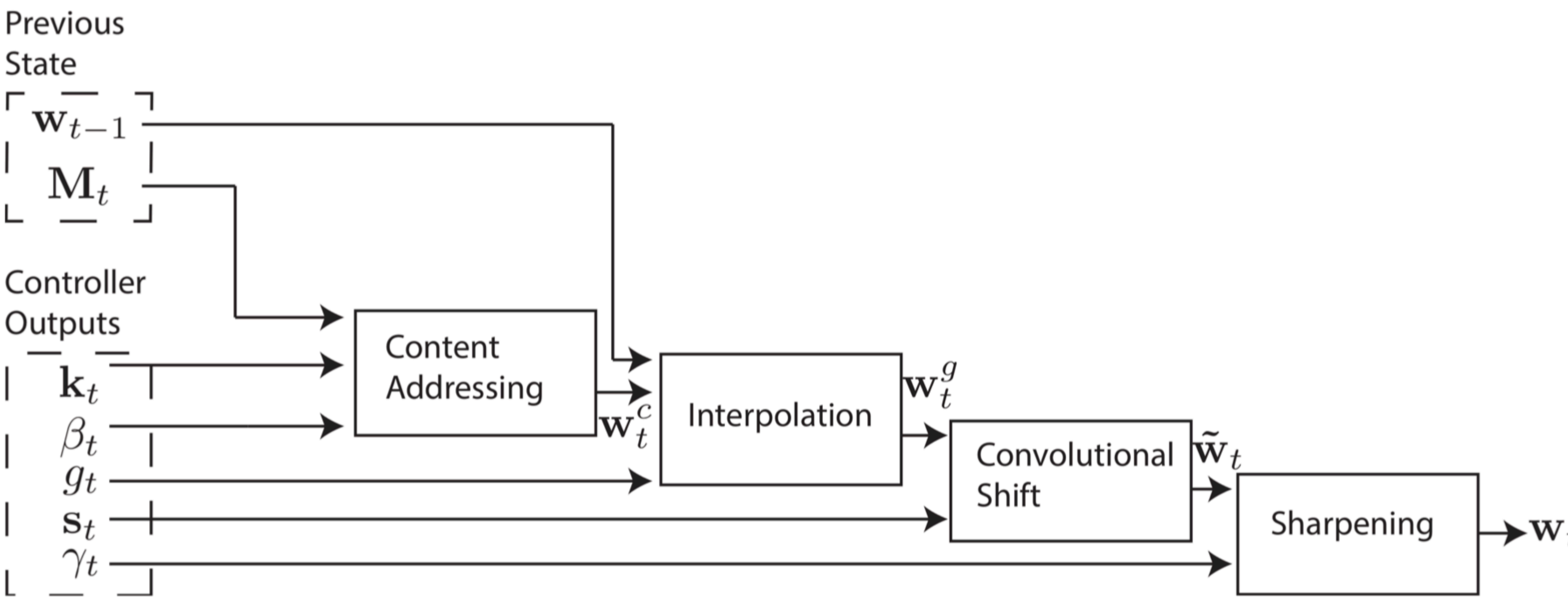
- Neural Turing Machine (NTM) *(arXiv 2014)*



Addressing

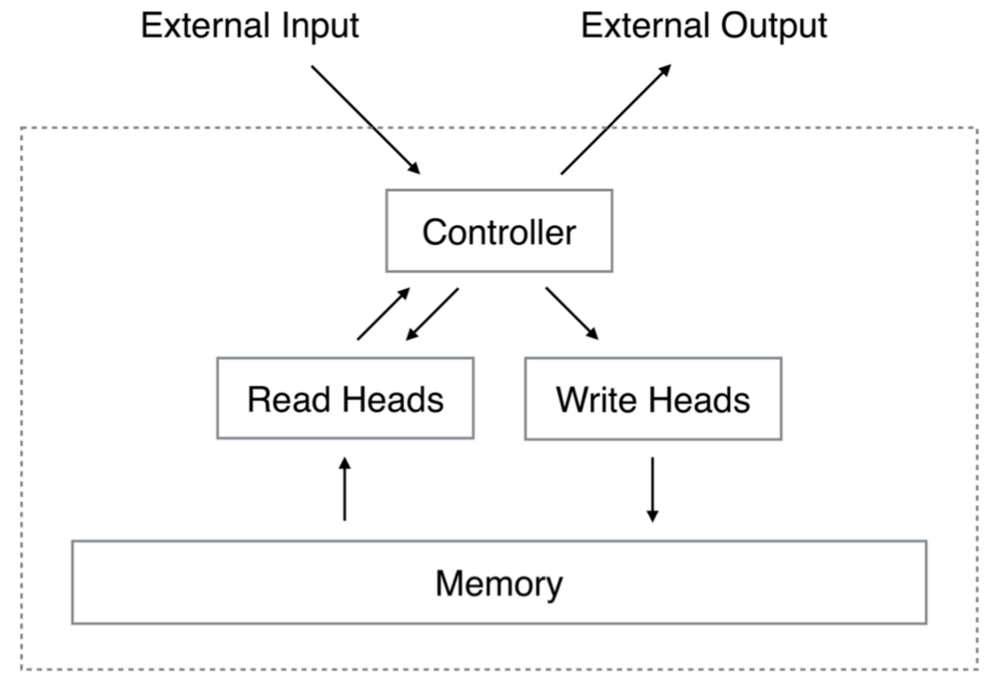
Content-based + Location-based

Identical procedure for both heads



Neural Networks with Memory

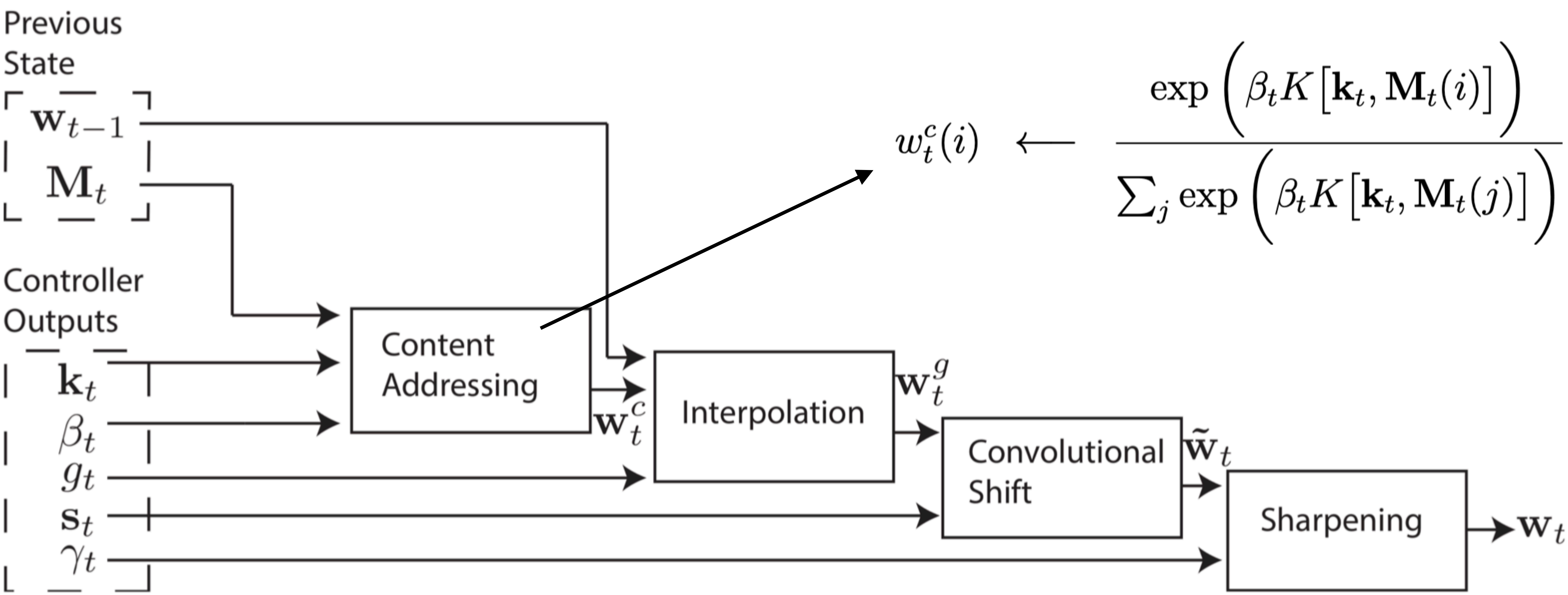
- Neural Turing Machine (NTM) *(arXiv 2014)*



