Neural Turing Machines

KLab group meeting 11/25


Big DATA
Coming 4U
SWAT
POLICE
Are You A
Risk To The State?

Neural Turing Machines

Alex Graves gravesa@google.com
Greg Wayne gregwayne@google.com
Ivo Danihelka danihelka@google.com

Google DeepMind, London, UK

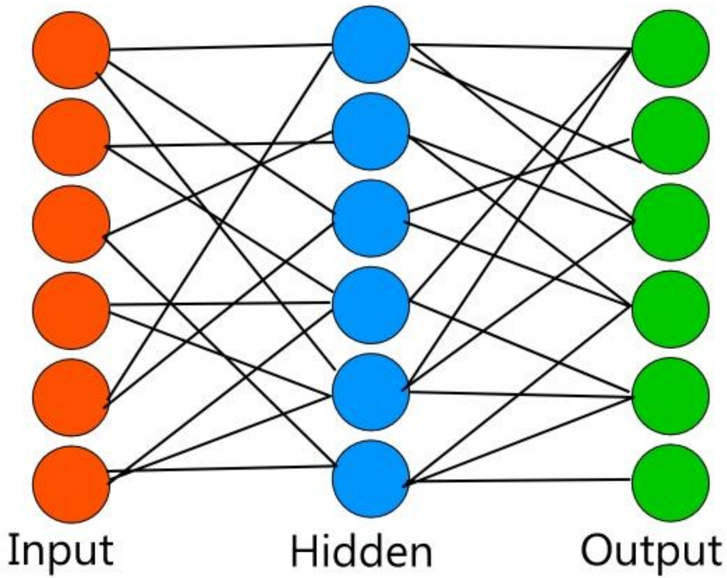
Goal: “Solve intelligence”

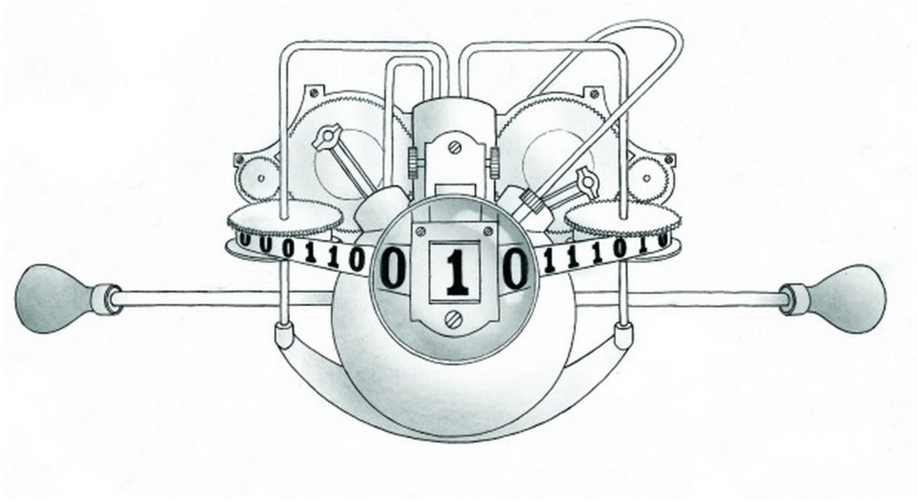
Price tag: *$400 million*

Abstract

We extend the capabilities of neural networks by coupling them to external memory resources, which they can interact with by attentional processes. The combined system is analogous to a Turing Machine or Von Neumann architecture but is differentiable end-to-end, allowing it to be efficiently trained with gradient descent. Preliminary results demonstrate that *Neural Turing Machines* can infer simple algorithms such as copying, sorting, and associative recall from input and output examples.

Building a Learning Machine

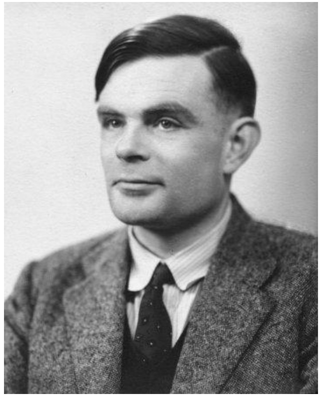




"Learning"
Input-Output mapping ~ rule

Formal model of solving a computational problem
rules + *memory*

Turing machine

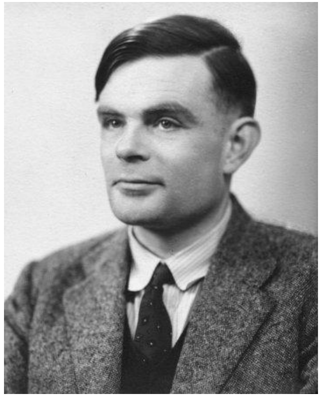


- What can be computed?

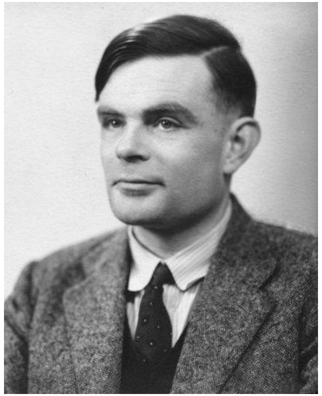
- Computability = instructions that lead to completion of task

