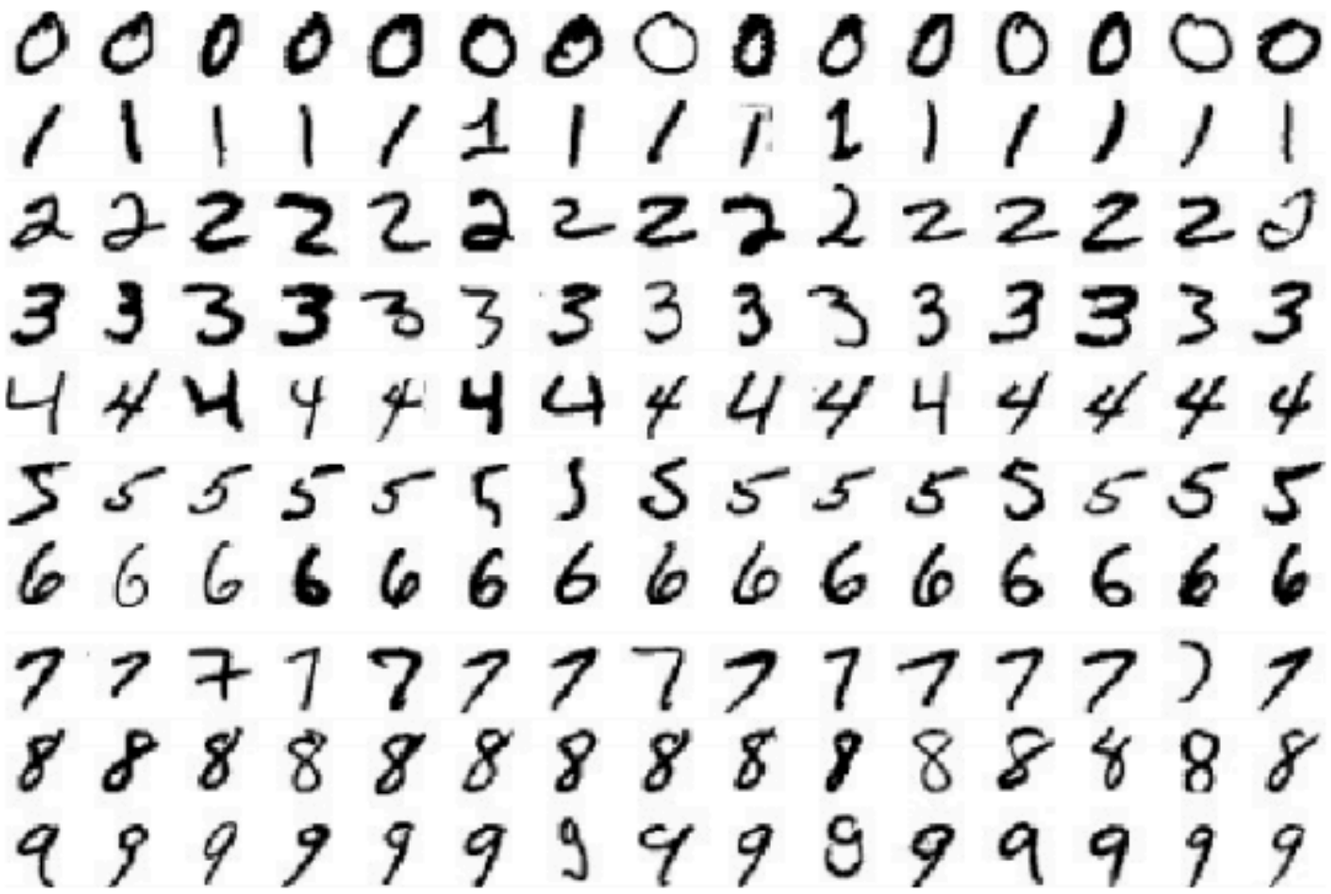
Meta-Learning

Xinkang Jia

Table

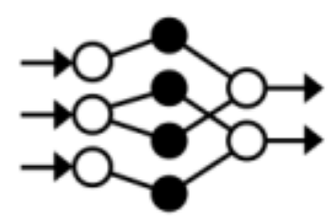
Introduction

Neural networks require a huge training data



50K images

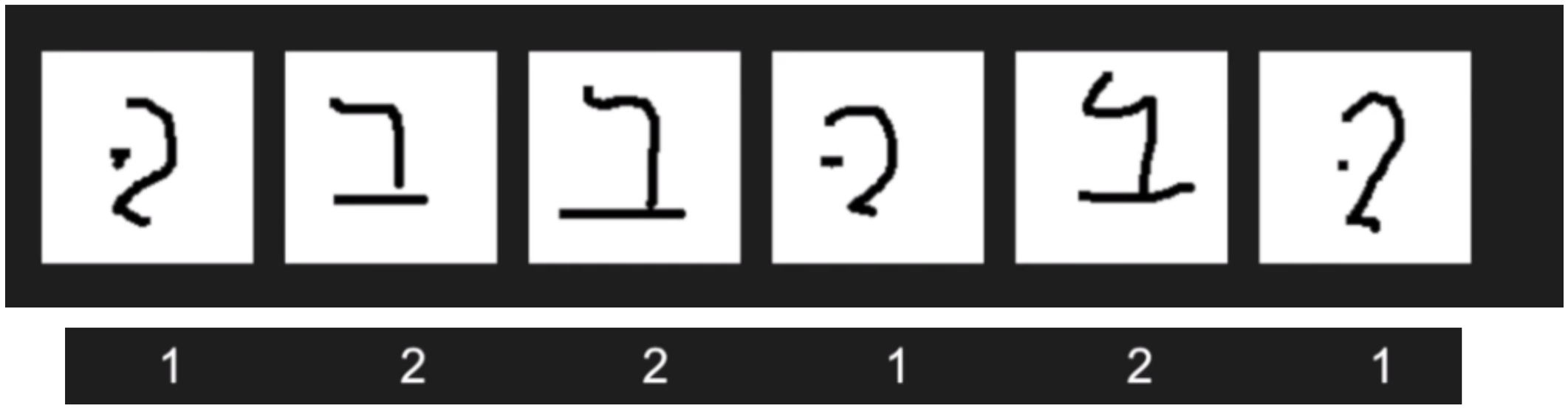
Train



Deep Neural Networks

Introduction

Few/one/zero shot learning



Introduction

What’s meta-learning

- Meta-Learning: Learning about learning
- Meta: beyond training

What’s meta

- Hyper Parameters
- Model Architecture
- Initialization
- Optimizer
- Parameters
- Loss Functions
- Back Propagation

- Some problems of interest require rapid inference from small quantities of data
- This kind of flexible adaption is a celebrated aspect of human learning