Addressing

Content-based + Location-based

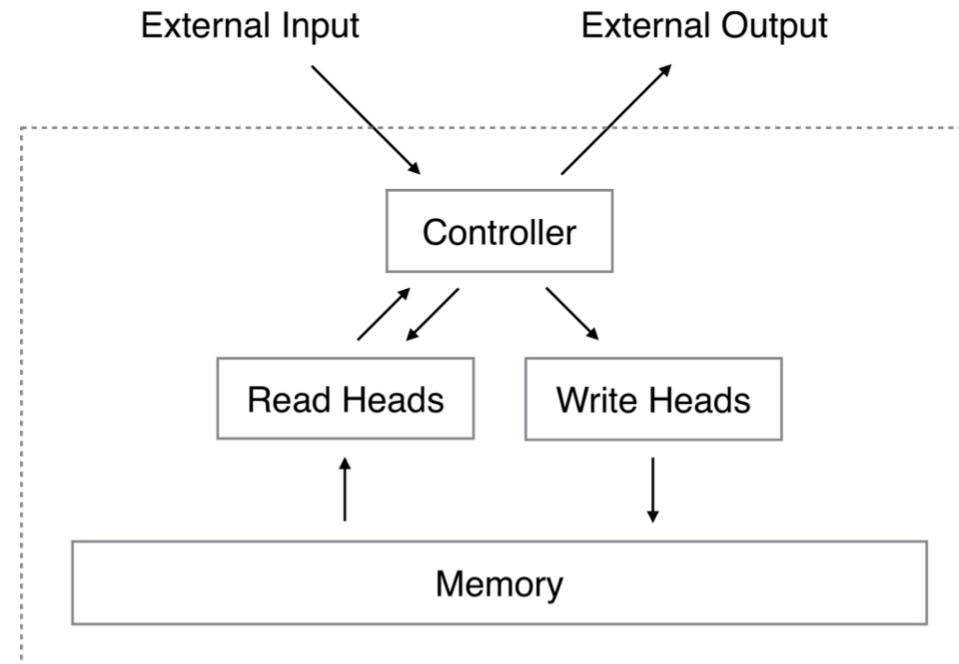
Identical procedure for both heads



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

Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*

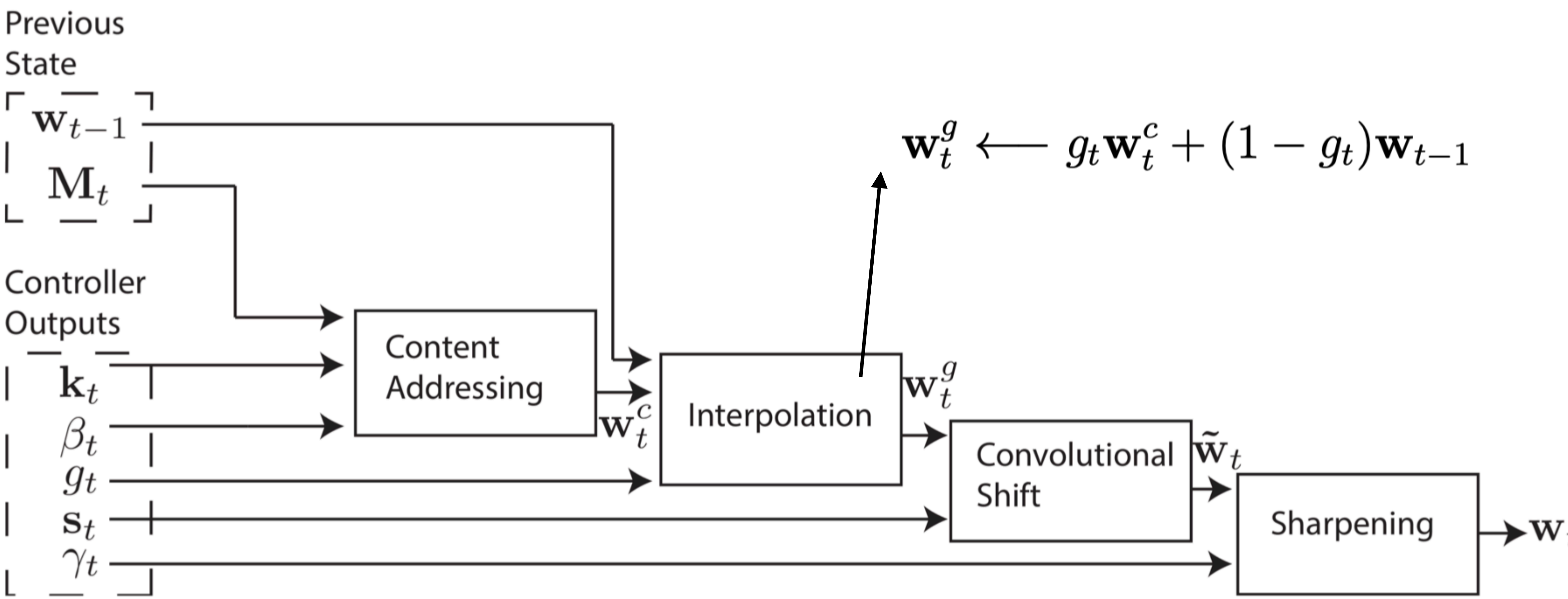


Addressing

Content-based + Location-based

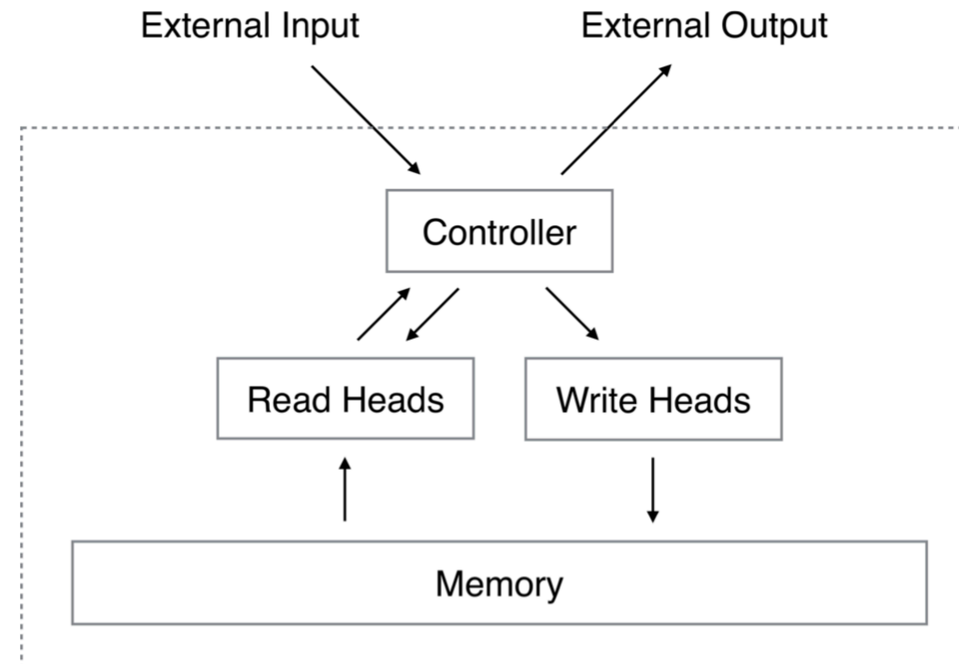
Identical procedure for both heads

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



Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*

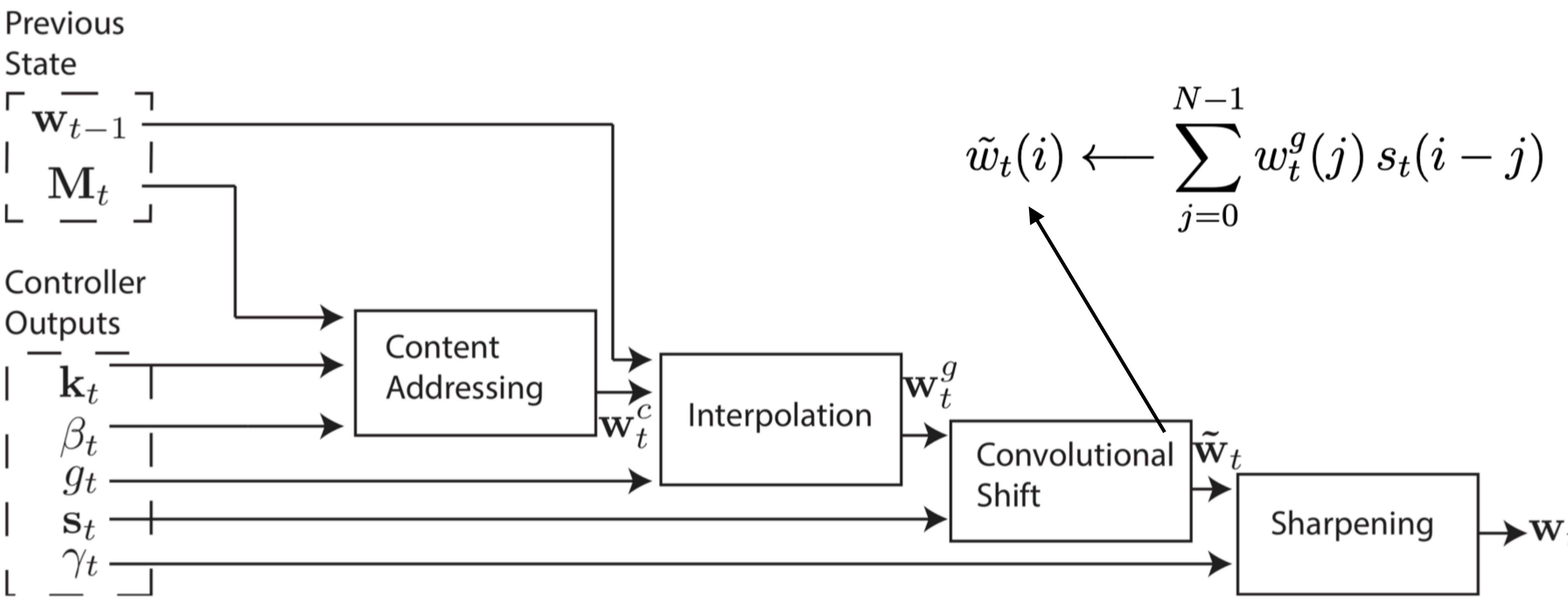


Addressing

Content-based + Location-based

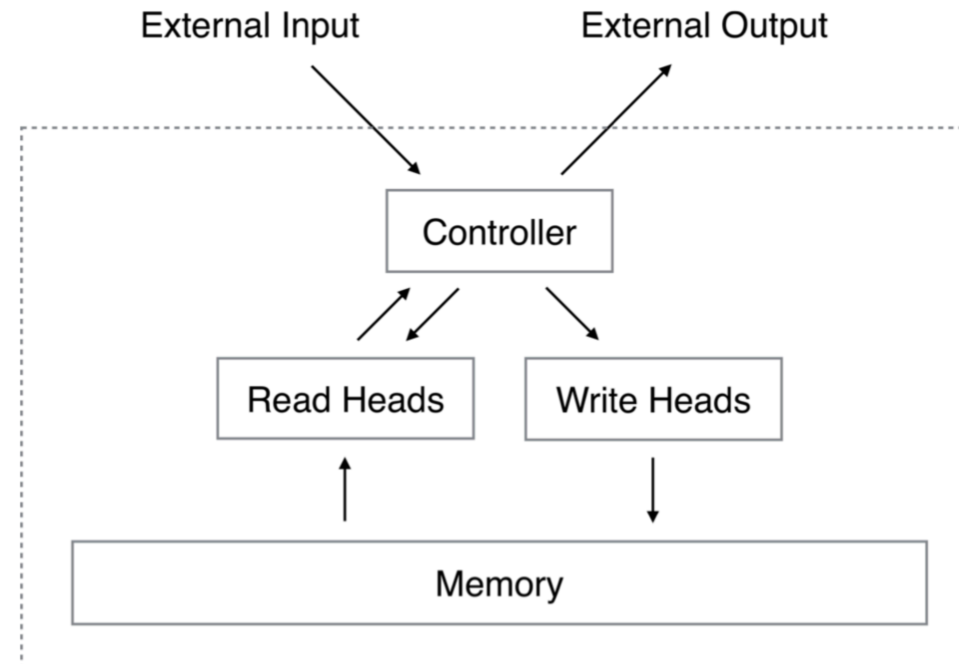
Identical procedure for both heads

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



Neural Networks with Memory

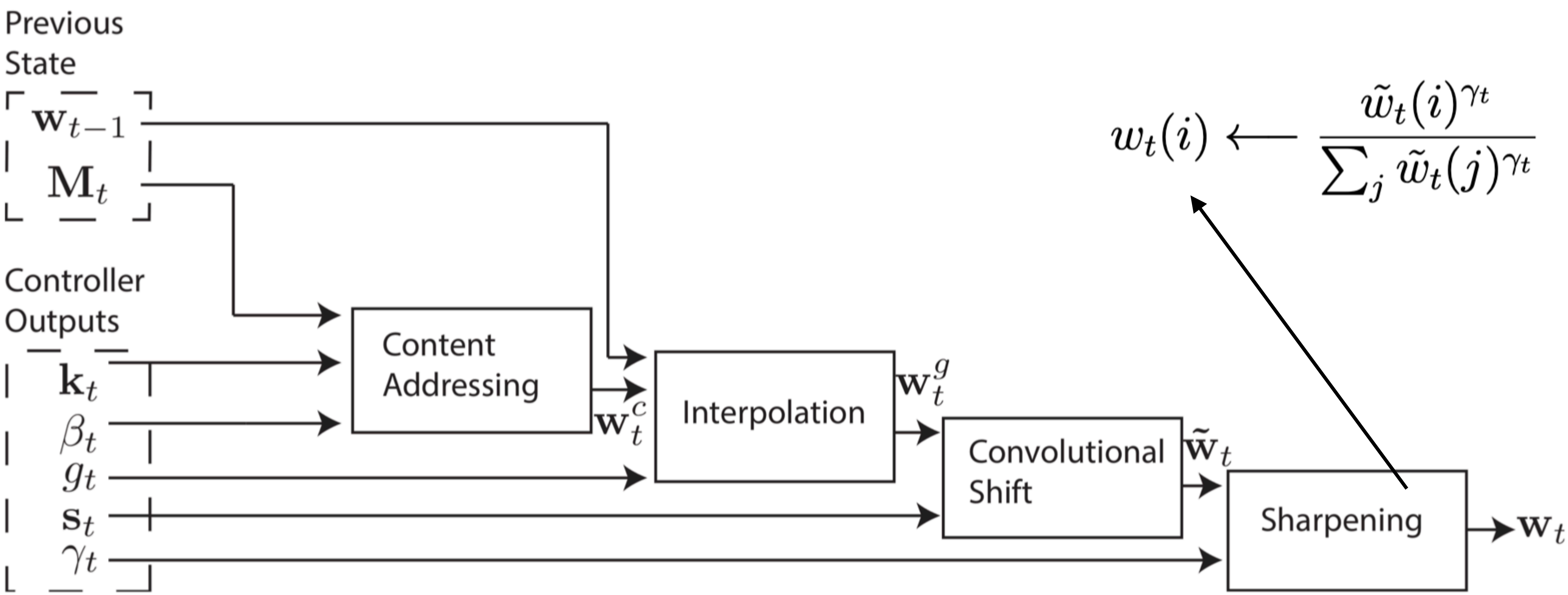
- Neural Turing Machine (NTM) *(arXiv 2014)*



Addressing

Content-based + Location-based

Identical procedure for both heads



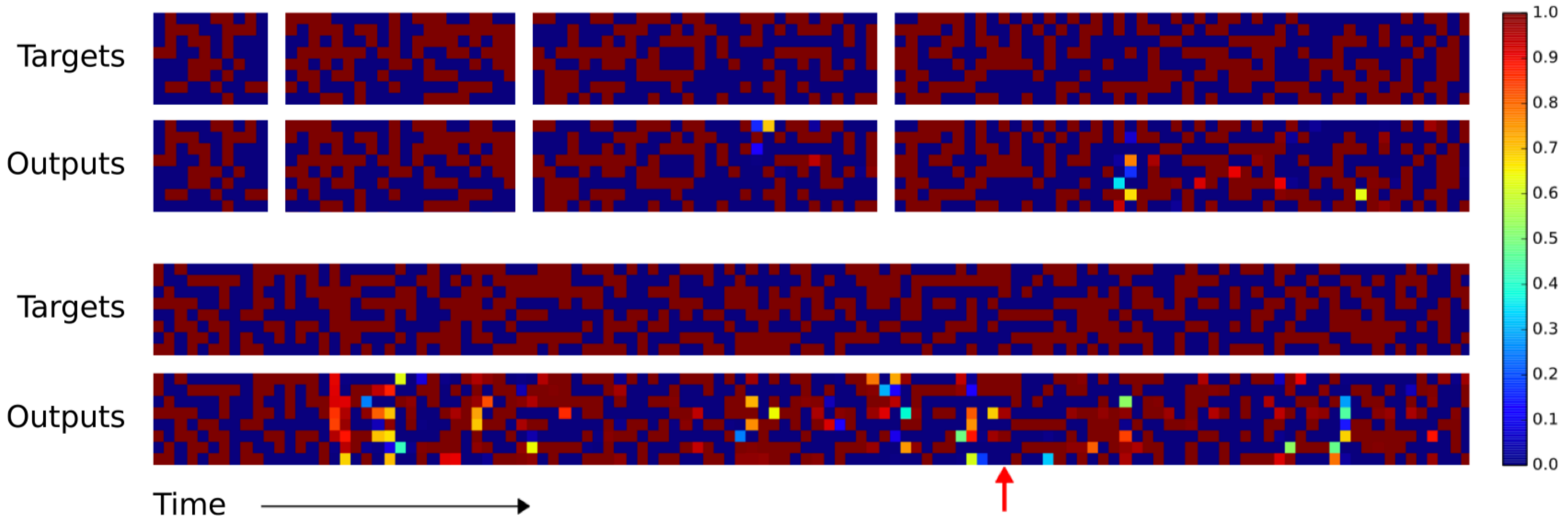
$$w_t(i) \longleftarrow \frac{\tilde{w}_t(i)^{\gamma_t}}{\sum_j \tilde{w}_t(j)^{\gamma_t}}$$

Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*

Experiments

copy

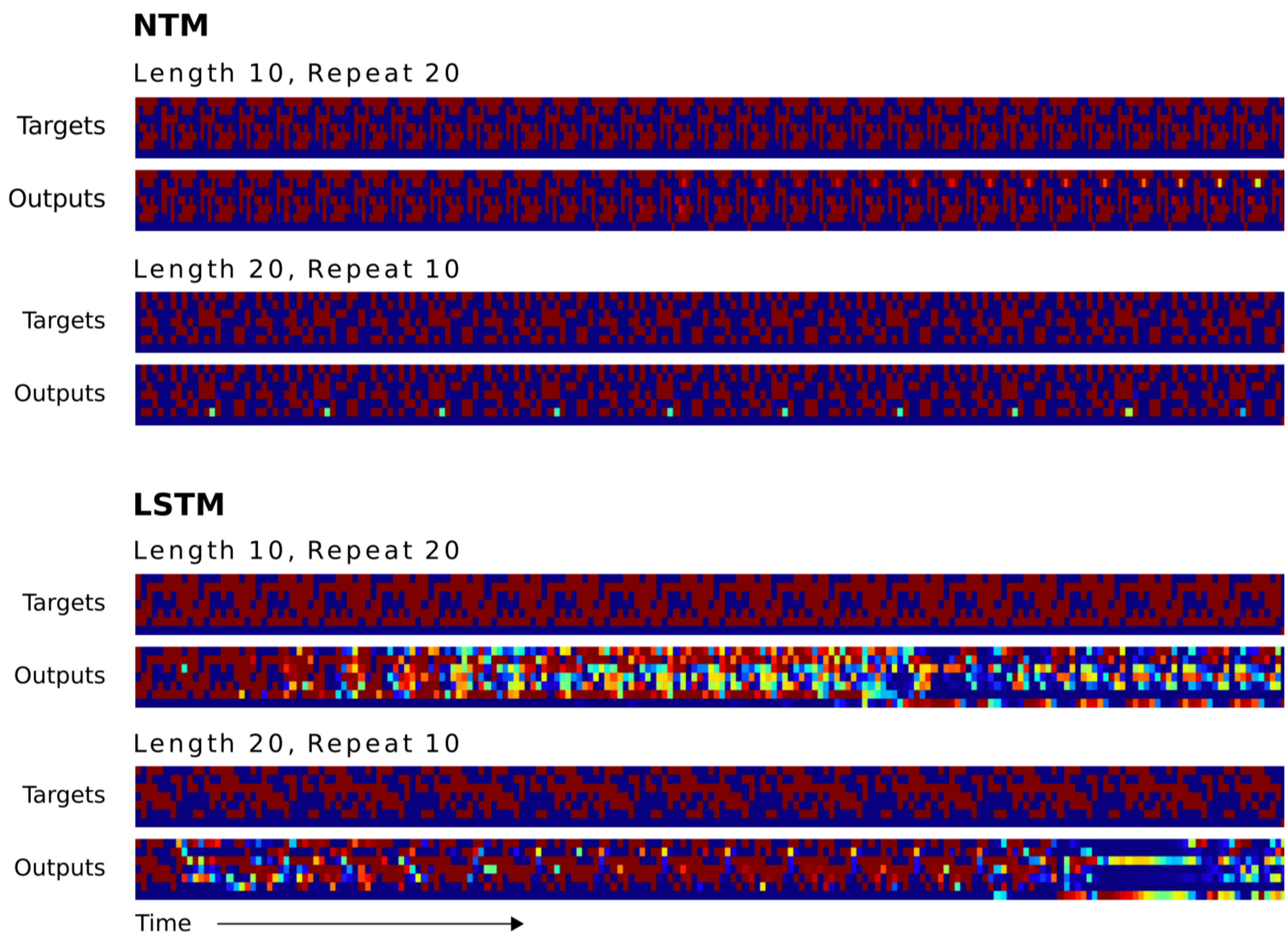


Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*

Experiments

repeat copy

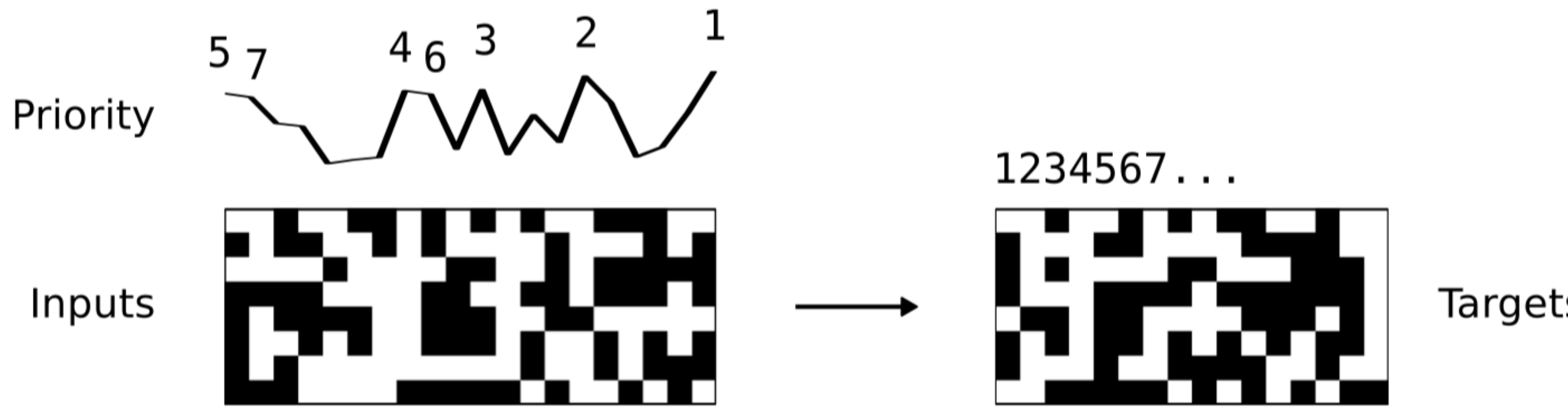


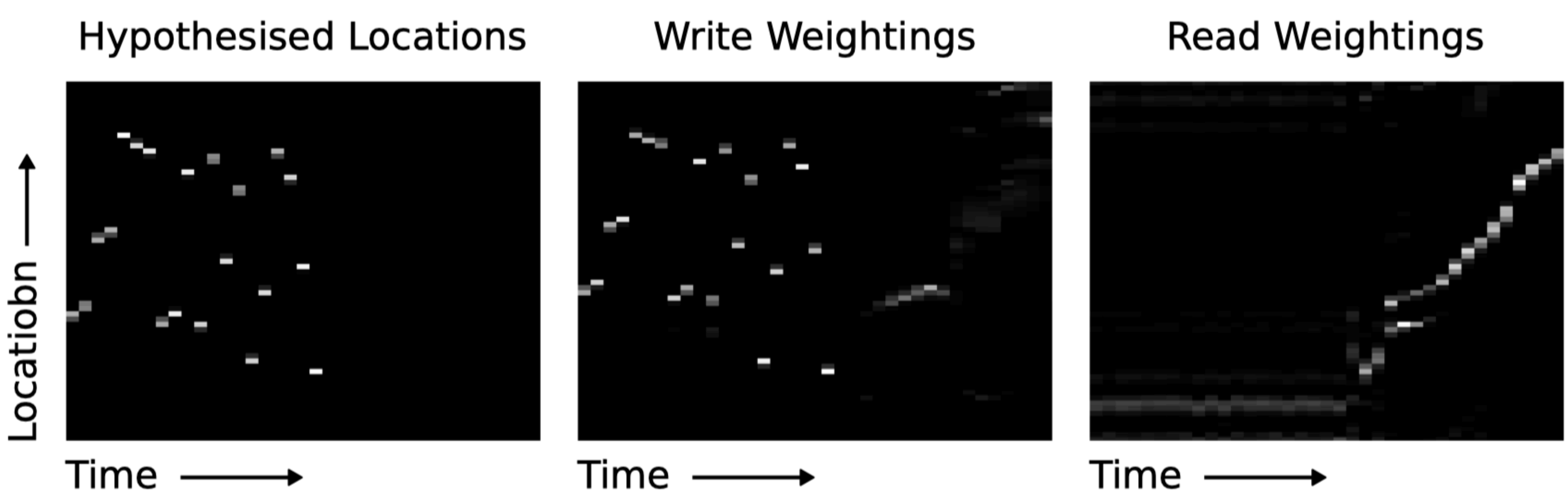
Neural Networks with Memory

- Neural Turing Machine (NTM) *(arXiv 2014)*

Experiments

priority sort





Neural Networks with Memory

- Neural Episodic Control (NEC) *(ICML 2017)*

A DQN agent consisted of:

1. Differentiable Neural Dictionary (DND):

   A key-value based memory module $M_a = (K_a, V_a)$

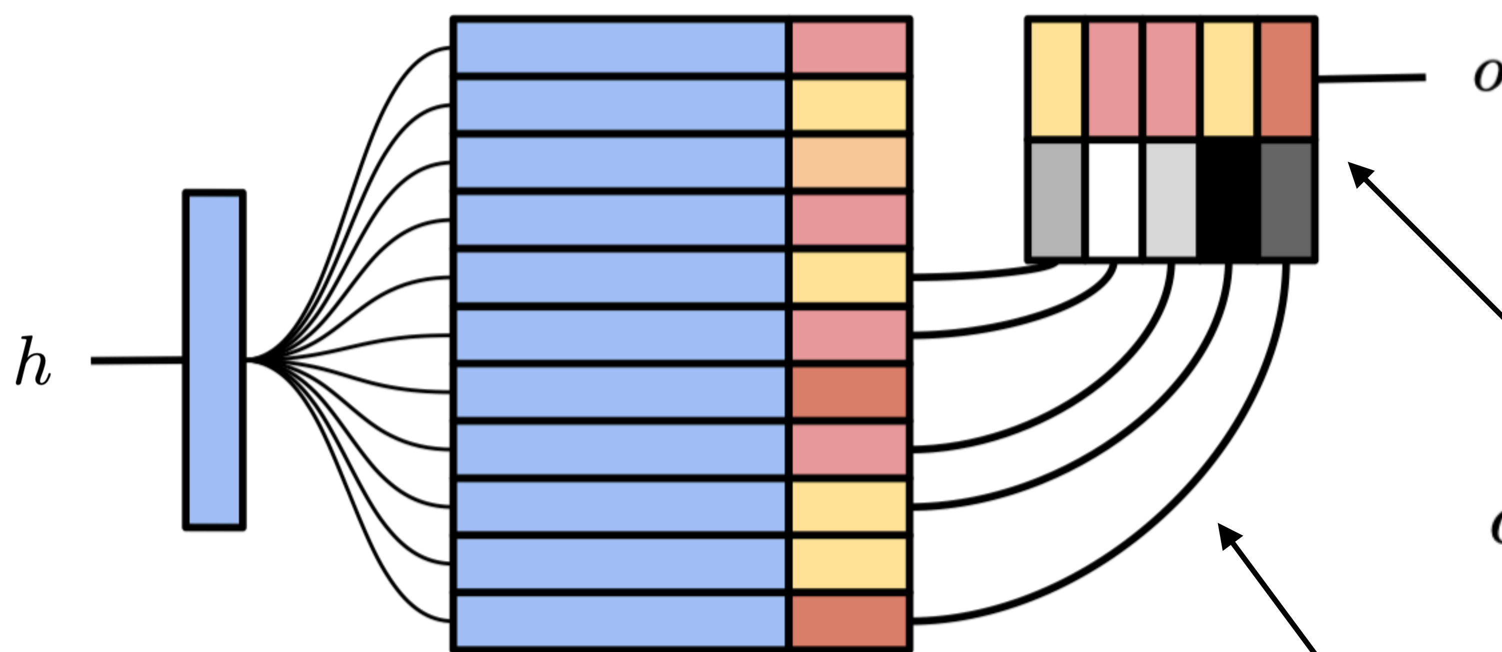
2. A CNN that processes pixel images s

3. A final network that converts memory read-outs to $Q(s, a)$ values

Neural Networks with Memory

- Neural Episodic Control (NEC) *(ICML 2017)*

Reading from memory



$$o = \sum_i w_i v_i$$

Nearest Neighbors

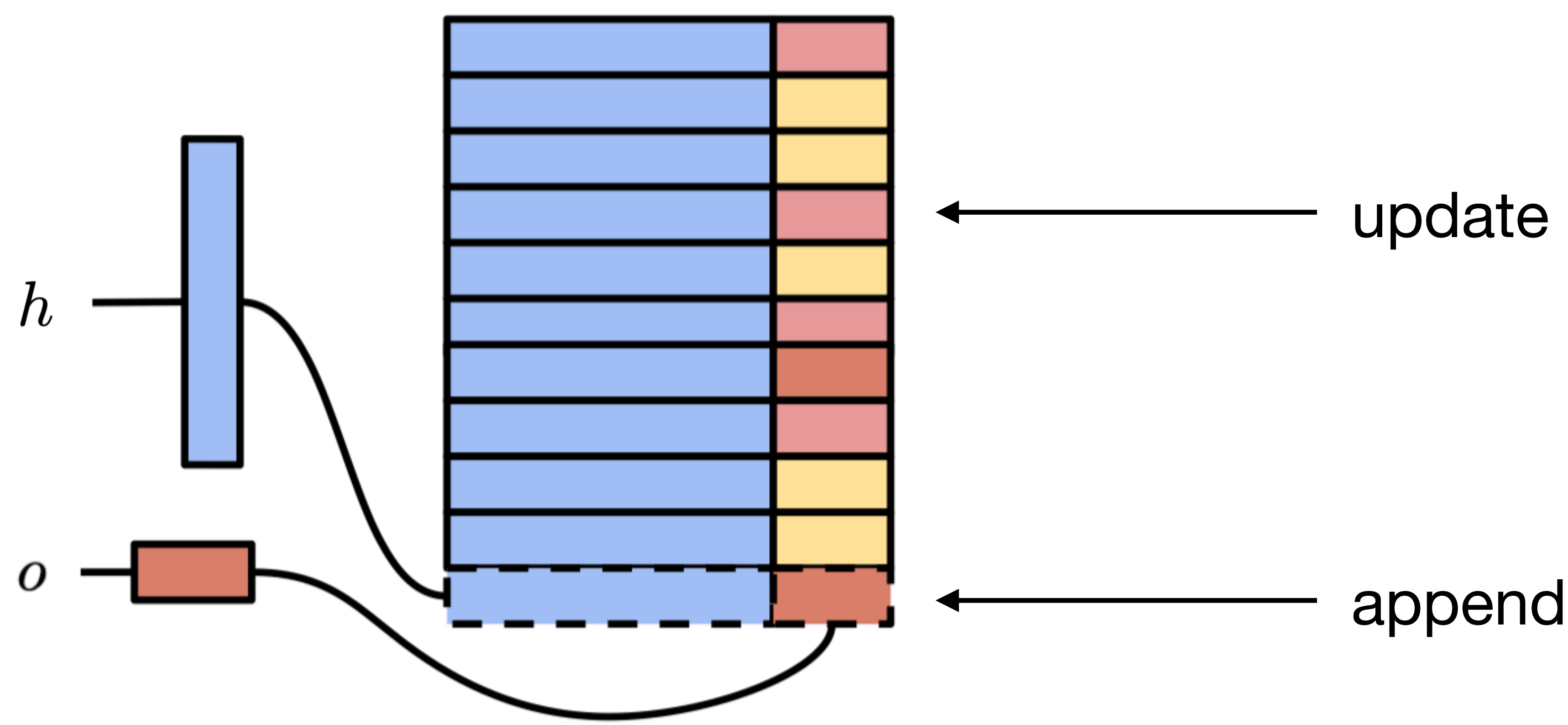
1. top $p(= 50)$
2. approximate

$$w_i = k(h, h_i) / \sum_j k(h, h_j)$$

Neural Networks with Memory

- Neural Episodic Control (NEC) *(ICML 2017)*

Writing to memory



Neural Networks with Memory

- Neural Episodic Control (NEC) *(ICML 2017)*

Algorithm 1 Neural Episodic Control

$\mathcal{D}$: replay memory.
M_a: a DND for each action a.
N: horizon for N-step Q estimate.
for each episode **do**
  for $t = 1, 2, \ldots, T$ **do**
    CNN | Receive observation s_t from environment with embedding h.
    Read from memory | Estimate $Q(s_t, a)$ for each action a via (1) from M_a
    $a_t \leftarrow \epsilon$-greedy policy based on $Q(s_t, a)$
    Take action a_t, receive reward r_{t+1}
    Write to memory | Append $(h, Q^{(N)}(s_t, a_t))$ to M_{a_t}. ← N-step Q-learning
    Append $(s_t, a_t, Q^{(N)}(s_t, a_t))$ to $\mathcal{D}$.
    Train on a random minibatch from $\mathcal{D}$.
  end for
end for

Memory + Meta-Learning

- One-Shot Learning with Memory-Augmented Neural Networks *(arXiv 2016)*

NNs with large memory are known to be quite capable of meta-learning.