http://plato.stanford.edu/entries/turing-machine

Turing machine

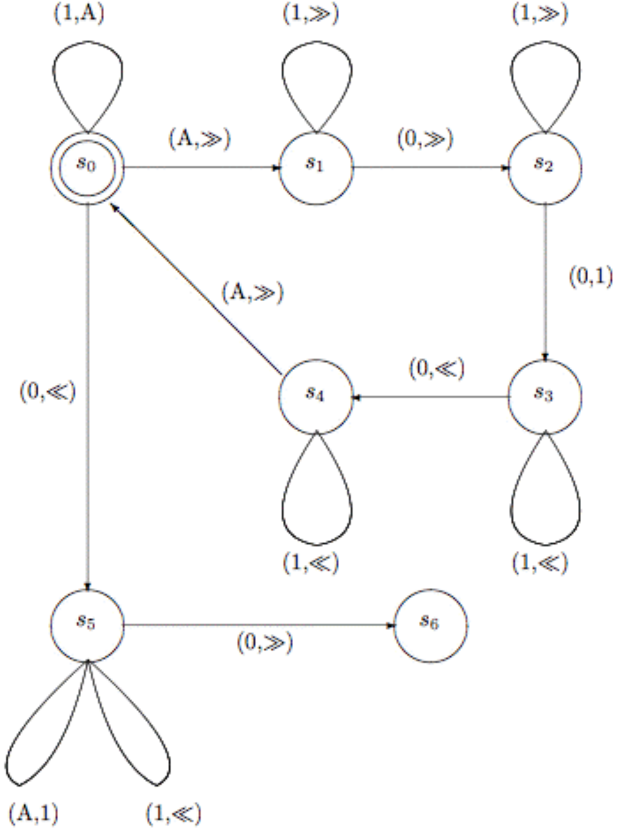


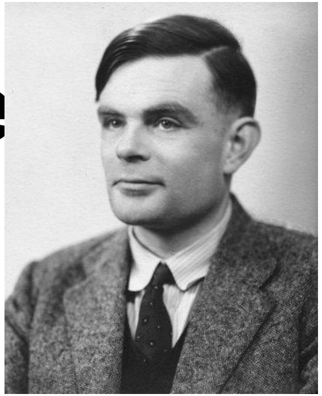
Turing machine



1. Tape (“memory”)
2. Read and write device (“head”)
3. Keeps track of current state (“state register”)
4. Instructions
    a. “If machine in state$_{current}$ and tape value is 0, go to state$_{next}$ and move left 1 space”

Turing machine - copy example

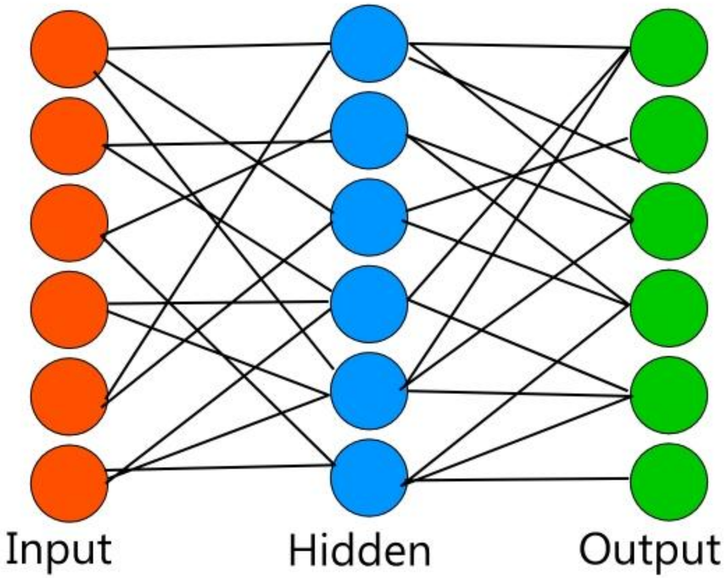


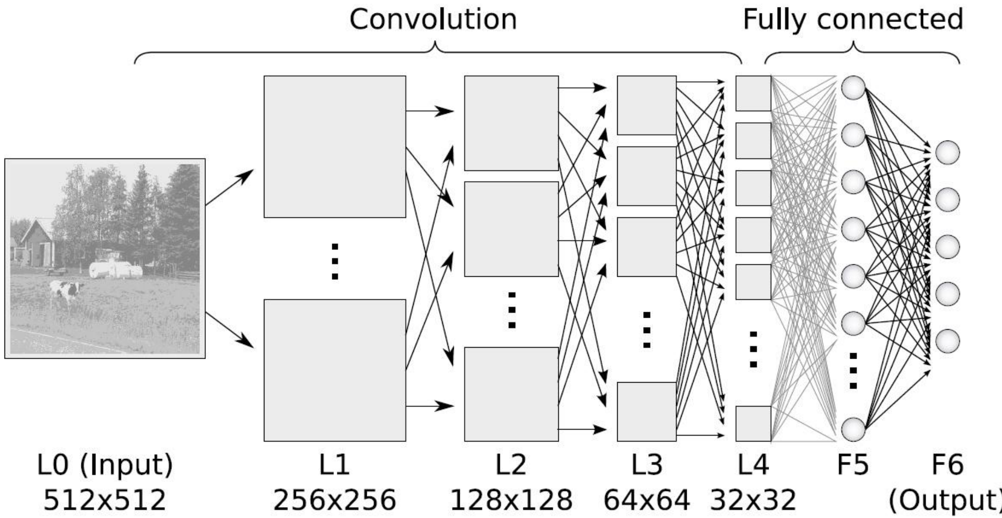


Neural networks

- 1950s - Perceptron
- 1969 - Proof that perceptron sucks! (Minsky)
- 1980s - Backpropagation
- 2000s-present - Fast computers, deep