Main Idea

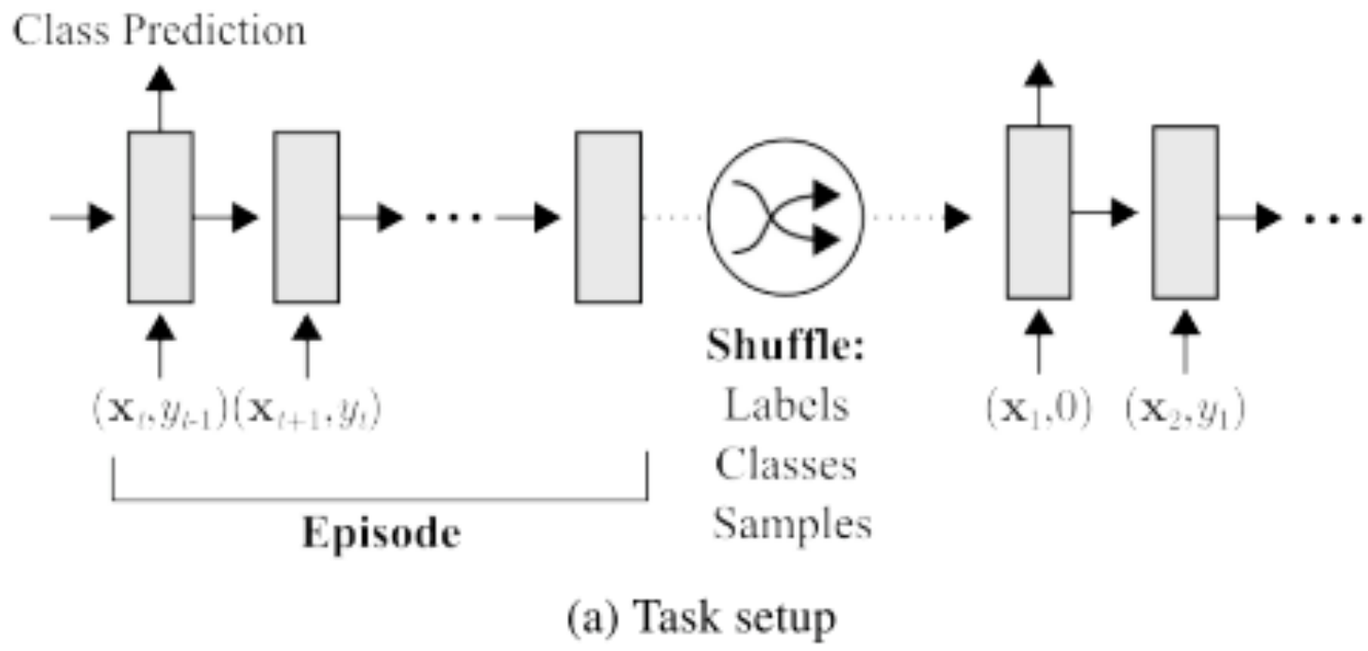
- Learn to do classification on unseen class
- Learn the sample-class binding on memory instead of weights
- Let the weights learn higher level knowledge

Meta learning task

$$\theta^* = \text{argmin}_\theta E_{D\sim p(D)}[\mathcal{L}(D;\theta)].$$

- Differences:
  - A distribution of dataset p(D)