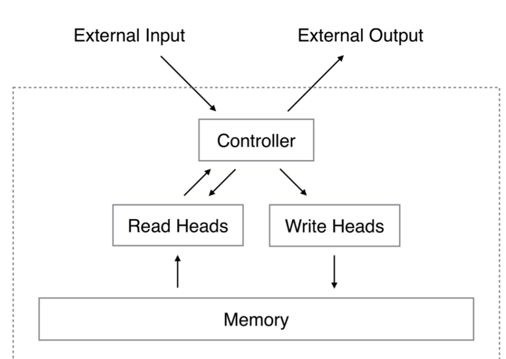
However, RNNs are not scalable enough.

Further requirements?

1. Stores information in memory in a representation that is both **stable** and **element-wise addressable.**
2. The number of parameters should not be tied with the size of the memory.

⇨ Memory-Augmented Neural Networks (MANN)

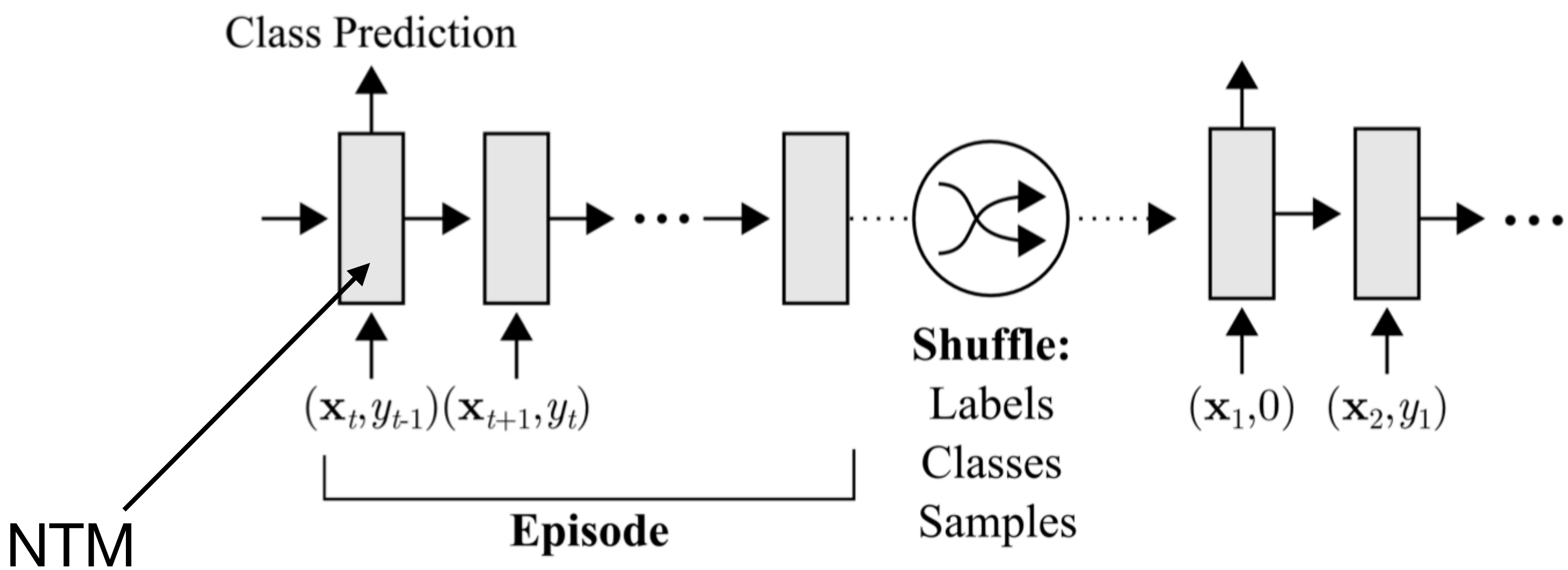
e.g. NTM (Graves et al.), Memory Networks (Weston et al.)



Memory + Meta-Learning

- One-Shot Learning with Memory-Augmented Neural Networks *(arXiv 2016)*

Setup



Task/Episode $D = \{(\mathbf{x}_t, y_t)\}_{t=1}^{T}$

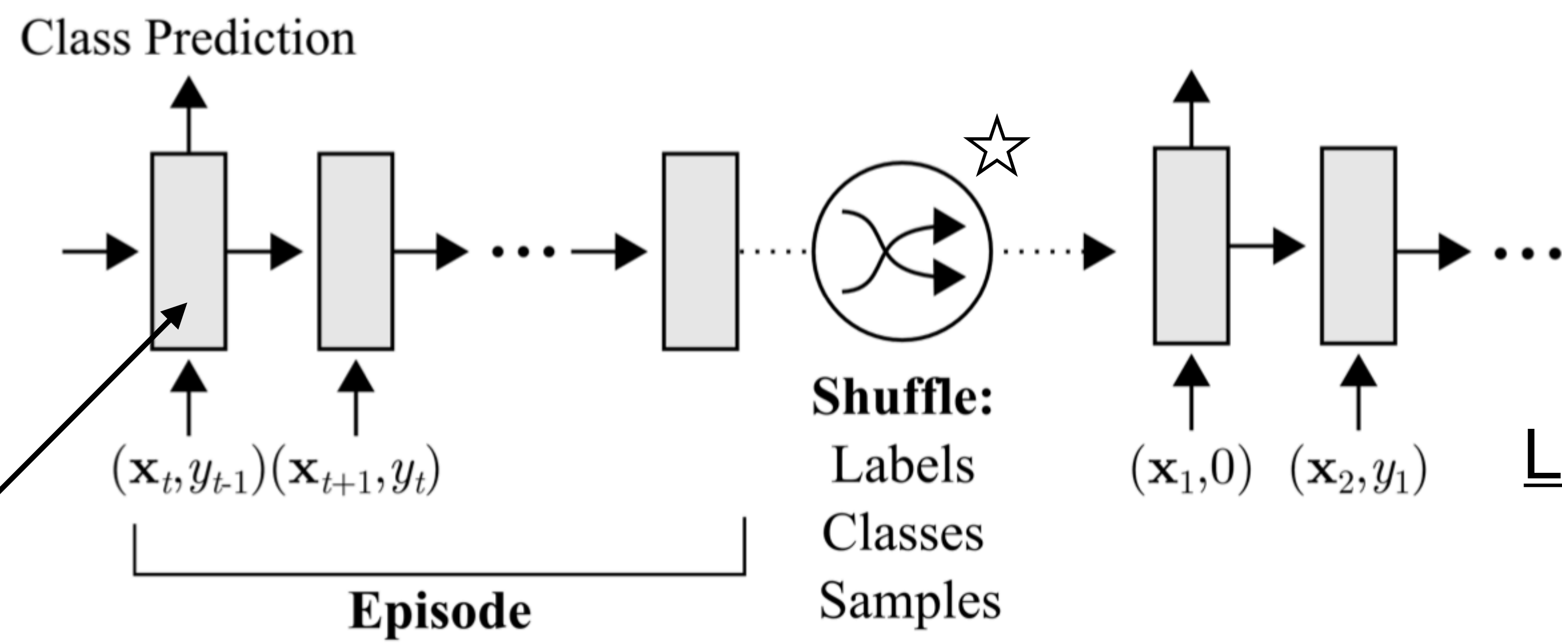
Optimization $\theta^* = \text{argmin}_\theta \, \mathbb{E}_{D \sim p(D)} \left[L(D; \theta) \right]$

Training input $(\mathbf{x}_0, \text{null}), (\mathbf{x}_1, y_0), \cdots, (\mathbf{x}_T, y_{T-1})$

Memory + Meta-Learning

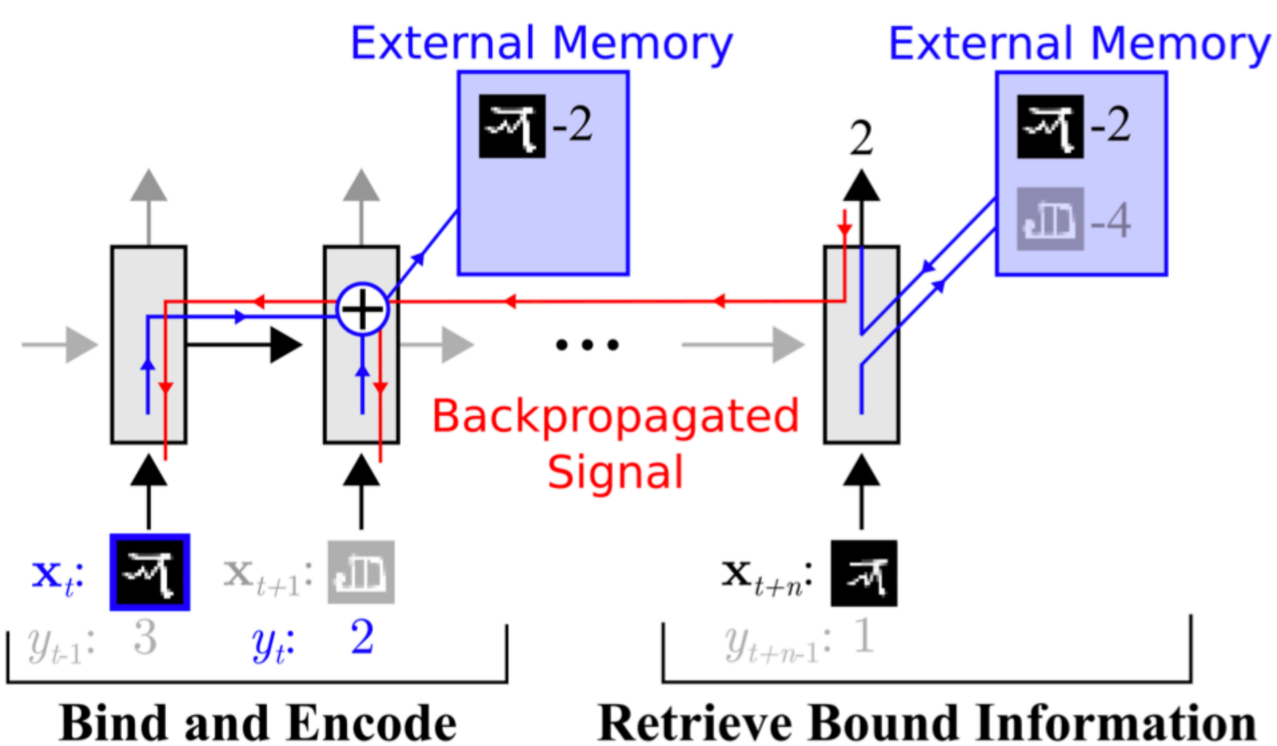
- One-Shot Learning with Memory-Augmented Neural Networks *(arXiv 2016)*

Setup



NTM

Learning to Learn!



At time t:

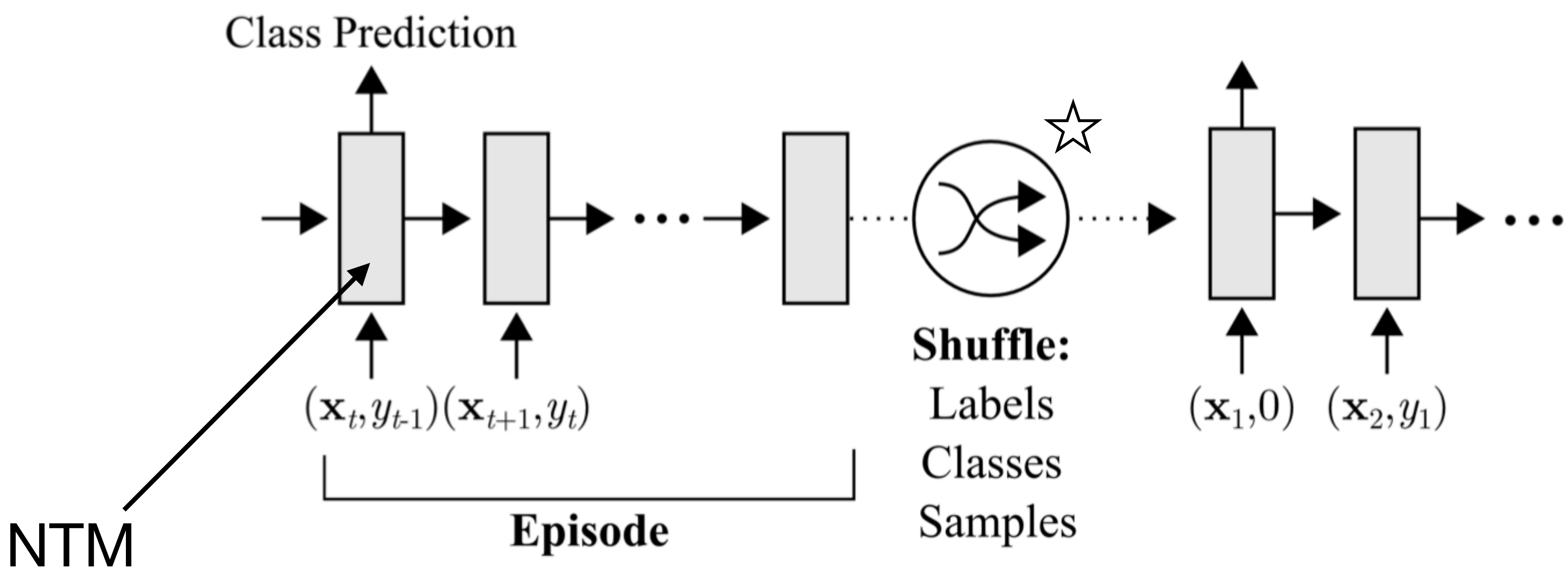
1. Read head searches memory for the closest representation to $\mathbf{x}_t$
2. Write head stores $\mathbf{x}_t$ to memory
3. Write head binds y_{t-1} with representation of $\mathbf{x}_{t-1}$ in memory