Neural networks



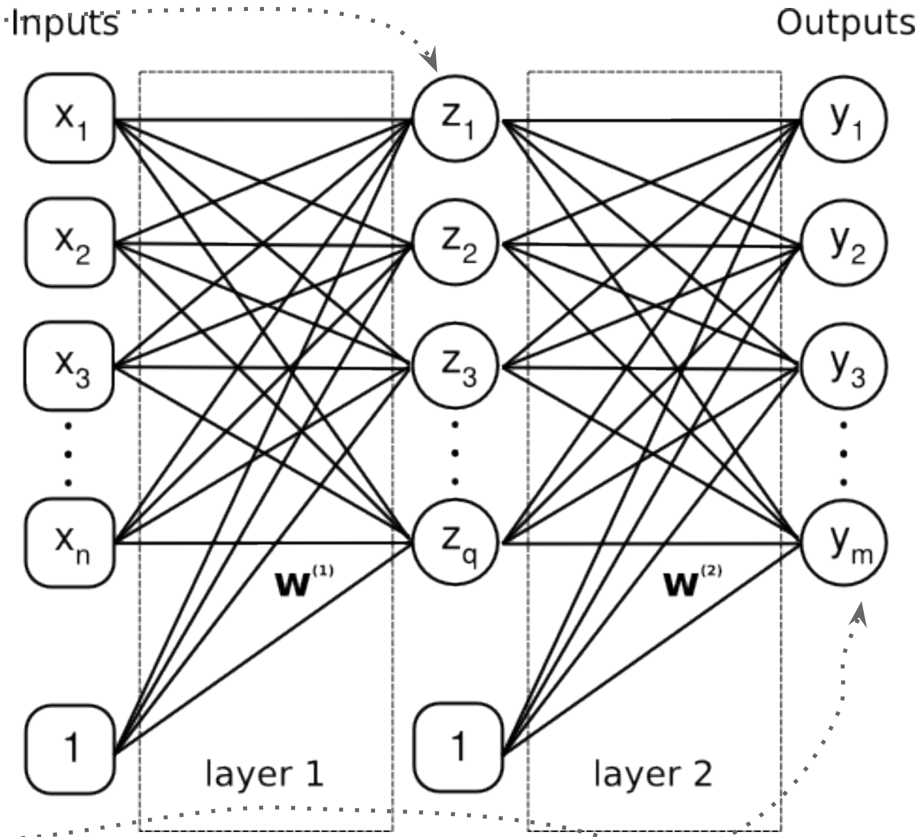


Feedforward neural networks

$$a_j = \sum_{i=1}^{n} w_{ji}^{(1)} x_i + w_{j0}^{(1)}$$

$$z_j = h(a) = \frac{1}{1 + \exp(-\kappa a)}$$

$$y_k = h(\sum_{j=1}^{q} w_{kj}^{(2)} z_j + w_{k0}^{(2)})$$

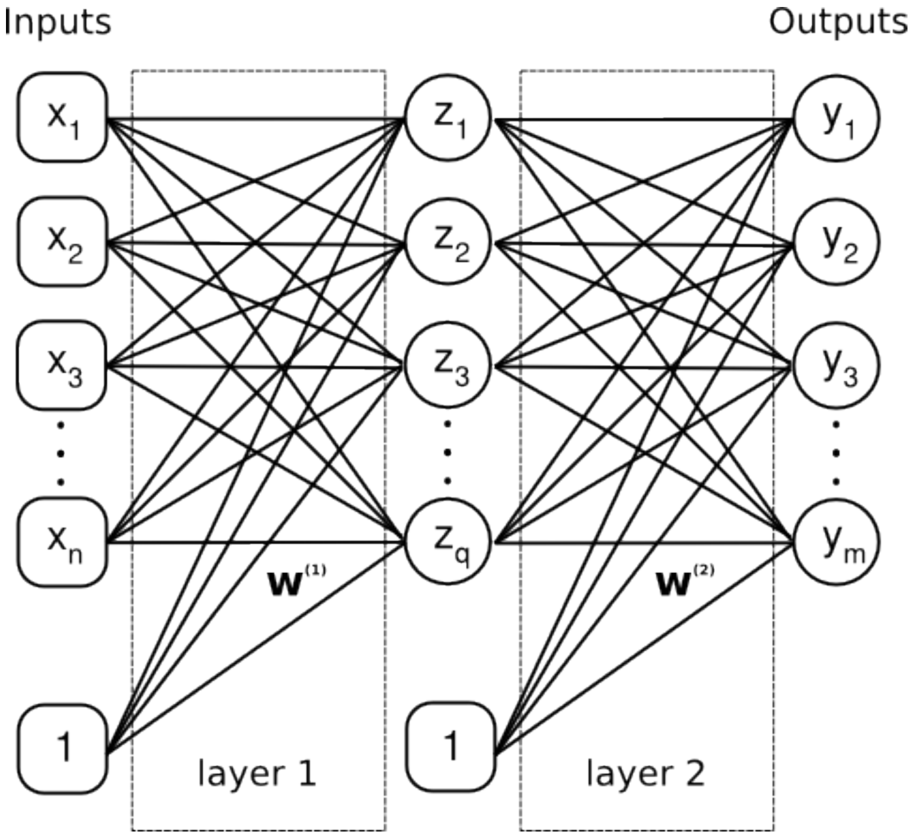


Feedforward neural networks

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}\{y(\mathbf{x}_n, \mathbf{w}) - t\}^2$$

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta\nabla E(\mathbf{w}^{(\tau)})$$