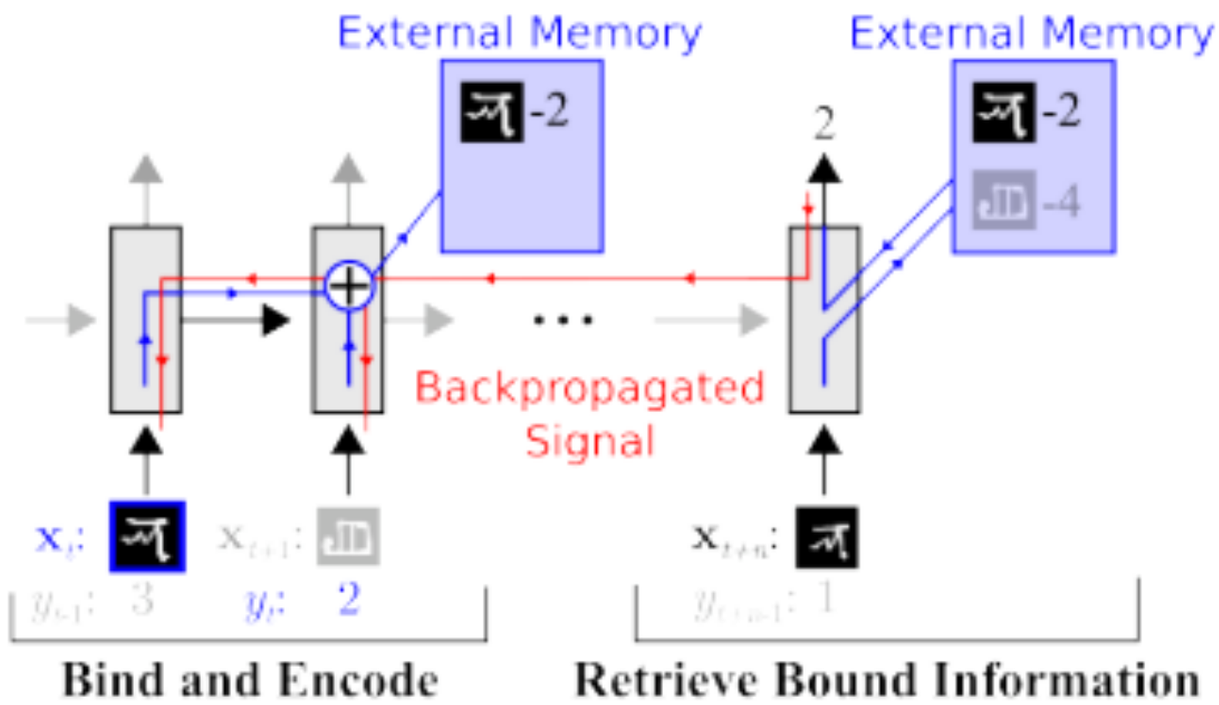
Meta learning task



(a) Task setup

Prevents the network from slowly learning sample-class bindings in its weights

Meta learning task



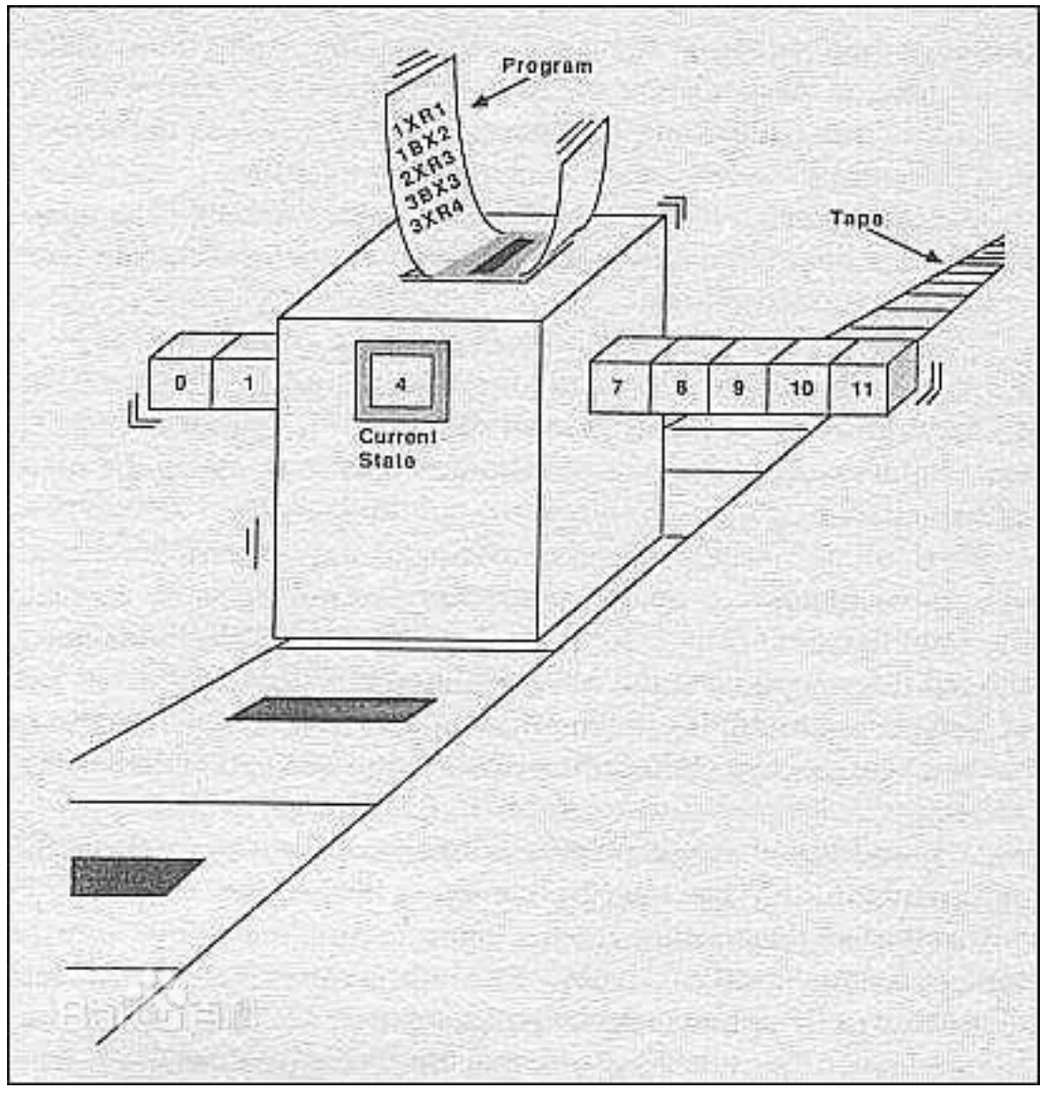
(b) Network strategy

Learn to hold data samples in memory until the appropriate labels are presented at the next time-step

Neural Turing Machines

What’s Turing machines

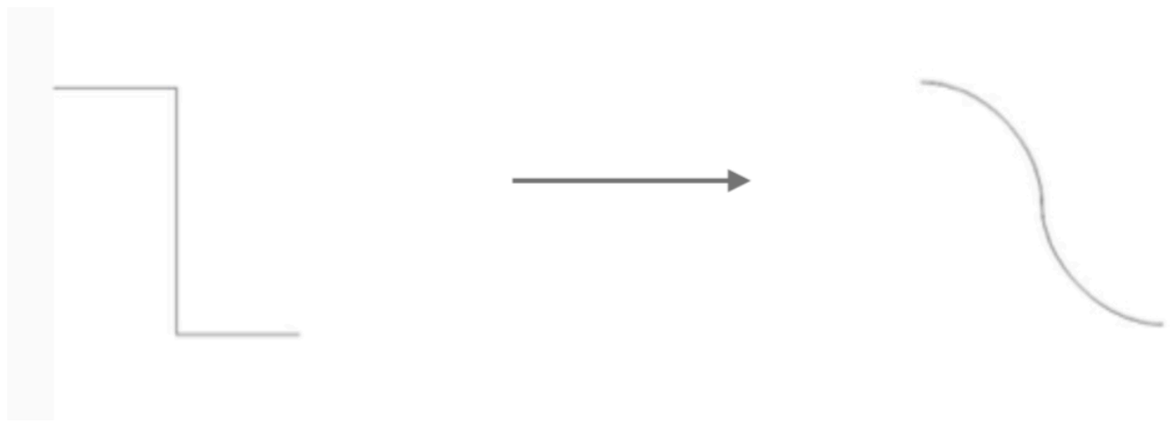
- A model of computer
- Memory tape with Read and Write heads
- Controller(Program) attends to specific element
- Discrete, so not possible train with backpropagation



Neural Turing Machines

What’s Neural Turing machines

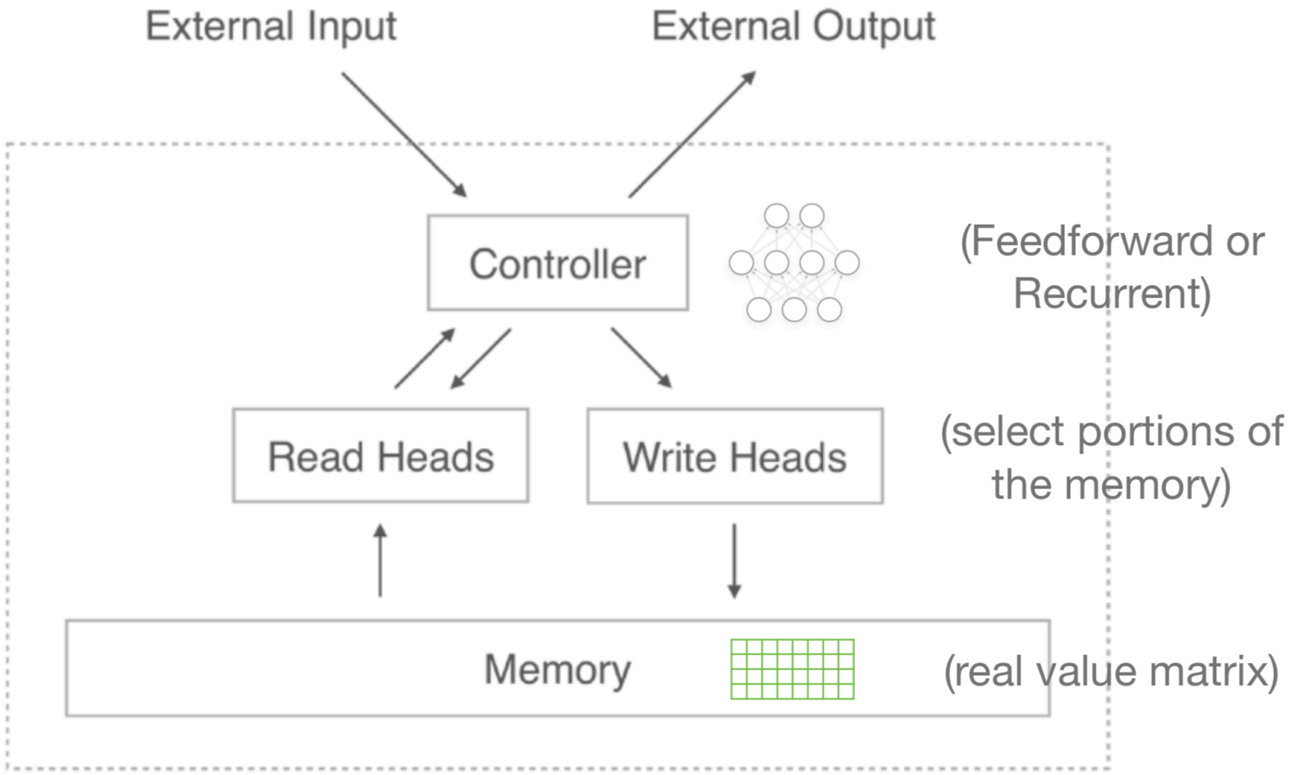
- Narrow gap between neural networks and algorithms
- ‘Differentiable’ Turing machine
- ‘Sharp’ functions made smooth and can train with backpropagation