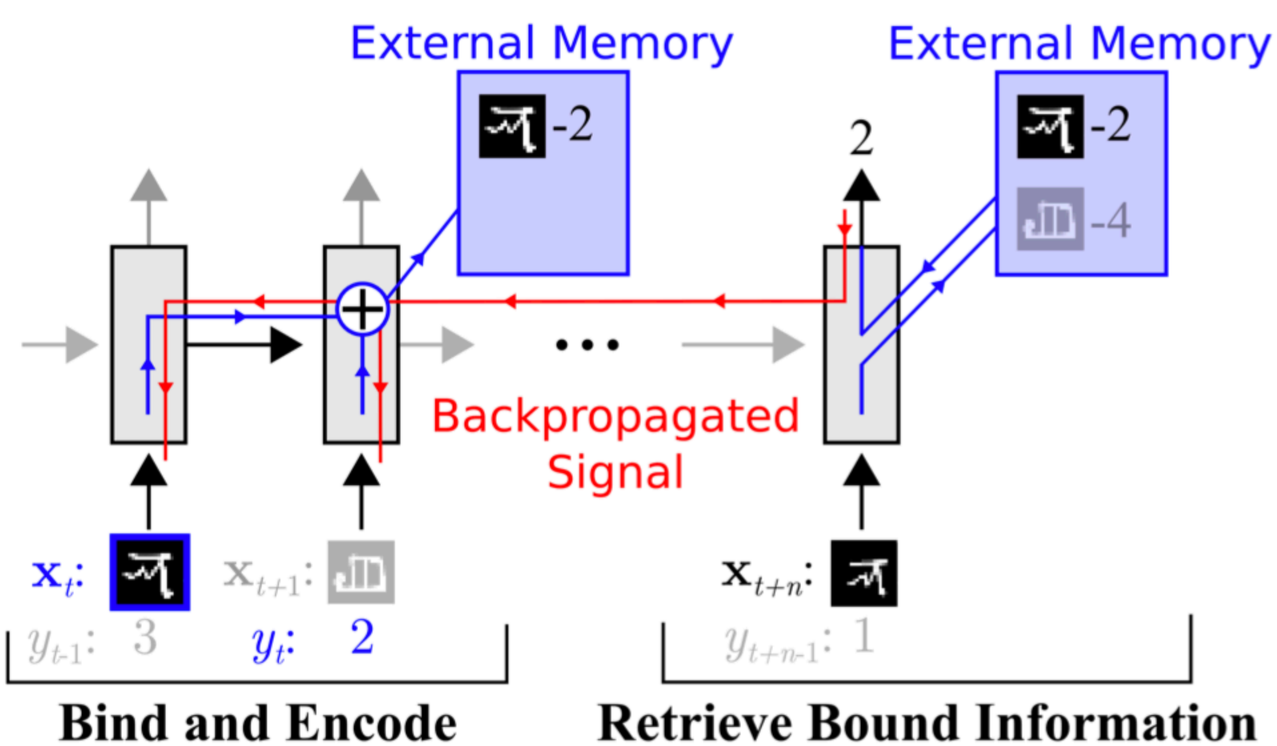
Wipe memory after each episode

Memory + Meta-Learning

- One-Shot Learning with Memory-Augmented Neural Networks *(arXiv 2016)*

Setup





Reading from memory

- Same as original NTM

Writing to memory

- Erase least recently used slot
- Update latest read slot or write to new slot

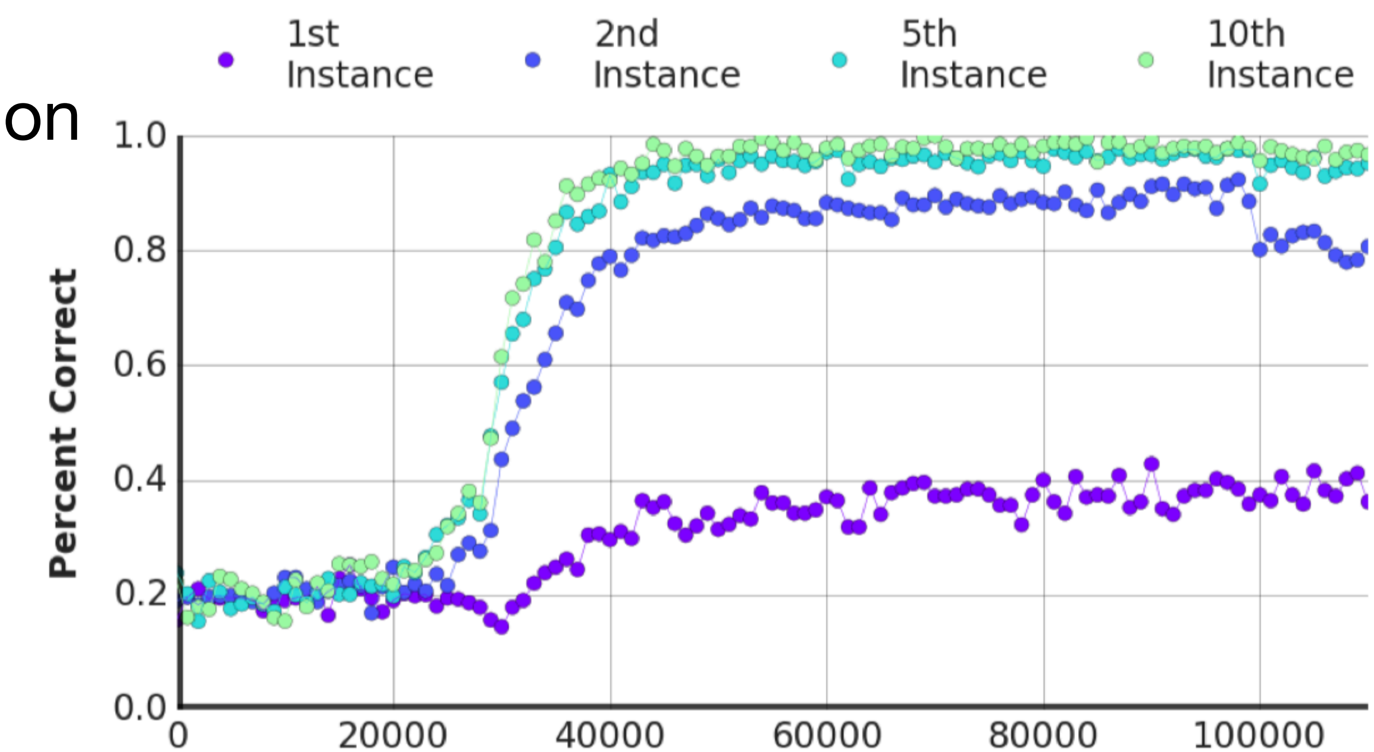
Memory + Meta-Learning

- One-Shot Learning with Memory-Augmented Neural Networks *(arXiv 2016)*

Experiments

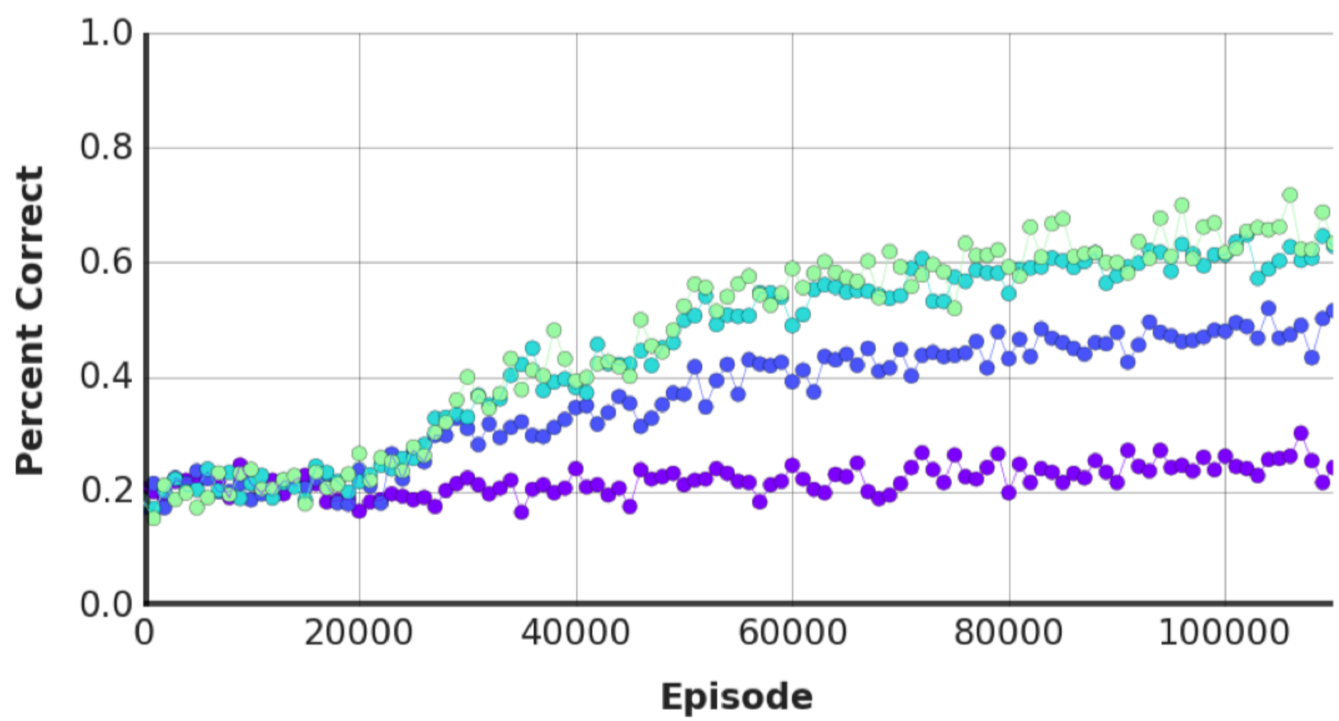
5-way classification

MANN



← intelligent guesses!

LSTM



Memory + Meta-Learning

- Been There, Done That: Meta-Learning with Episodic Recall *(ICML 2018)*

Meta-learning agents are good at rapidly learning new tasks.

However, they forget previously learned tasks.

In naturalistic environments, learners are confronted with

1. an open-ended series of related yet novel tasks, within which
2. previously encountered tasks **identifiably reoccur**.

Sample uniformly without replacement from from a bag of tasks

$$S = \{t_1, t_2, \cdots, t_{|S|}\}$$

that **contains duplicates of each task**.

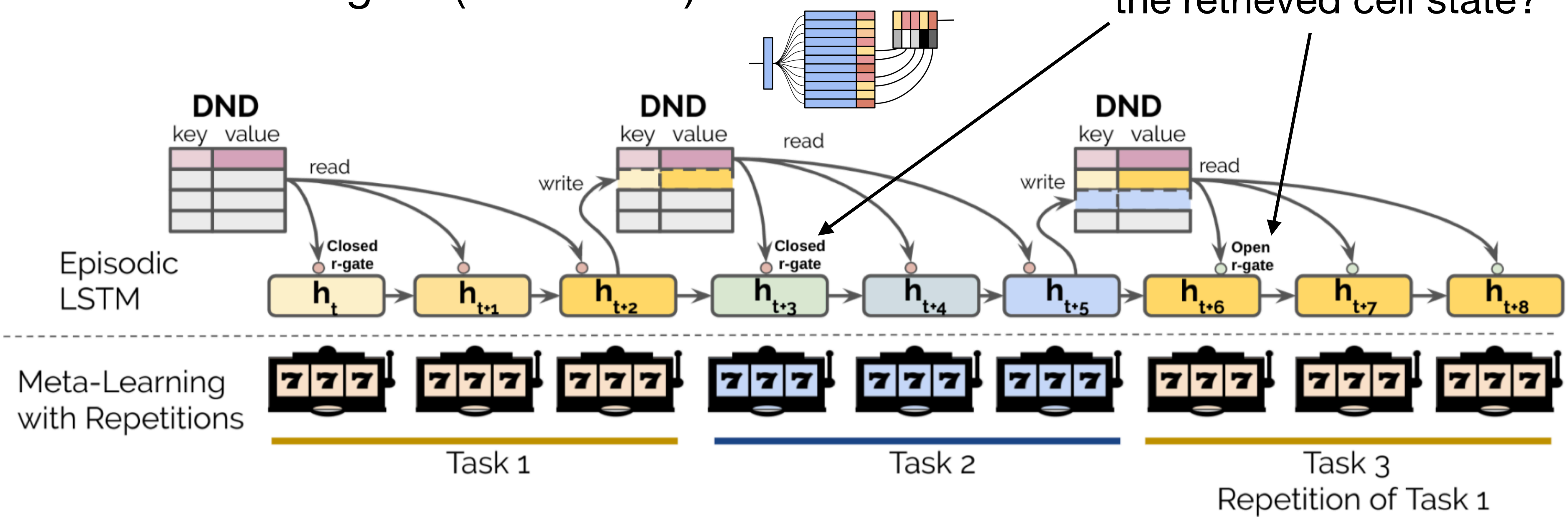
Each task is consisted of MDPs m and **context** c. That is, $t_n = (m_n, c_n)$.

Memory + Meta-Learning

- Been There, Done That: Meta-Learning with Episodic Recall *(ICML 2018)*

LSTM-based agent (from L2RL)

How do we reinstate the retrieved cell state?