Feedforward neural networks



Feedforward neural networks

Backpropagation

Wanted: ∇E with respect to all weights; each $\dfrac{\partial E}{\partial w}$

$$\frac{\partial E}{\partial w_{kj}^{(2)}} = \frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial w_{kj}^{(2)}} = (y_k - t_k)z_j \equiv \delta_k z_j$$

For hidden layer -> output layer

$$\frac{\partial E}{\partial w_{ji}^{(1)}} = \frac{\partial E}{\partial a_j}\frac{\partial a_j}{\partial w_{ji}^{(1)}} \equiv \delta_j x_i \qquad \delta_j \equiv \frac{\partial E}{\partial a_j} = \sum_k \frac{\partial E}{\partial a_k}\frac{\partial a_k}{\partial a_j} = h'(a_j)\sum_k w_{kj}\delta_k$$

For input layer -> output layer

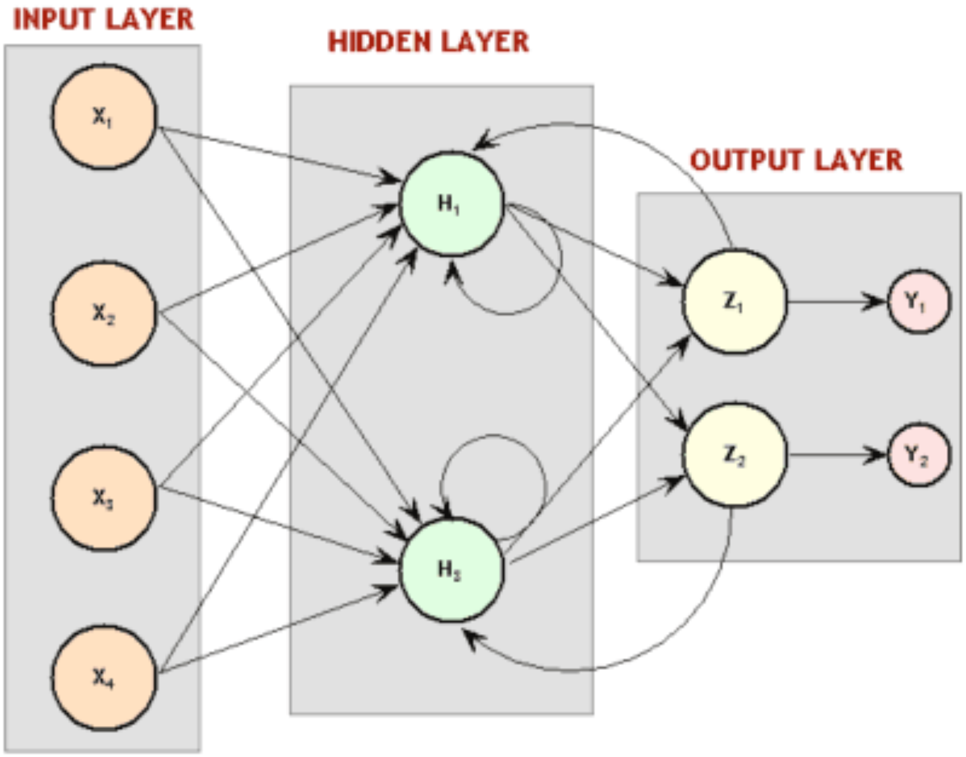
Feedforward neural networks

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E(\mathbf{w}^{(\tau)})$$

Gradient descent

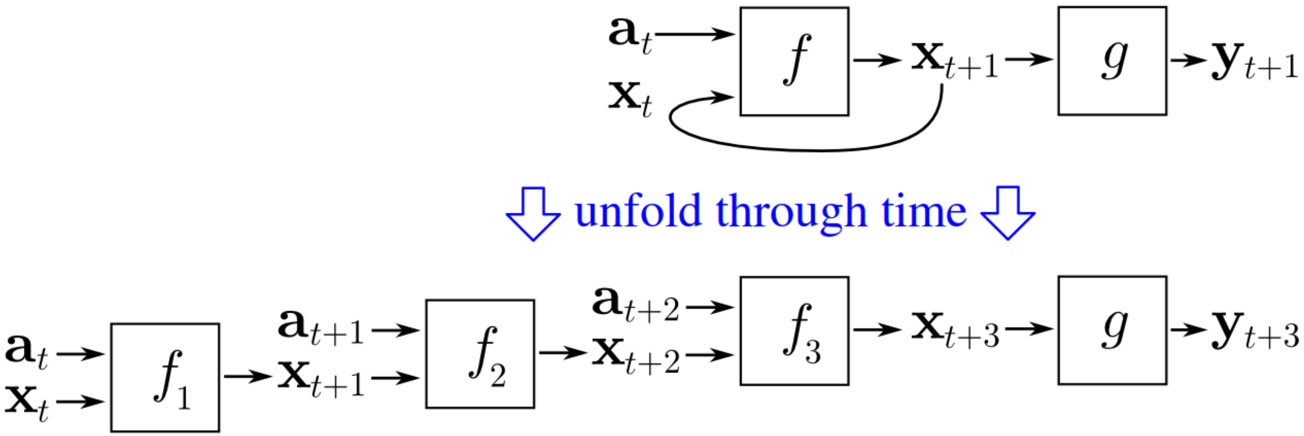
- Classic
- Conjugate gradient descent
- Stochastic gradient descent