Neural Turing Machines

What's Neural Turing machines

- Neural Network 'Controller'
- Memory
- Learns from sequence



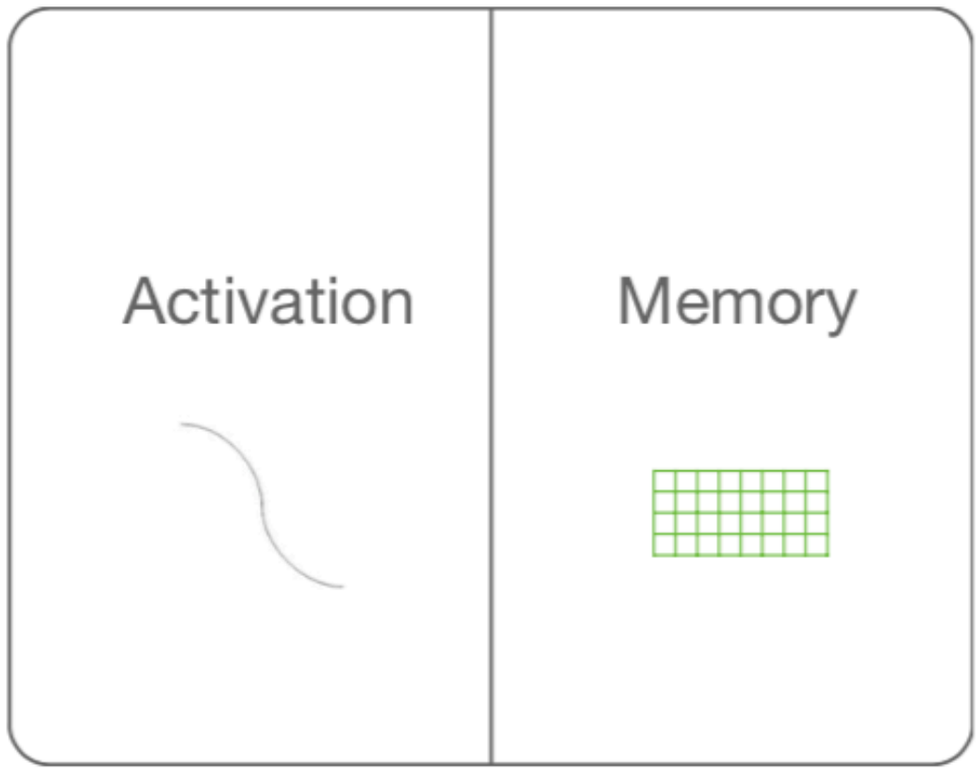
Neural Turing Machines

Why not RNN or LSTM

- Memory is tied up in the activations of the latent state of the network

- High Computation Cost: More memory, bigger size of the network

- Content Fragility: Being constantly updated with new info



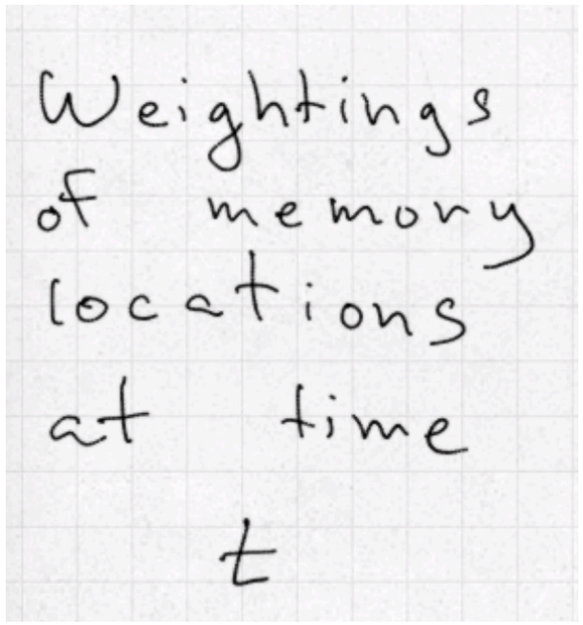
Neural Turing Machines

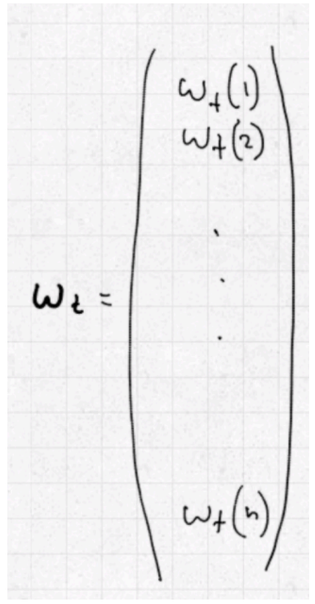
Why NTM?

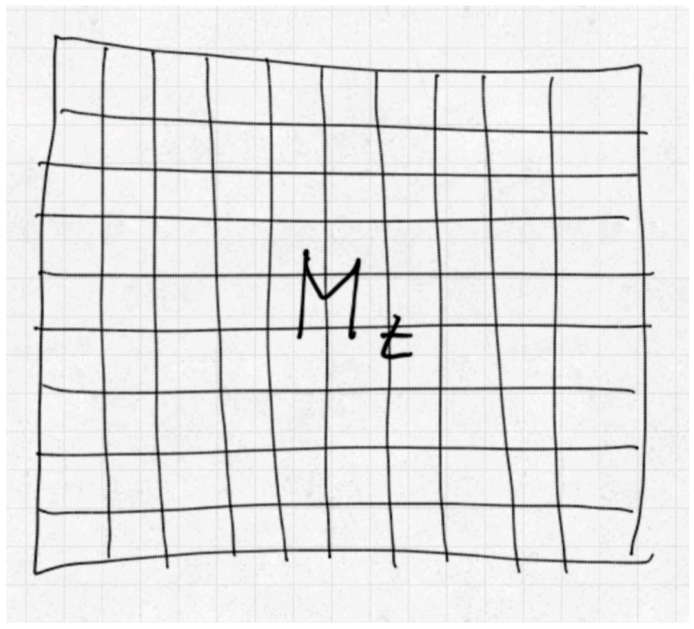
- Rather than artificially increasing the size of the hidden state in the RNN or LSTM

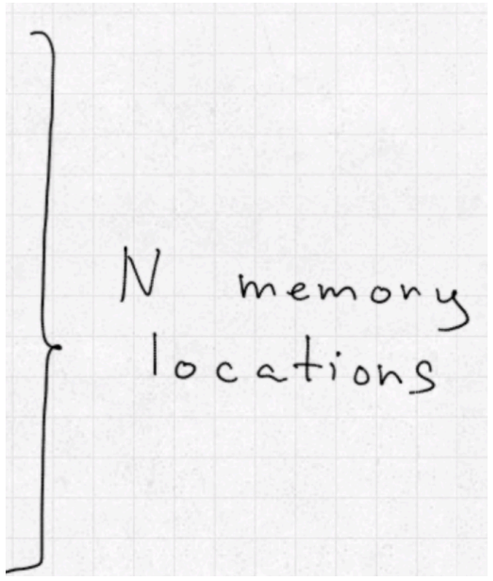
- Increase the amount of knowledge we add to the model while making minimal changes to the model itself