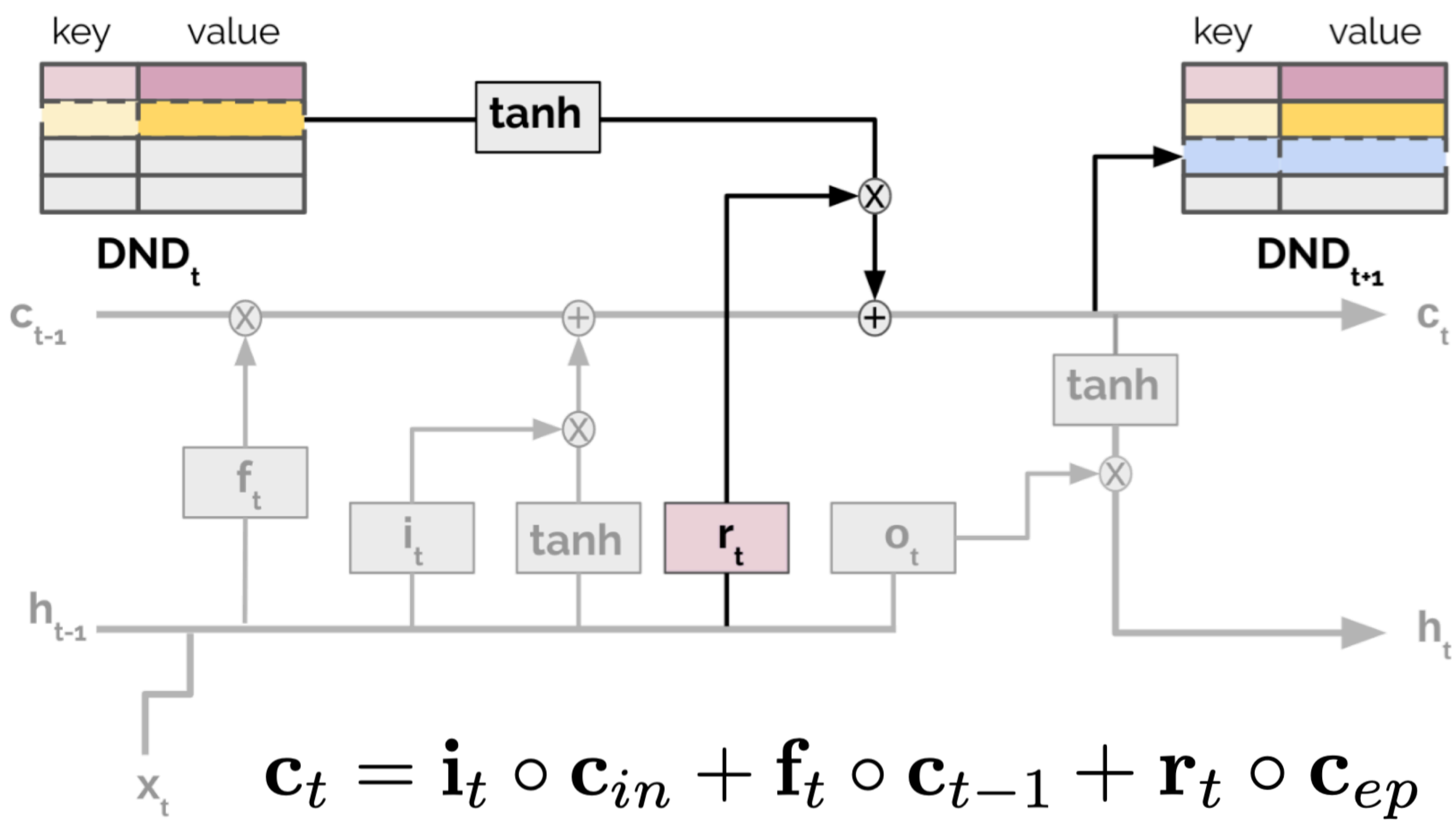
Key: Context c

Value: LSTM cell state

Memory + Meta-Learning

- Been There, Done That: Meta-Learning with Episodic Recall *(ICML 2018)*

Episodic LSTM



$$\mathbf{c}_t = \mathbf{i}_t \circ \mathbf{c}_{in} + \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{r}_t \circ \mathbf{c}_{ep}$$

Reinstatement Gate

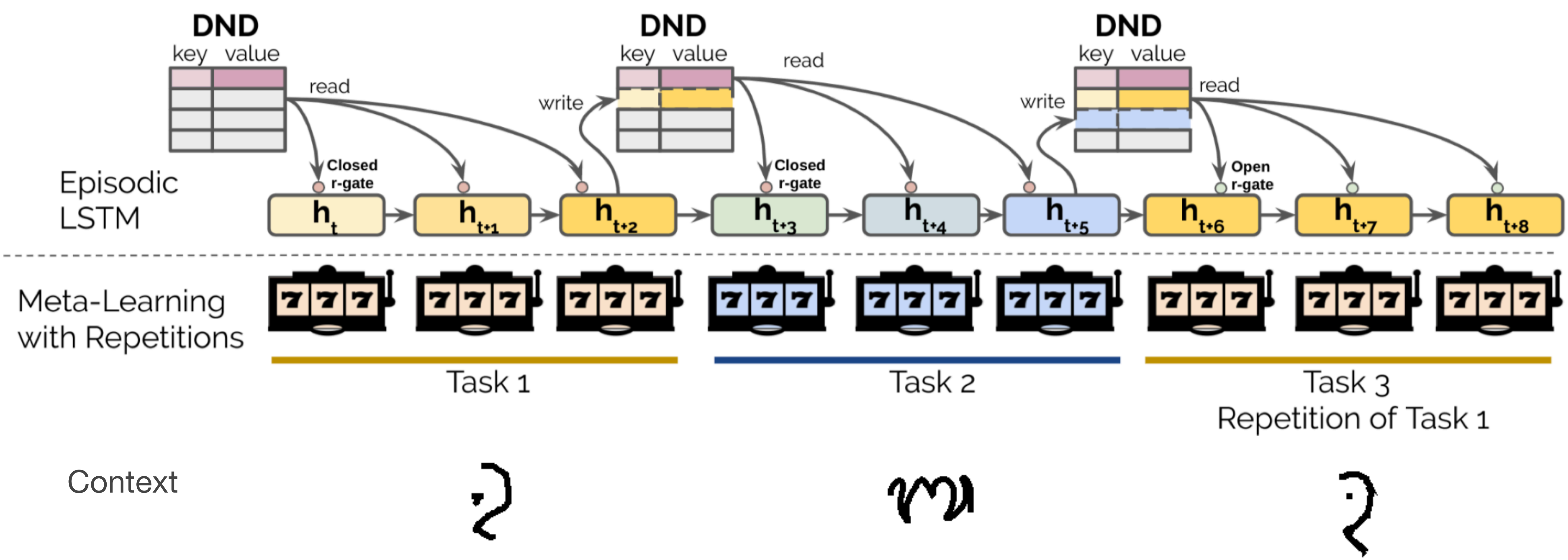
Memory + Meta-Learning

- Been There, Done That: Meta-Learning with Episodic Recall *(ICML 2018)*

Experiments

Using Omniglot characters as contexts

Each time a task reoccurs, a different drawing of the character shown to the agent!



Memory + Meta-Learning

- Rapid Adaptation with Conditionally Shifted Neurons *(ICML 2018)*

Can we shift neuron activation values based on the current task?

⇨ Conditionally Shifted Neurons (CSN)

Description Phase

1. Process D_τ and extracts conditioning information.
2. Generate activation shifts and stores them in a key-value memory.

Prediction Phase

1. Retrieve shifts from memory and applies them to the neurons.
2. Produces predictions for unseen datapoints.

Memory + Meta-Learning

- Rapid Adaptation with Conditionally Shifted Neurons *(ICML 2018)*

Conditionally Shifted Neurons (CSN)

for simple feed-forward networks

$$h_t = \begin{cases} \sigma(a_t) + \sigma(\beta_t) & t \neq T \\ \text{softmax}(a_t + \beta_t) & t = T \end{cases}$$