Recurrent neural networks



Recurrent neural networks

Backpropagation through time

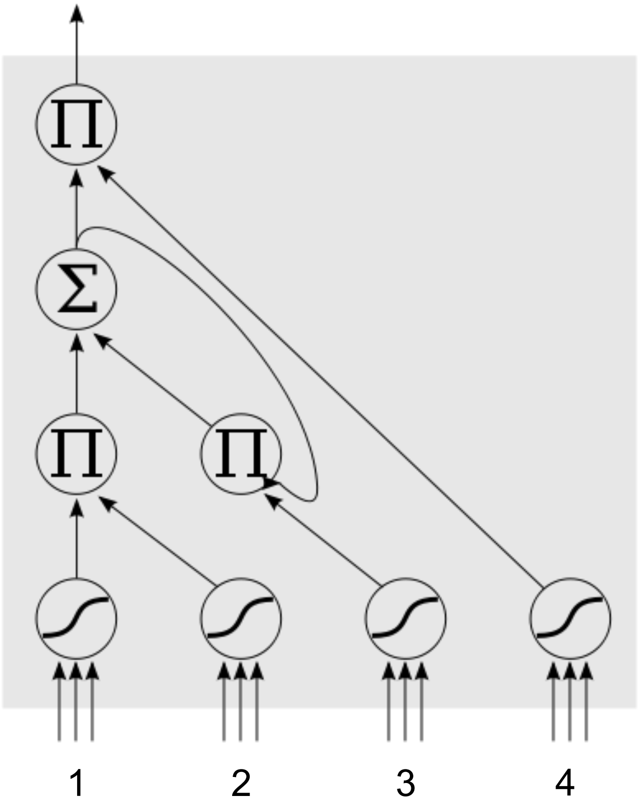


Problem: Vanishing/Exploding gradients!

Recurrent neural networks

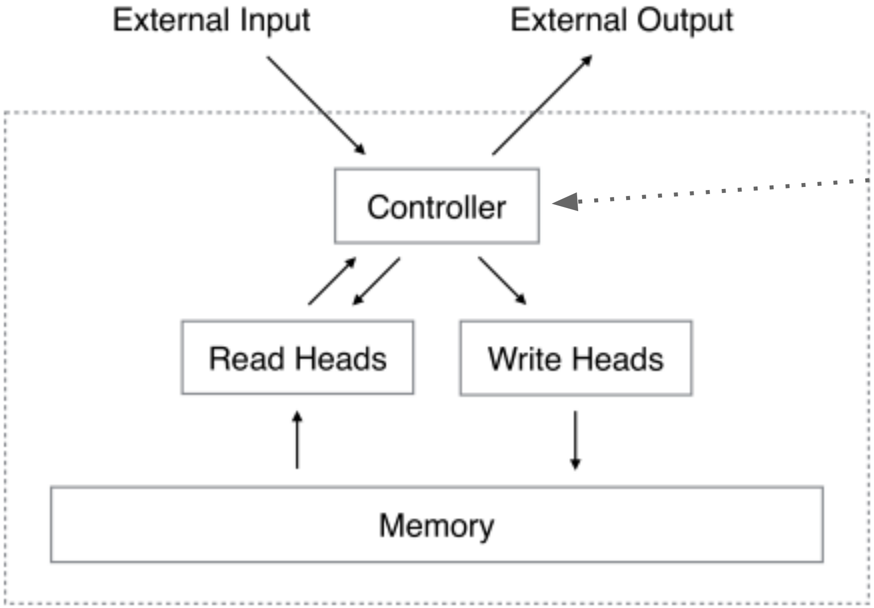
Long Short-Term Memory

1. Input
2. Input gate
3. “Remember” gate
4. Output gate



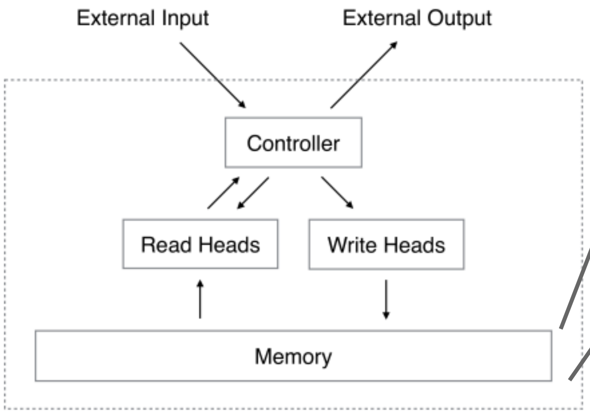
Somewhat complicated, lots of parameters

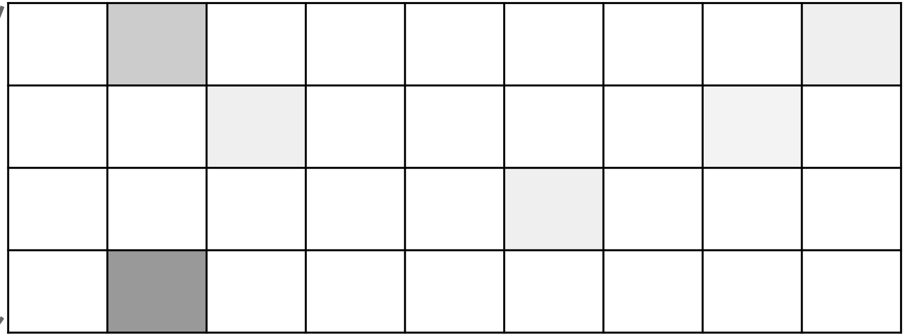
Neural Turing Machines



Neural Network

Neural Turing Machines - memory



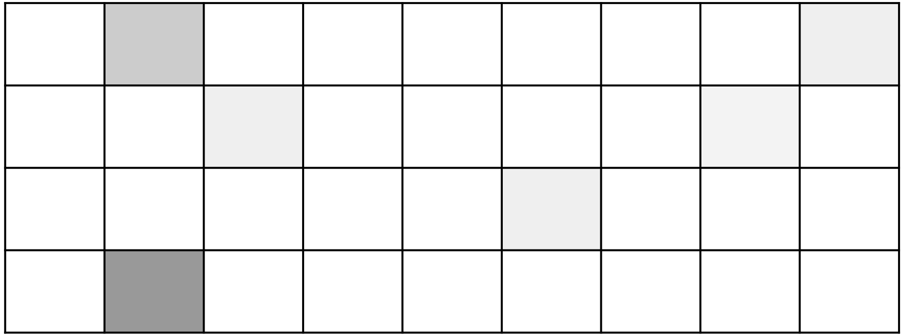


N chunks (rows) X M bits each (columns)