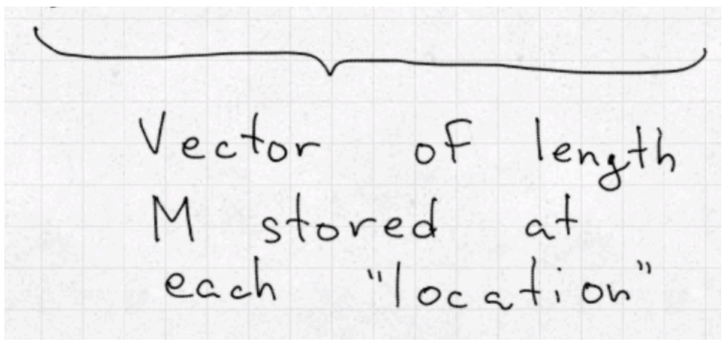
Memory Matrix



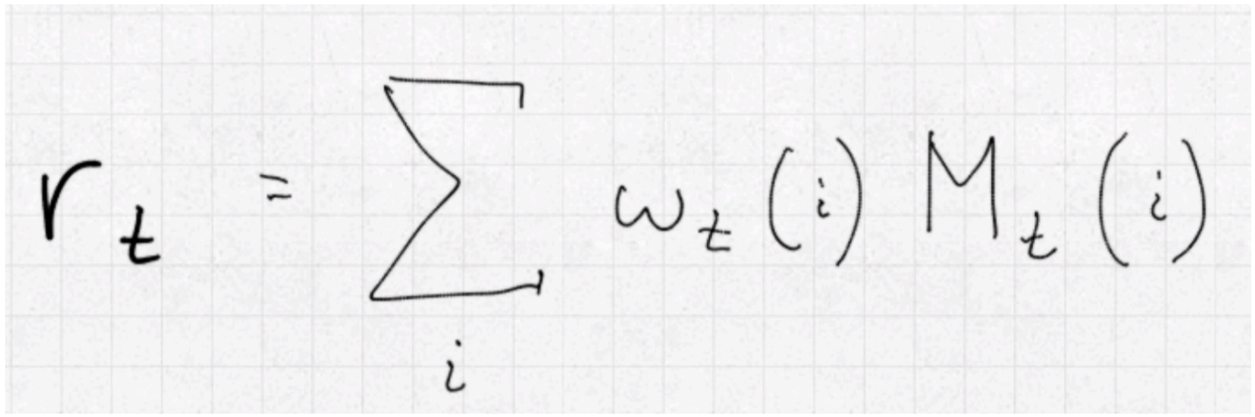


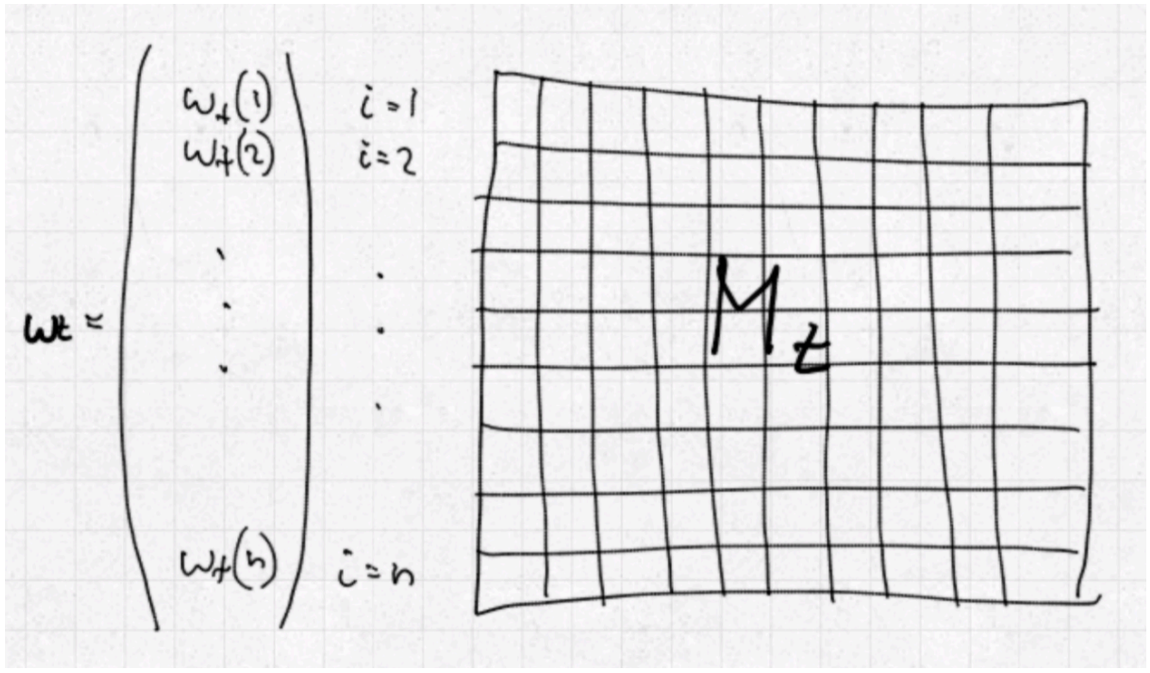






Reading





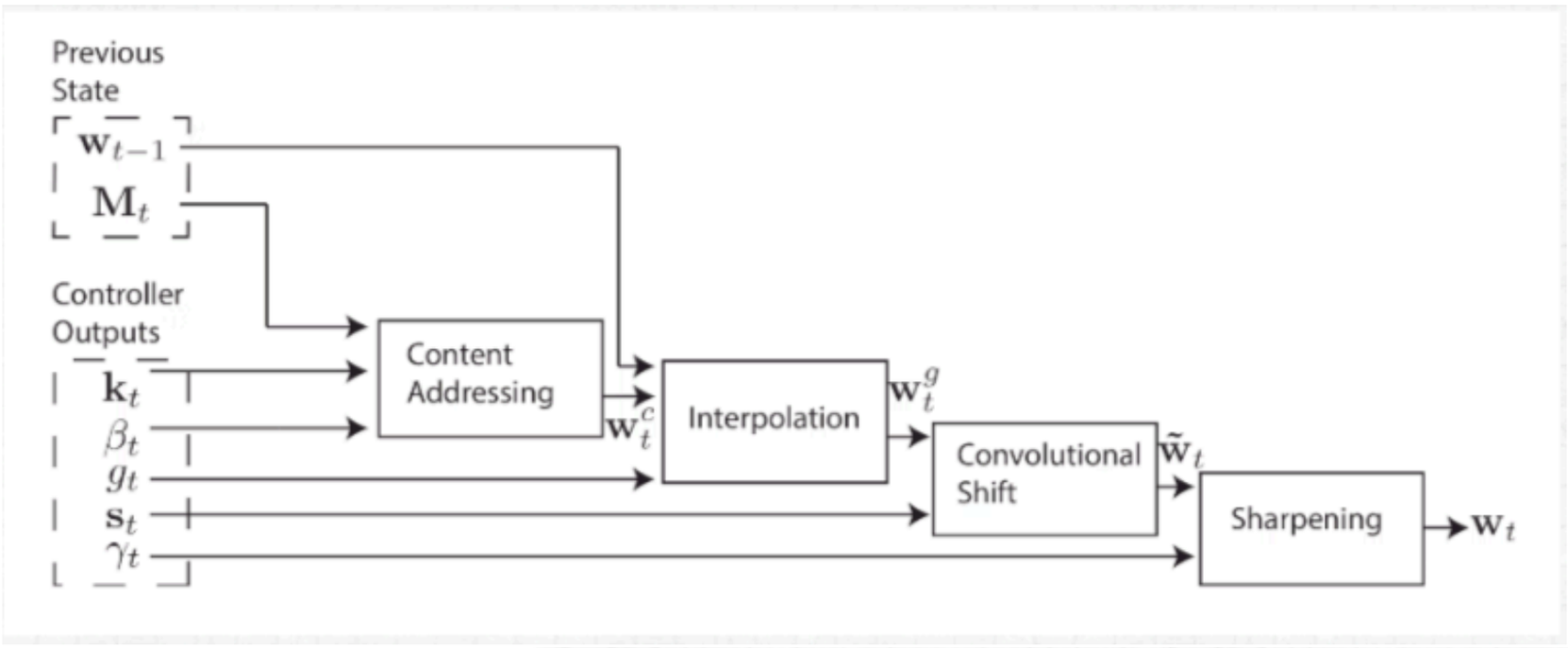
Addressing

Focus on content

① Reading or writing head before doing its job will produce vector

k_t

compare it to all $M_t(i)$ and output weightings



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

where

$$K[\mathbf{u}, \mathbf{v}] = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \cdot ||\mathbf{v}||}.$$

Writing

$$M_t(i) \leftarrow M_{t-1}(i) + w_t^w(i)k_t$$

Each row in memory is then updated using the write-weight vector and the key kt issued by the controller.