non-output layers

output layer

pre-activation vector

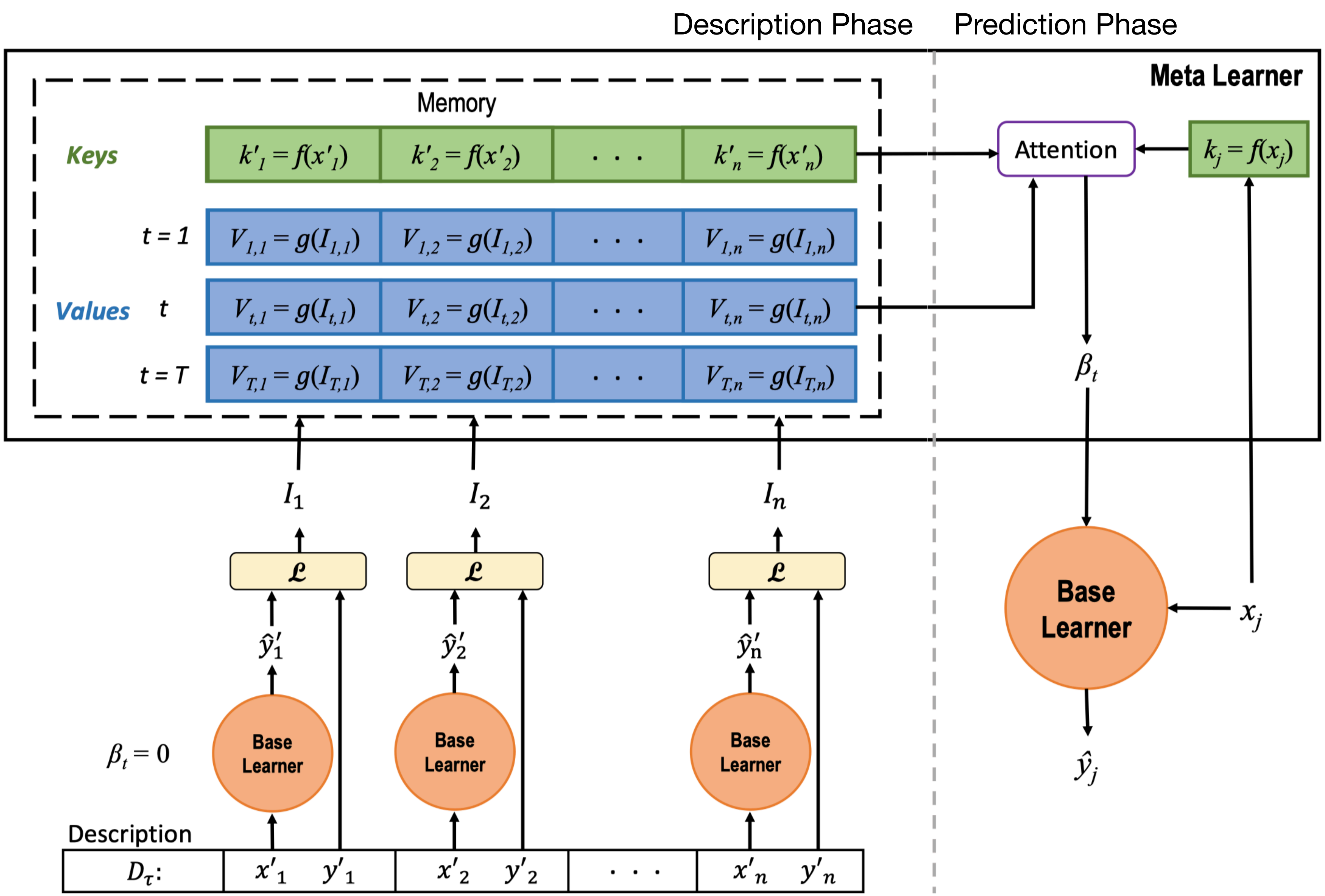
conditional shift vector

$$a_t \; = \; W_t h_{t-1} + b_t$$

Memory + Meta-Learning

- Rapid Adaptation with Conditionally Shifted Neurons *(ICML 2018)*

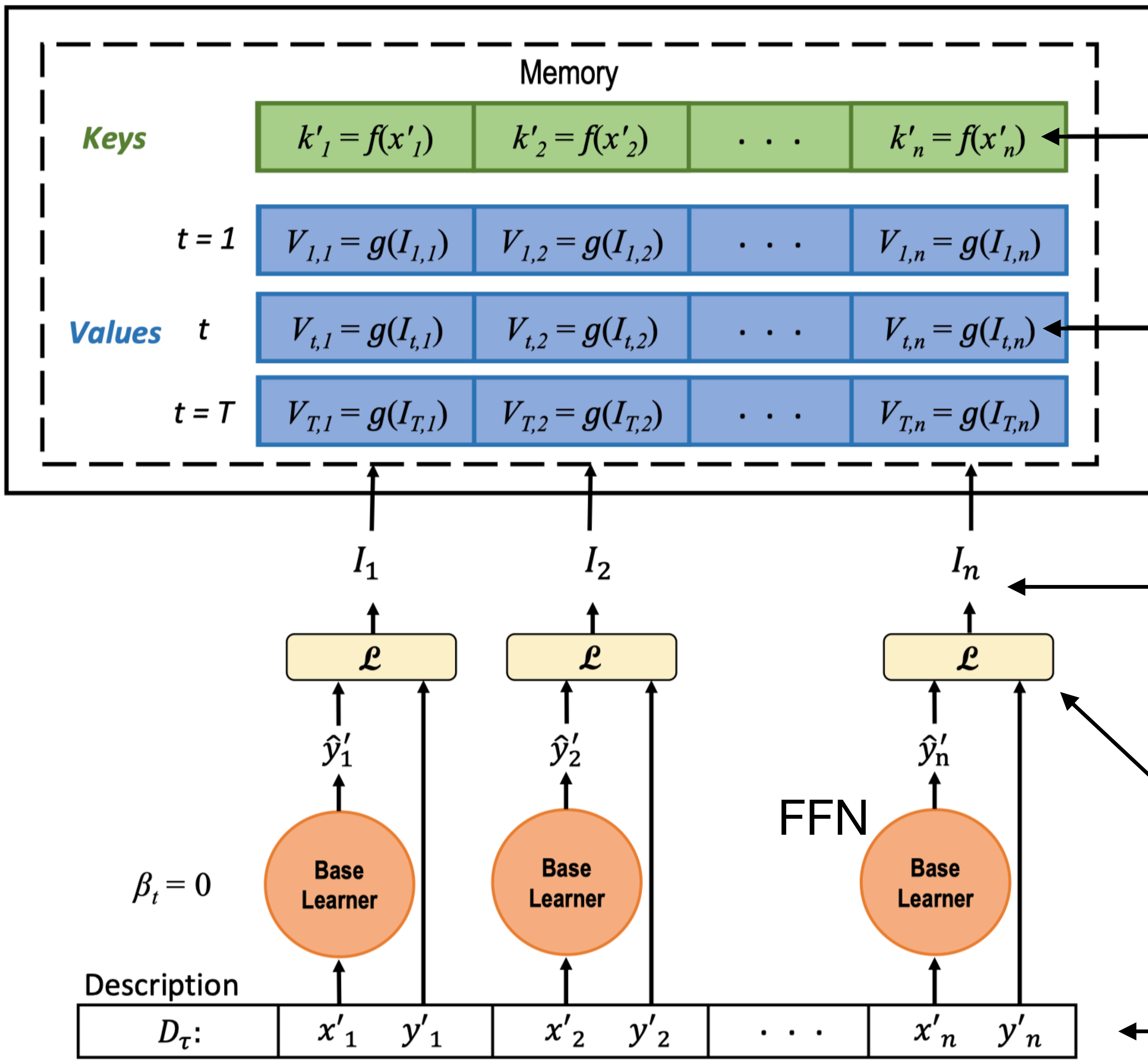
Procedure



Memory + Meta-Learning

- Rapid Adaptation with Conditionally Shifted Neurons *(ICML 2018)*

Description Phase (1) Extract conditioning vectors and (2) store them in memory



Compute keys

$k'_i = f(x'_i)$

where $f(\cdot)$ is an MLP

Compute values

$V_i = g(I_{t,i})$

where $g(\cdot)$ is an MLP

Compute conditioning information vector

$I_{t,i} = \sigma'(a_t) \cdot (\hat{y}'_i - y'_i)$

Compute loss

Training inputs

Memory + Meta-Learning

- Rapid Adaptation with Conditionally Shifted Neurons *(ICML 2018)*

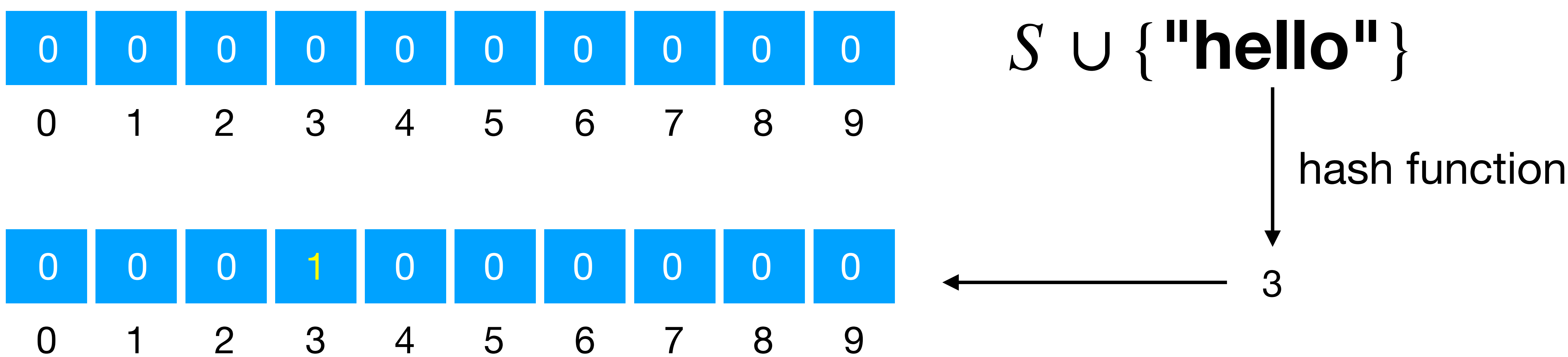
CSN can be applied pretty easily to ResNets, CNNs, and LSTMs.

SOTA performance on Mini-ImageNet 5-way classification at its time!

Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Bloom Filters: answers set membership queries



$$S = \{\textbf{"hello"}, \textbf{"world"}, \textbf{"deepest"}\}$$

0	1	0	1	0	0	0	1	0	0
0	1	2	3	4	5	6	7	8	9

Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

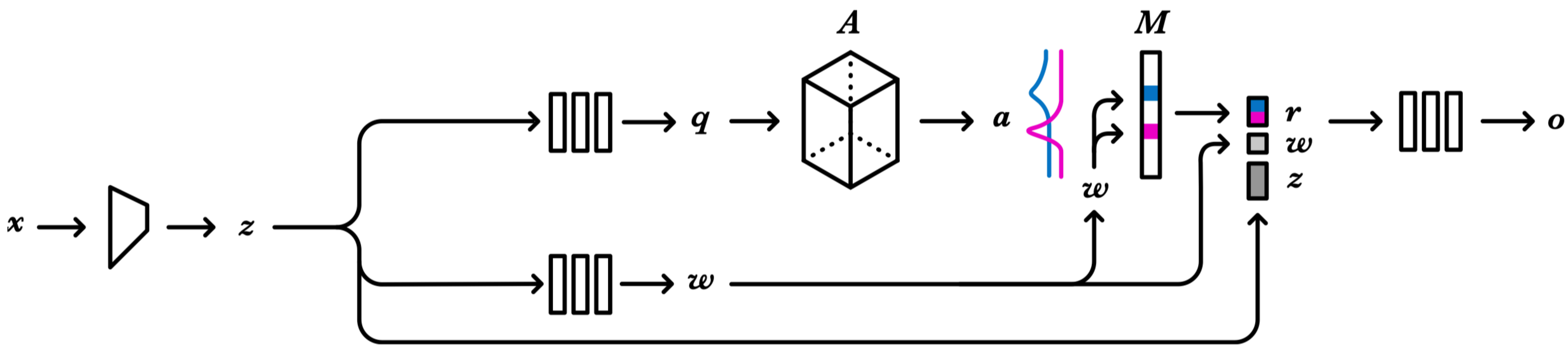
Replacing algorithms with neural networks

1. those that are configured by heuristics
2. those that do not take advantage of the data distribution

Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

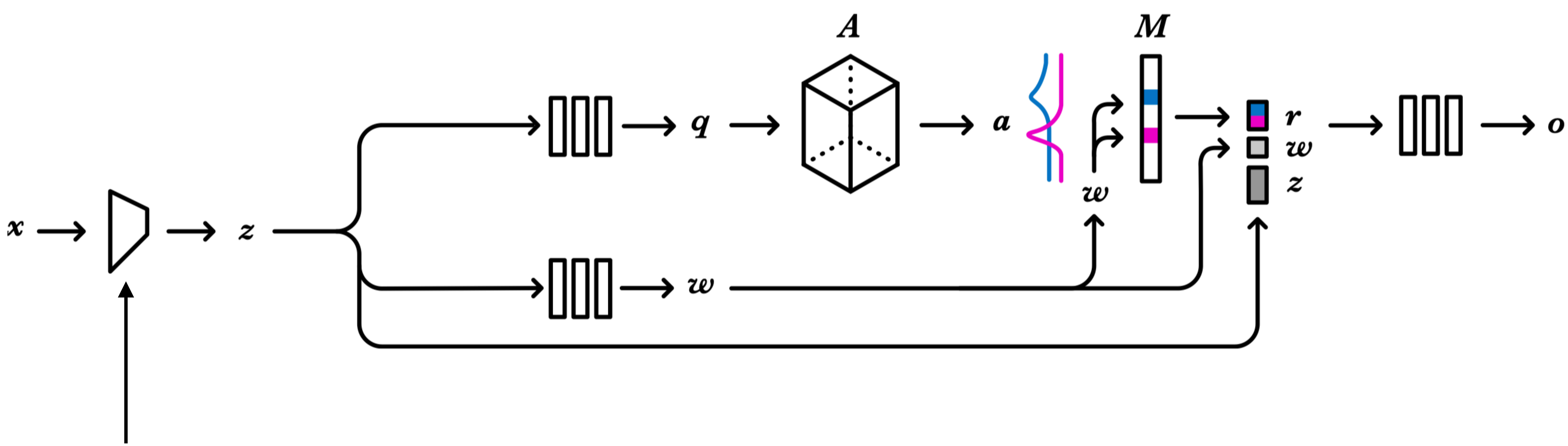
Neural Bloom Filter



Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Neural Bloom Filter



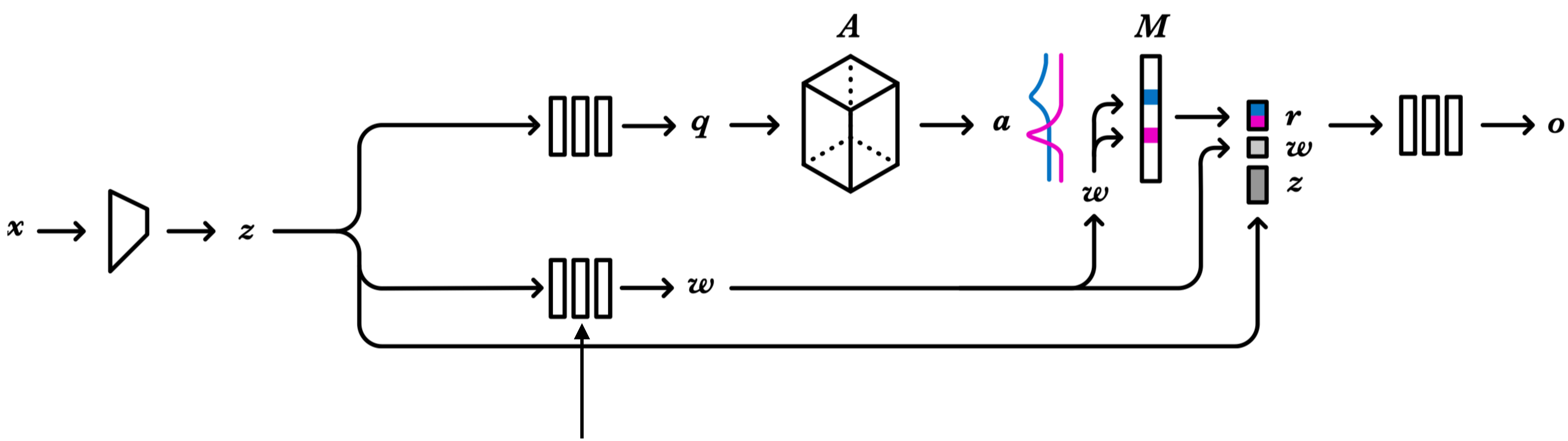
Encoder network

- 3-layer CNN for images
- 128-hidden unit LSTM for text

Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Neural Bloom Filter



Query and write word network

- 3-layer MLPs

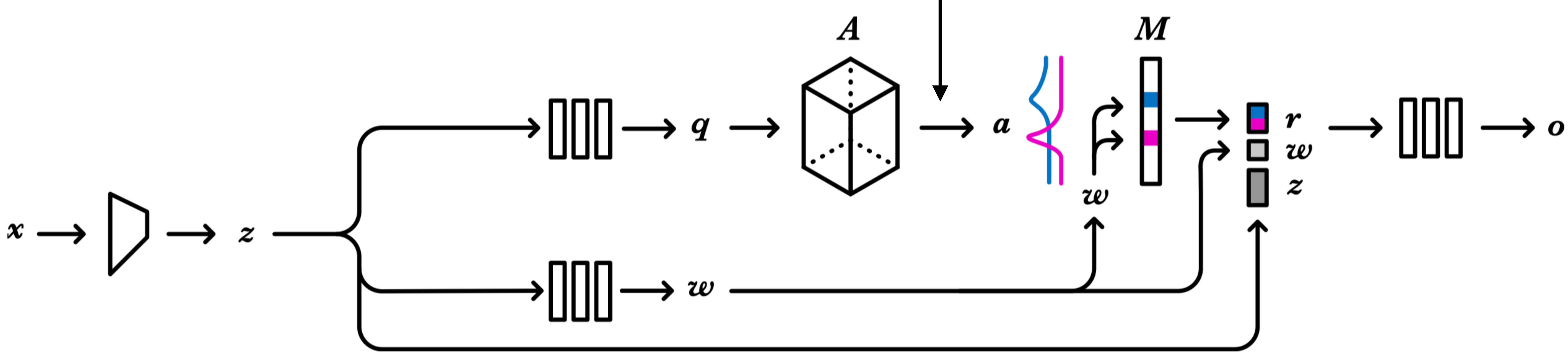
Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Neural Bloom Filter

Address over memory M

$a = softmax(q^T A)$



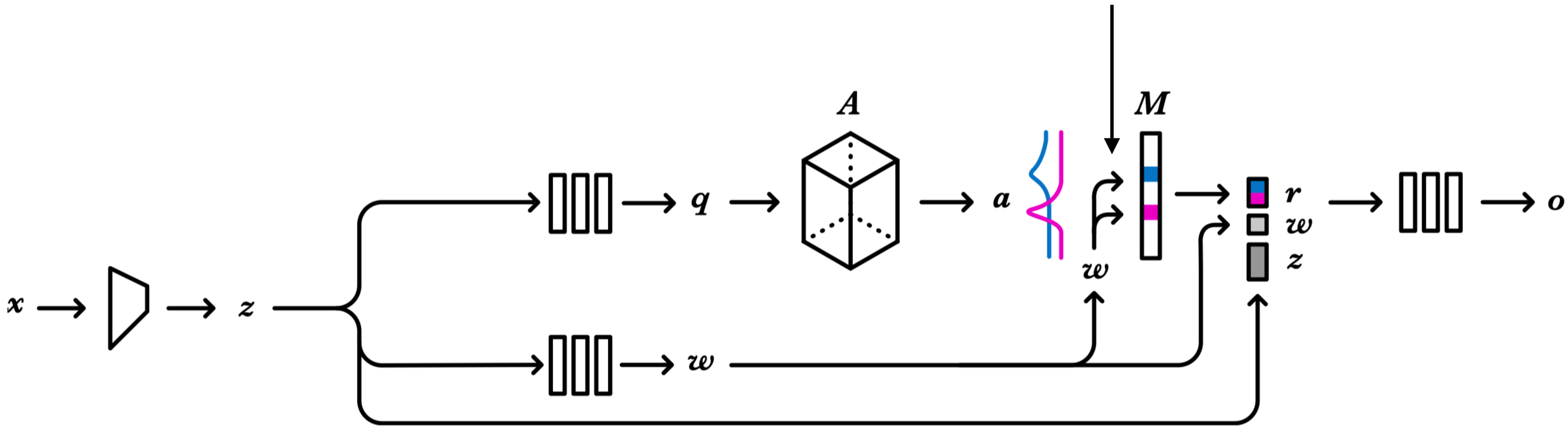
Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Neural Bloom Filter