Neural Turing Machines - memory

Read from memory ("blurry")

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



N chunks (rows) X M bits each (columns)

Write to memory ("blurry")

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

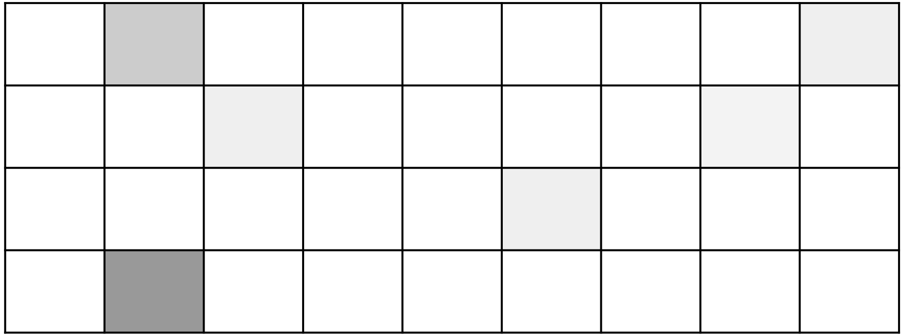
$$\mathbf{M}_t(i) \longleftarrow \tilde{\mathbf{M}}_t(i) + w_t(i)\,\mathbf{a}_t.$$

Neural Turing Machines - memory

Addressing by content (similarity)

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

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



N chunks (rows) X M bits each (columns)

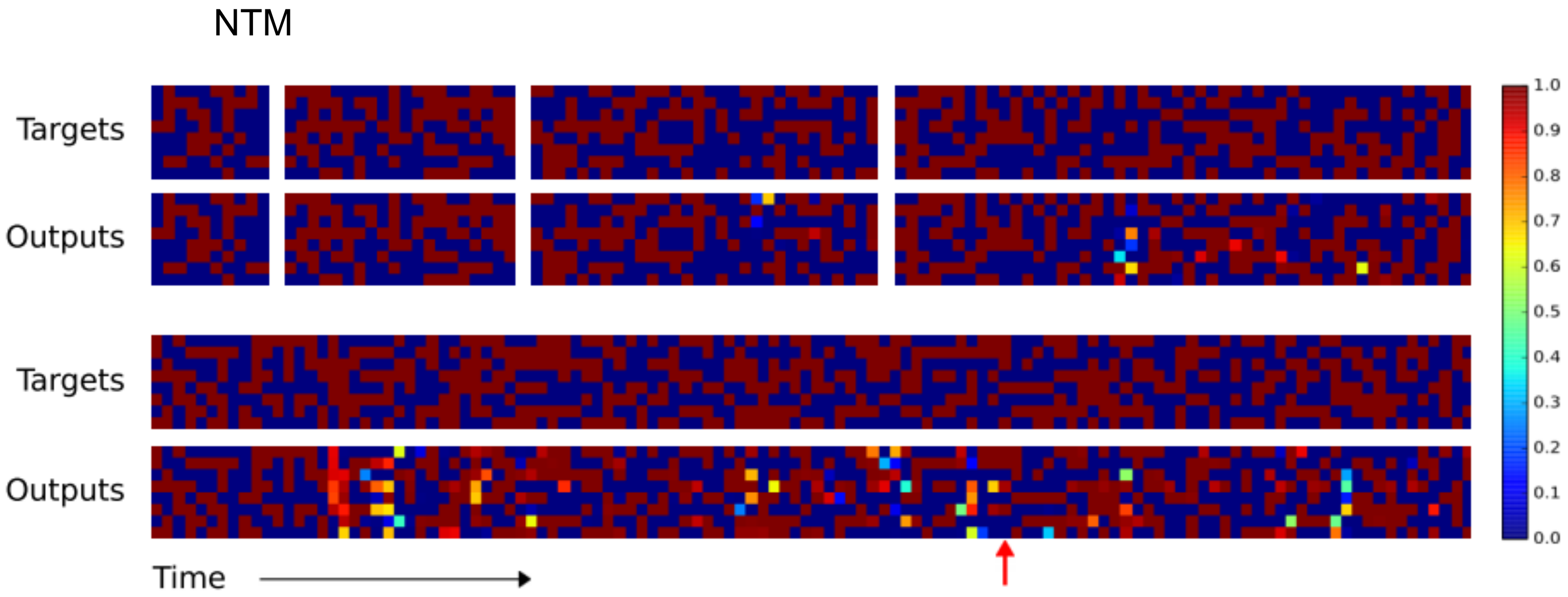
Addressing by location (shift)