Kt is produced by controller and used for memory retrieval

Writing

Write to memory using Least Recent Used Access(LRUA)

Least: Do you use this knowledge often?

Recent: Do you just learn it?

Least Recent Used Access(LRUA)

$$w_t^u \leftarrow \gamma w_{t-1}^u + w_t^r + w_t^w$$

- Usage weights w_t^u keep track of the locations most recently read or written to

- gamma is the decay parameter

Least Recent Used Access(LRUA)

$$w_t^{lu}(i) = \begin{cases} 0 & \text{if } w_t^u(i) > m(\mathbf{w}_t^u, n) \\ 1 & \text{if } w_t^u(i) \leq m(\mathbf{w}_t^u, n) \end{cases}$$

- $m(\mathbf{w}_t^u, n)$ denotes the n smallest element of the w_t^u
- n: number of read

Least Recent Used Access(LRUA)

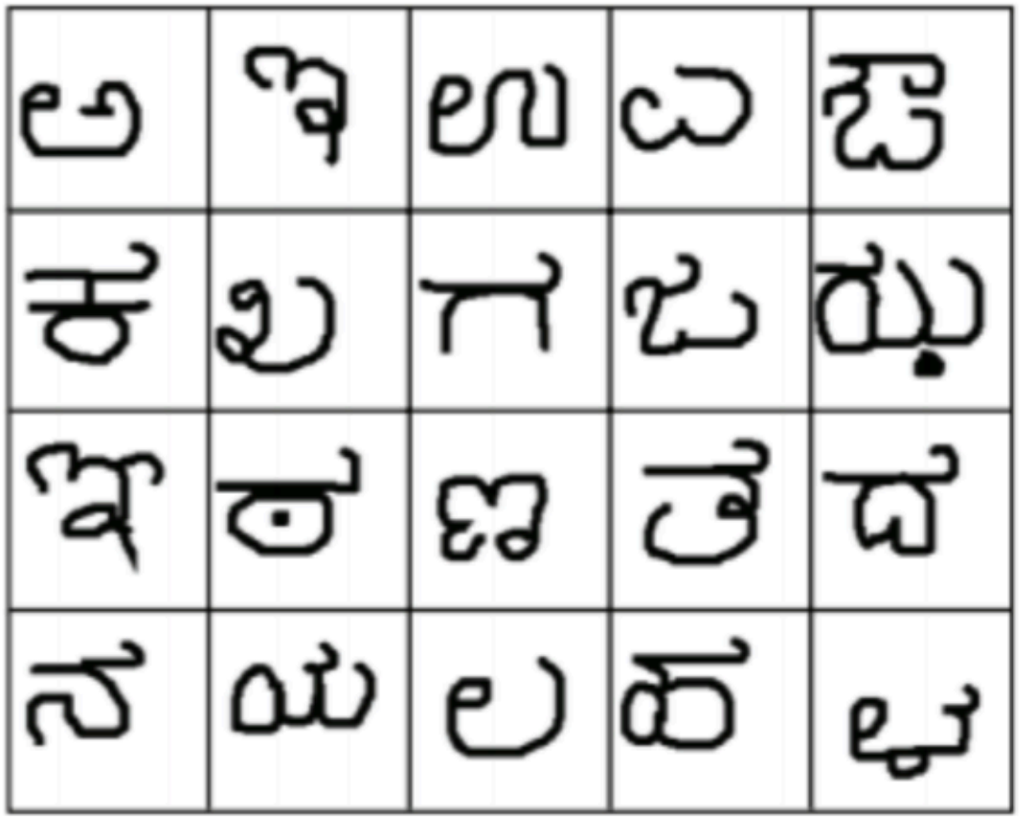
$$w_t^w \leftarrow \sigma(\alpha) w_{t-1}^r + (1 - \sigma(\alpha)) w_{t-1}^{lu}$$

$$M_t(i) \leftarrow M_{t-1}(i) + w_t^w(i) k_t$$

- alpha: learnable parameter
- Before writing to memory, the least used memory location is set zero

Experiments

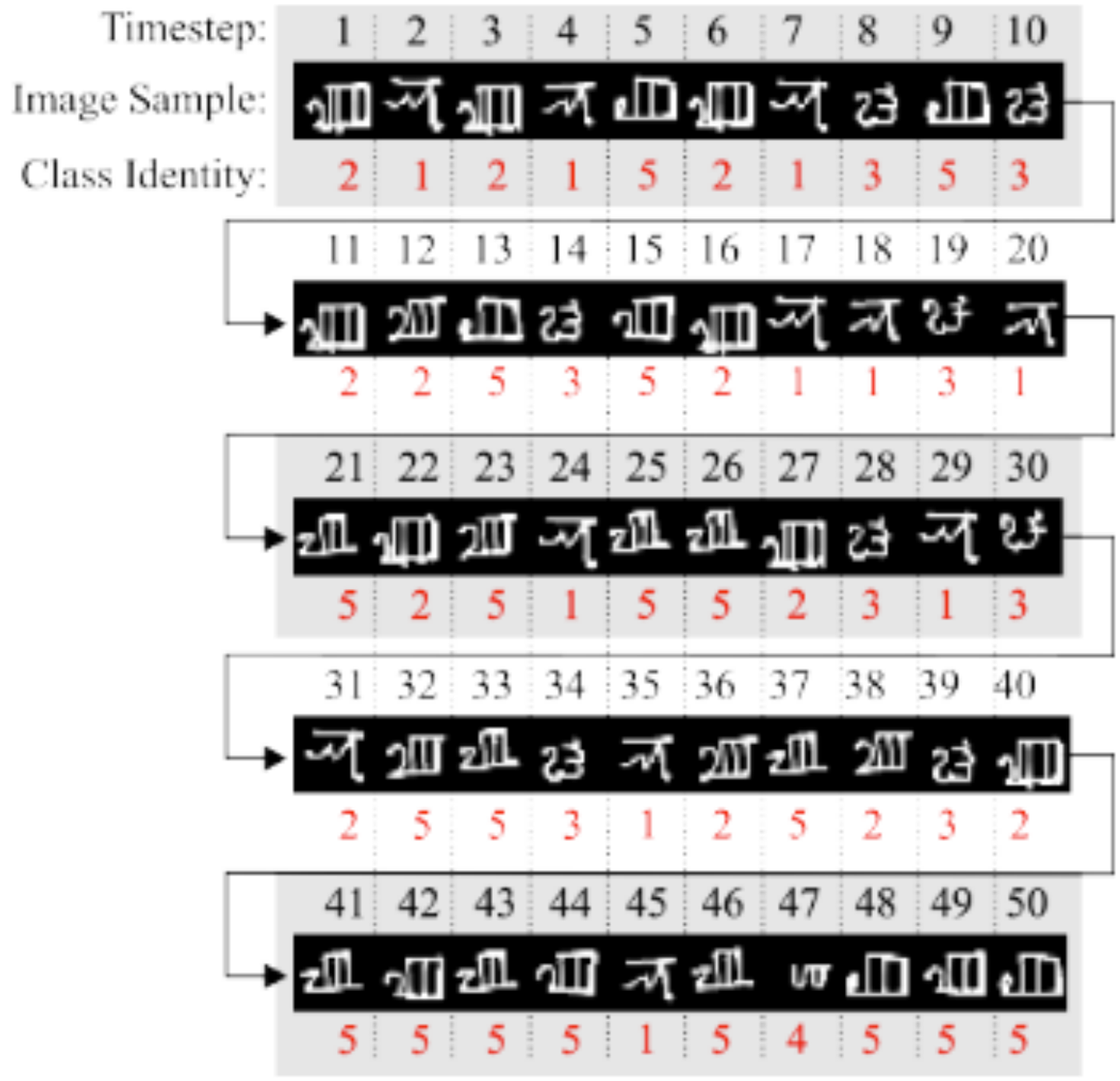
- Dataset: Omniglot (the transpose of the MNIST)
  - 1643 classes with only a few example per class
  - 1200 training classes
  - 443 test classes