Writing to memory

$$M_{t+1} = M_t + wa^T$$



Memory + Meta-Learning

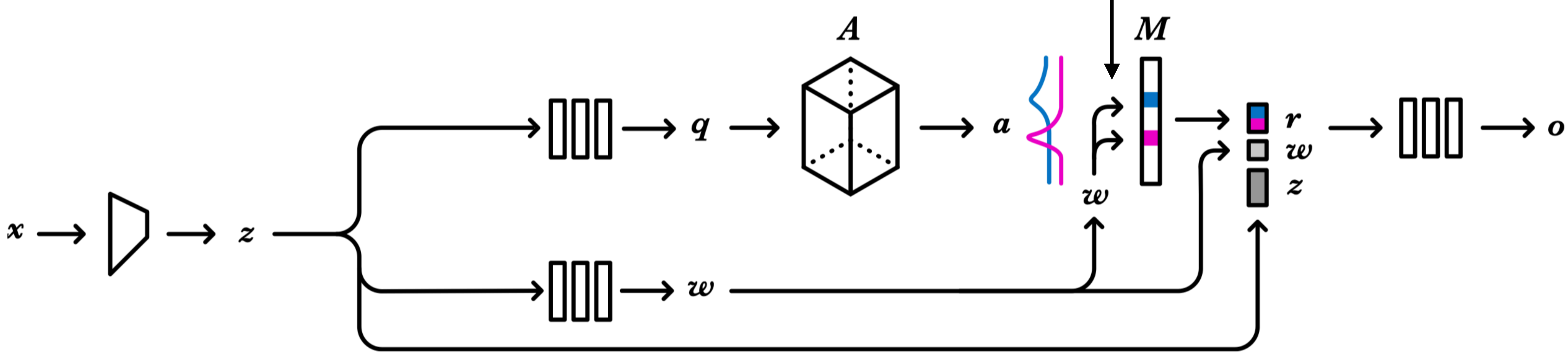
- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Only additive write: parallelizable!

Neural Bloom Filter

Writing to memory

$$M_{t+1} = M_t + wa^T$$



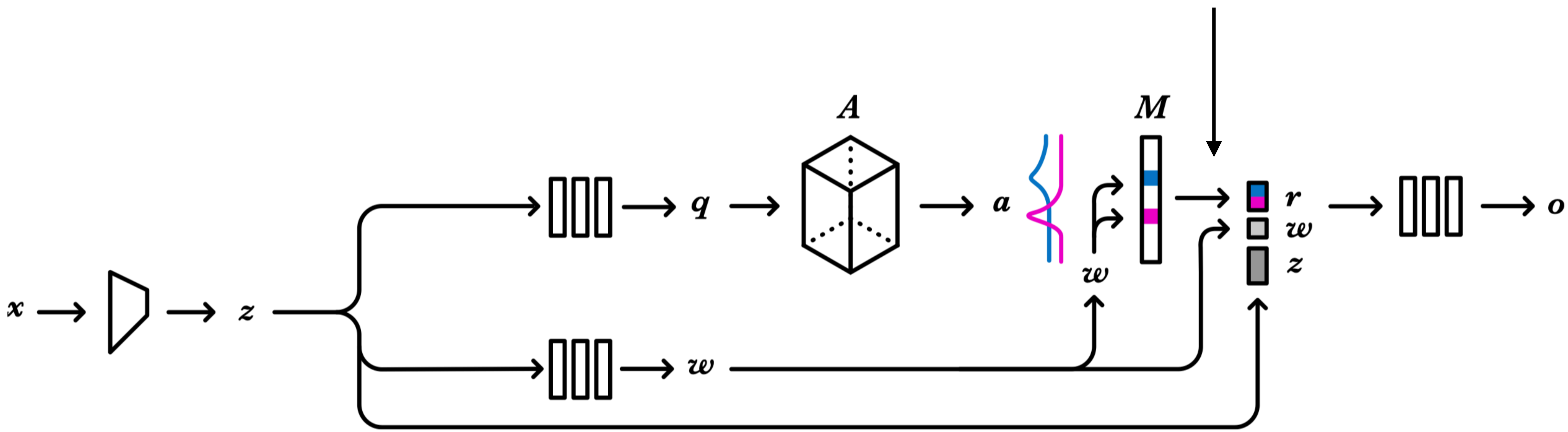
Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Neural Bloom Filter

Reading from memory

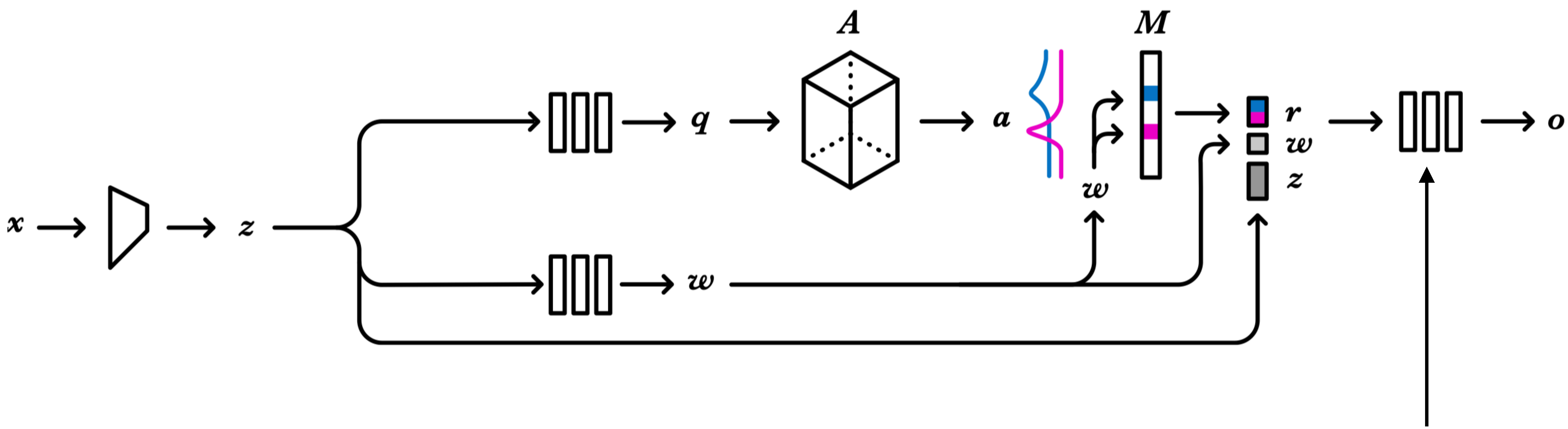
$r = M \odot a$



Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

Neural Bloom Filter



Output network
- 3-layer MLP

Memory + Meta-Learning

- Meta-Learning Neural Bloom Filters *(ICML 2019)*

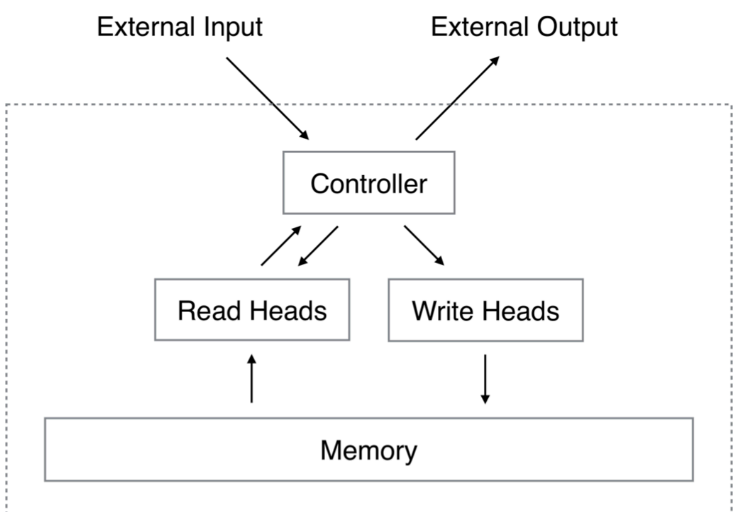
Algorithm 2 Meta-Learning Training

Let S^{train} denote the distribution over sets to store.
Let Q^{train} denote the distribution over queries.
for $i = 1$ **to** max train steps **do**
Sample task:
Support Set | Sample set to store: $S \sim \mathcal{S}^{train}$
Query Set | Sample t queries: $x_1, \ldots, x_t \sim Q^{train}$
Targets: $y_j = 1$ if $x_j \in S$ else 0; $j = 1, \ldots, t$
One-Shot Learning | Write entries to memory: $M \leftarrow f_\theta^{write}(S)$
Calculate logits: $o_j = f_\theta^{read}(M, x_j)$; $j = 1, \ldots, t$
XE loss: $L = \sum_{j=1}^{t} y_j \log o_j + (1 - y_j)(1 - \log o_j)$
Learning to Learn | Backprop through queries and writes: $dL/d\theta$
Update parameters: $\theta_{i+1} \leftarrow \text{Optimizer}(\theta_i, dL/d\theta)$
end for

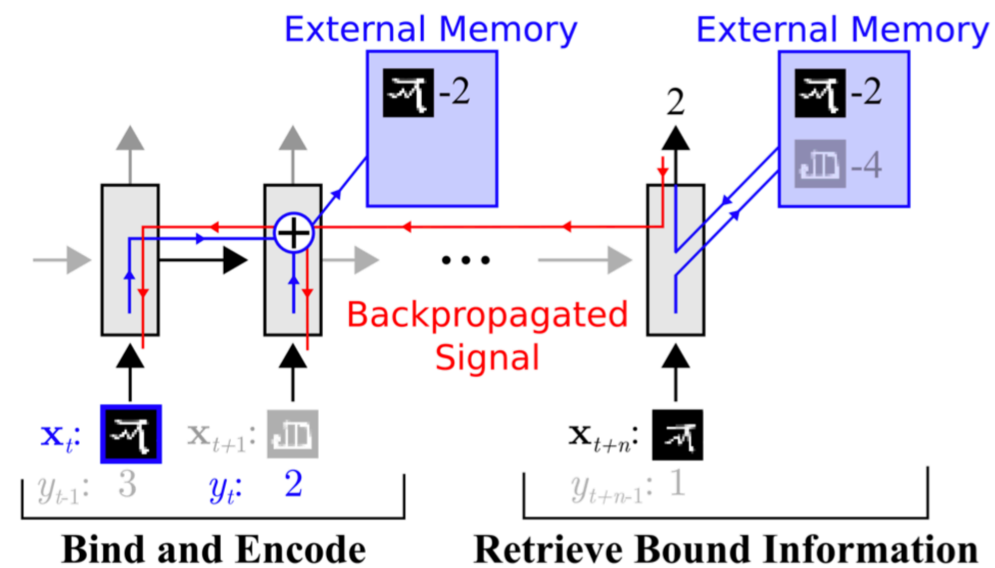
Summary

Memory Matrix

Neural Turing Machines

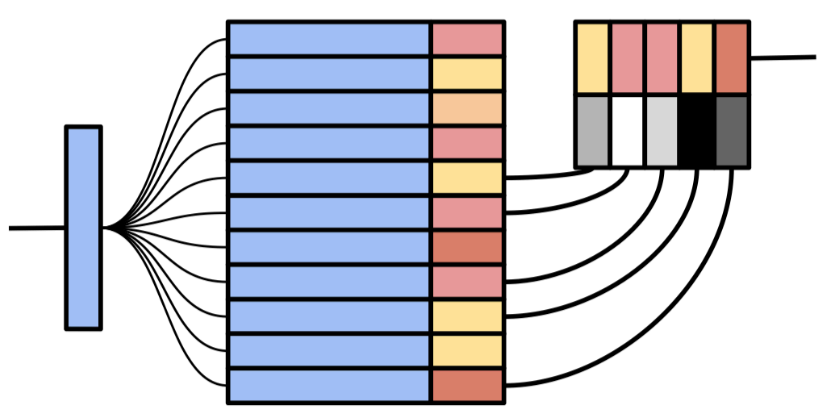


Meta-Learning with MANNs

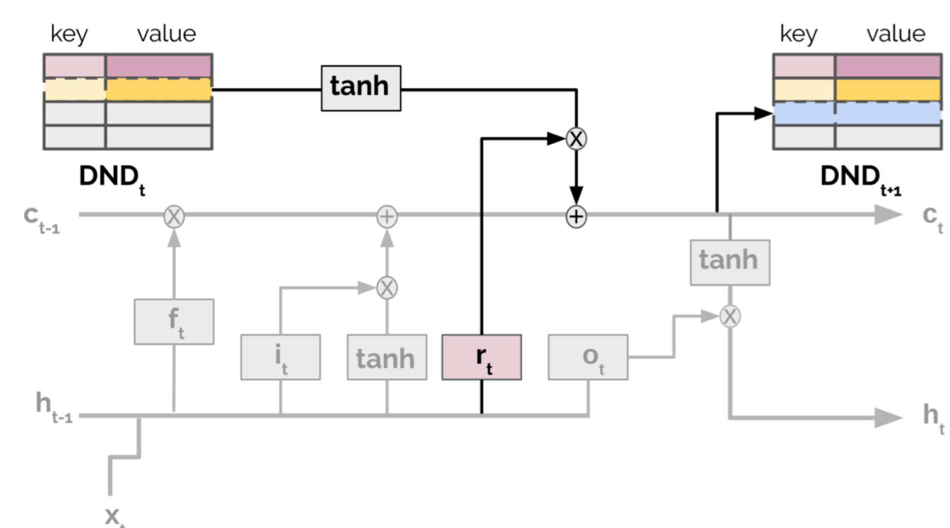


Key-Value based Memory

Differentiable Neural Dictionaries



Meta-RL with Episodic LSTMs



Summary

Key-Value based Memory

Conditionally Shifted Neurons ⟶ Rapid Adaptation to task at hand