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

Neural Turing Machines - examples

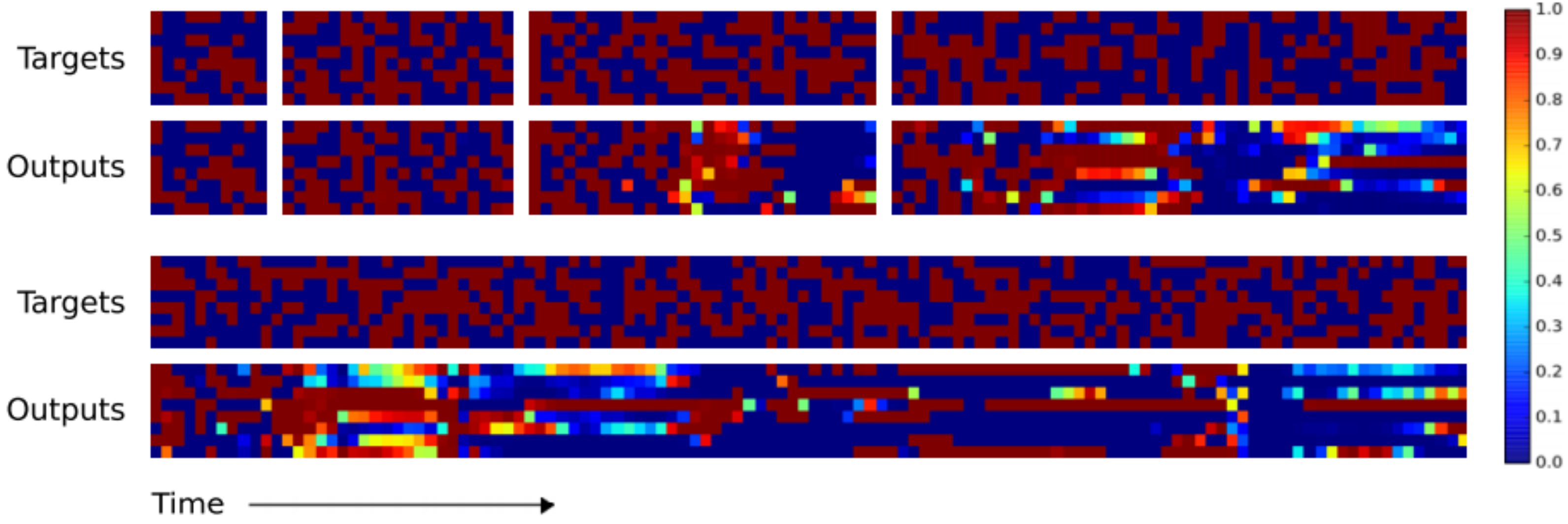
- NTM can learn to do basic things
  - Copy
  - Associative recall
  - N-gram lookup
  - Sorting
- Better than LSTM alone