Experiments

- Train for 100,000 episodes with five randomly chosen classes with five randomly chosen labels and 10 instances each

- Test on never-seen classes



Experiments

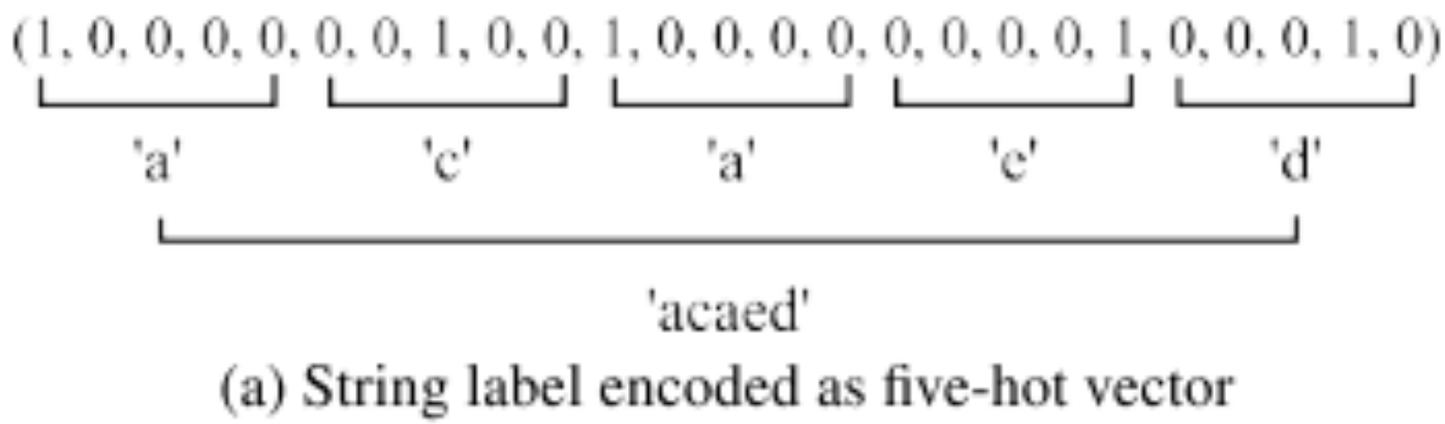
Table 1. Test-set classification accuracies for humans compared to machine algorithms trained on the Omniglot dataset, using one-hot encodings of labels and five classes presented per episode.

Model	Instance (% Correct)					
	1st	2nd	3rd	4th	5th	10th
Human	34.5	57.3	70.1	71.8	81.4	92.4
Feedforward	24.4	19.6	21.1	19.9	22.8	19.5
LSTM	24.4	49.5	55.3	61.0	63.6	62.5
MANN	**36.4**	**82.8**	**91.0**	**92.6**	**94.9**	**98.1**

Experiments

- A different approach for labeling classes was employed so that the number of classes presented in a given episode could be arbitrarily increased

- Characters for each label were uniformly sampled from the set {‘a’, ‘b’, ‘c’, ‘d’, ‘e’}, producing random strings such as ‘ecdba’

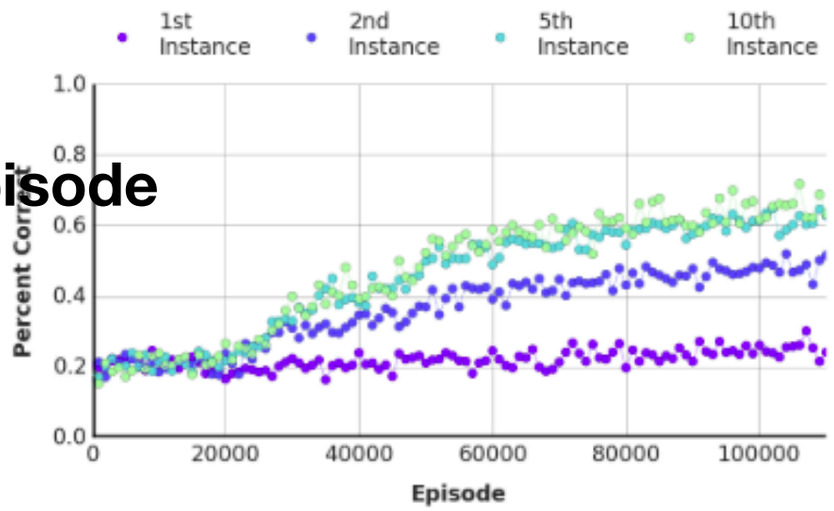


(a) String label encoded as five-hot vector

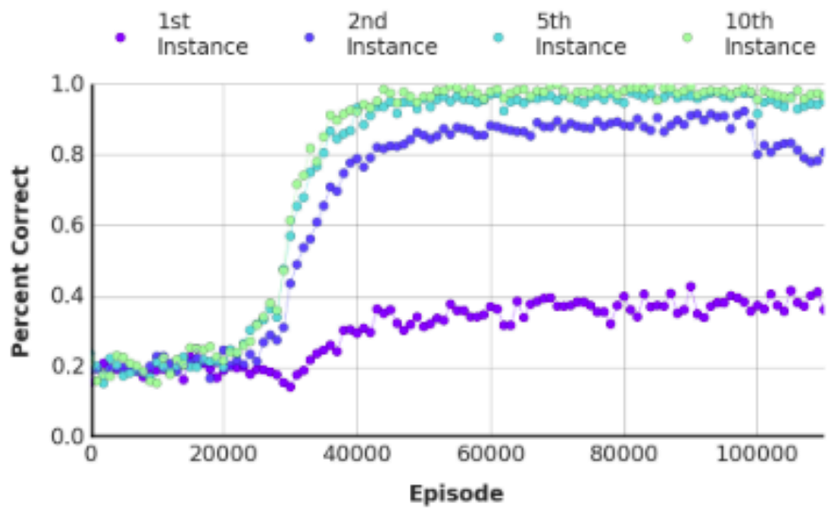
- This combinatorial approach allows for 3125 possible labels, which is nearly twice the number of classes in the dataset

Experiments

5 classes/episode

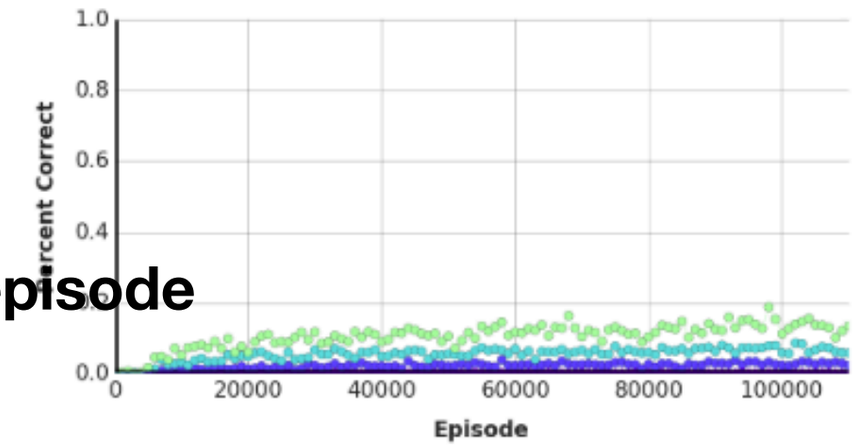


(a) LSTM, five random classes/episode, one-hot vector labels

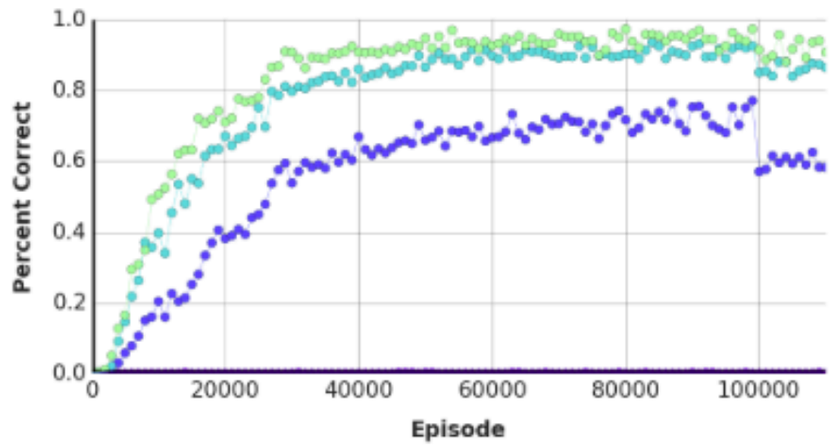


(b) MANN, five random classes/episode, one-hot vector labels

15 classes/episode



(c) LSTM, fifteen classes/episode, five-character string labels



(d) MANN, fifteen classes/episode, five-character string labels

Figure 2. Omniglot classification. The network was given either five (a-b) or up to fifteen (c-d) random classes per episode, which were of length 50 or 100 respectively. Labels were one-hot vectors in (a-b), and five-character strings in (c-d). In (b), first instance accuracy is above chance, indicating that the MANN is performing "educated guesses" for new classes based on the classes it has already seen and stored in memory. In (c-d), first instance accuracy is poor, as is expected, since it must make a guess from 3125 random strings. Second instance accuracy, however, approaches 80% during training for the MANN (d). At the 100,000 episode mark the network was tested, without further learning, on distinct classes withheld from the training set, and exhibited comparable performance.

Conclusion

- Gradual, incremental learning encodes background knowledge that spans tasks, while a more flexible memory resource binds information particular to newly encountered tasks(wipe out the external memory between episode in this experiment)

- Demonstrate the ability of a memory-augmented neural network to do meta-learning

- Introduce a new method access external memory

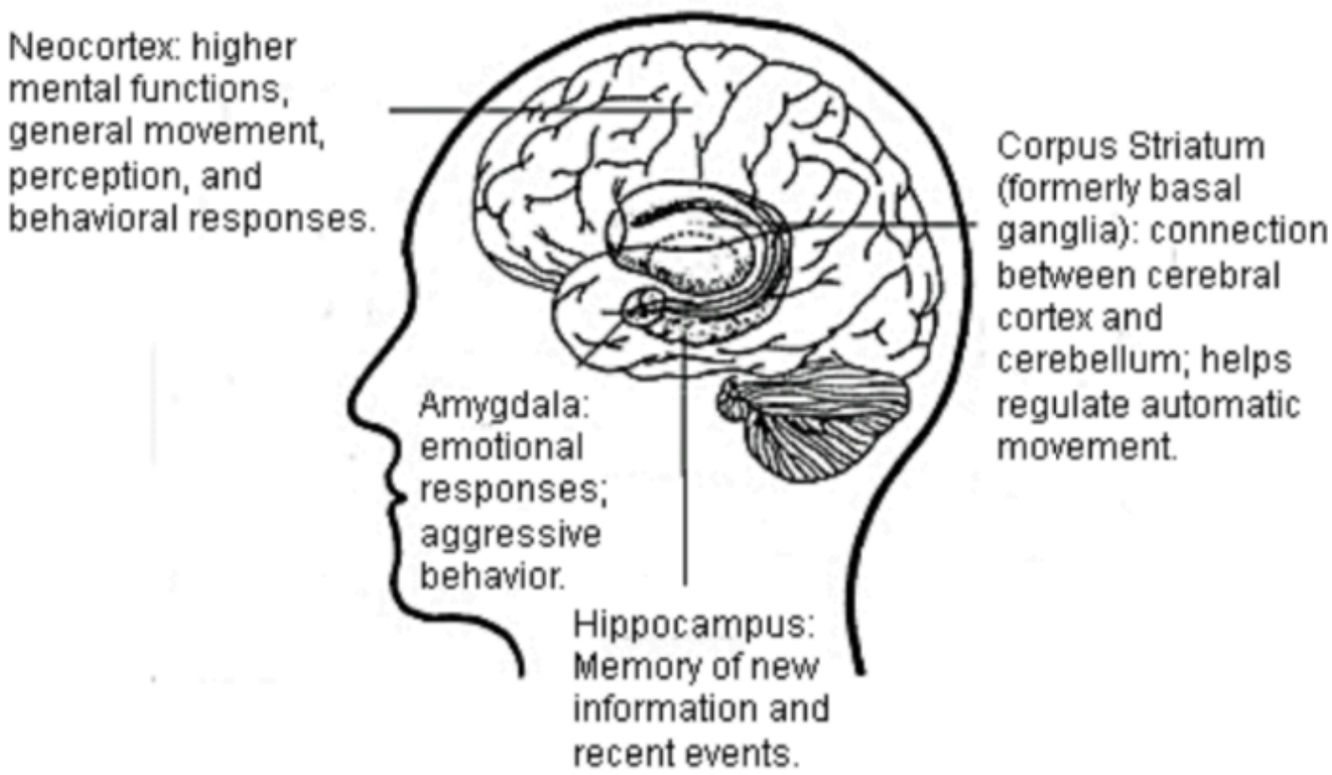
Conclusion

- It is acceptable for our learning algorithms to suffer from forgetting, but they may need complementary algorithms to reduce the information loss

Conclusion

- The controller is like the CPU/hippocampus, in charges of the long term memory

- The external memory is like the RAM/neocortical, in charges of the short term memory and the new coming information



Addition

- Information must be stored in memory in a representation that is both stable(reliably accessed when needed） and element-wise addressable（relevant pieces of information can be accessed selectively）

- Number of parameters shouldn’t be tied to the size of the memory