Neural Turing Machines - copy

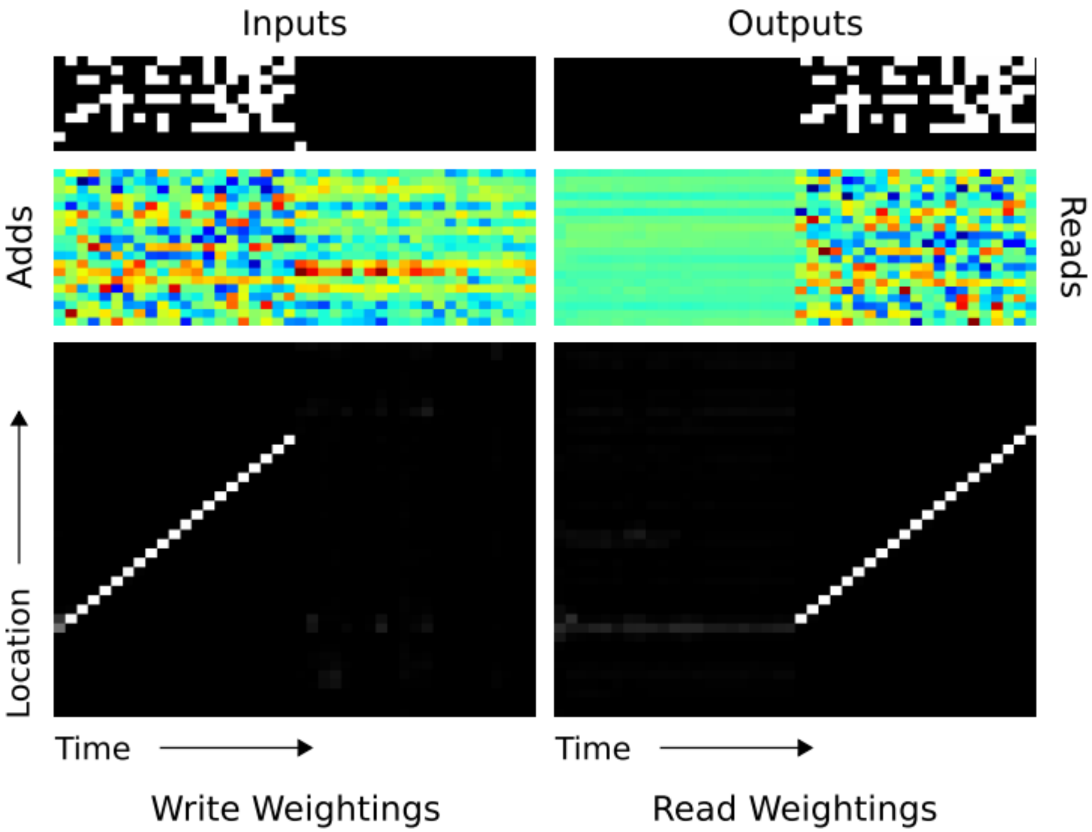


Neural Turing Machines - copy

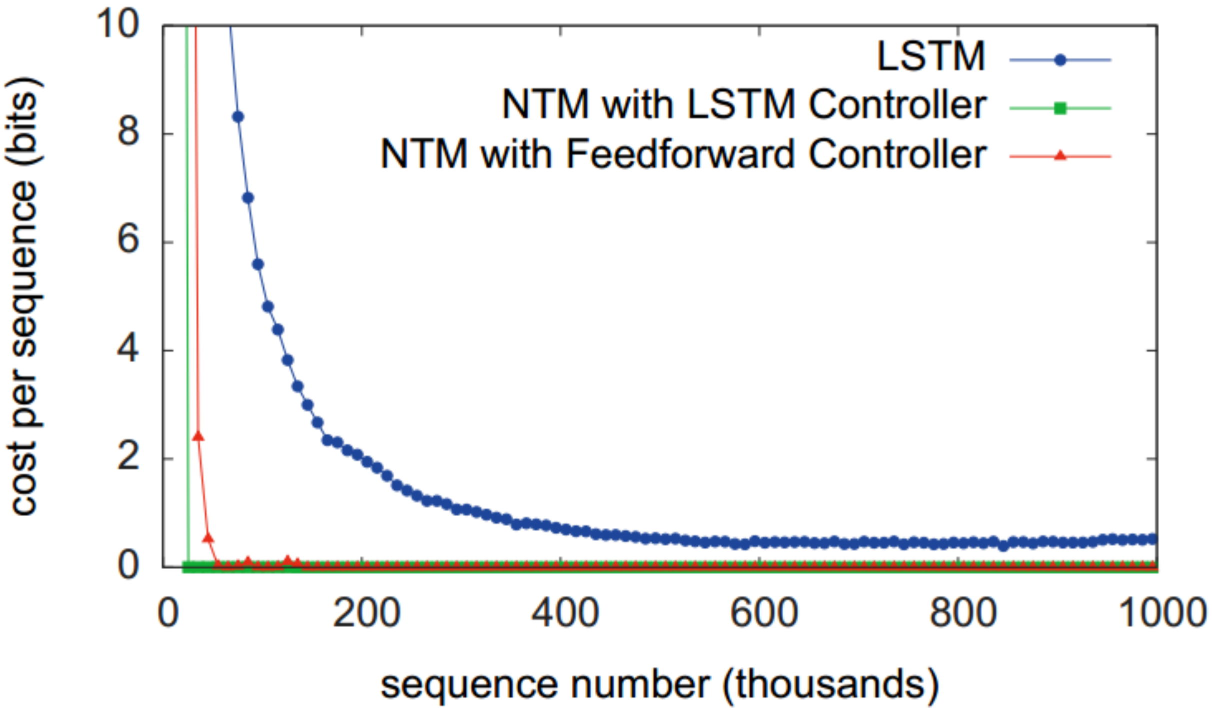
LSTM



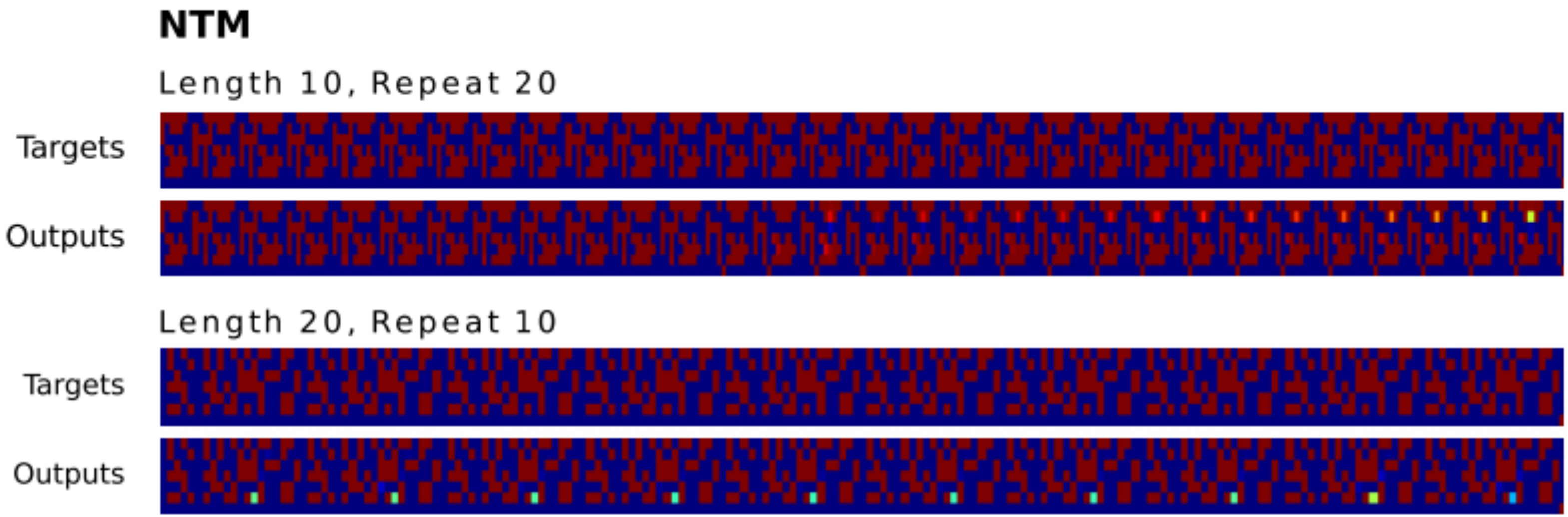
Neural Turing Machines - copy



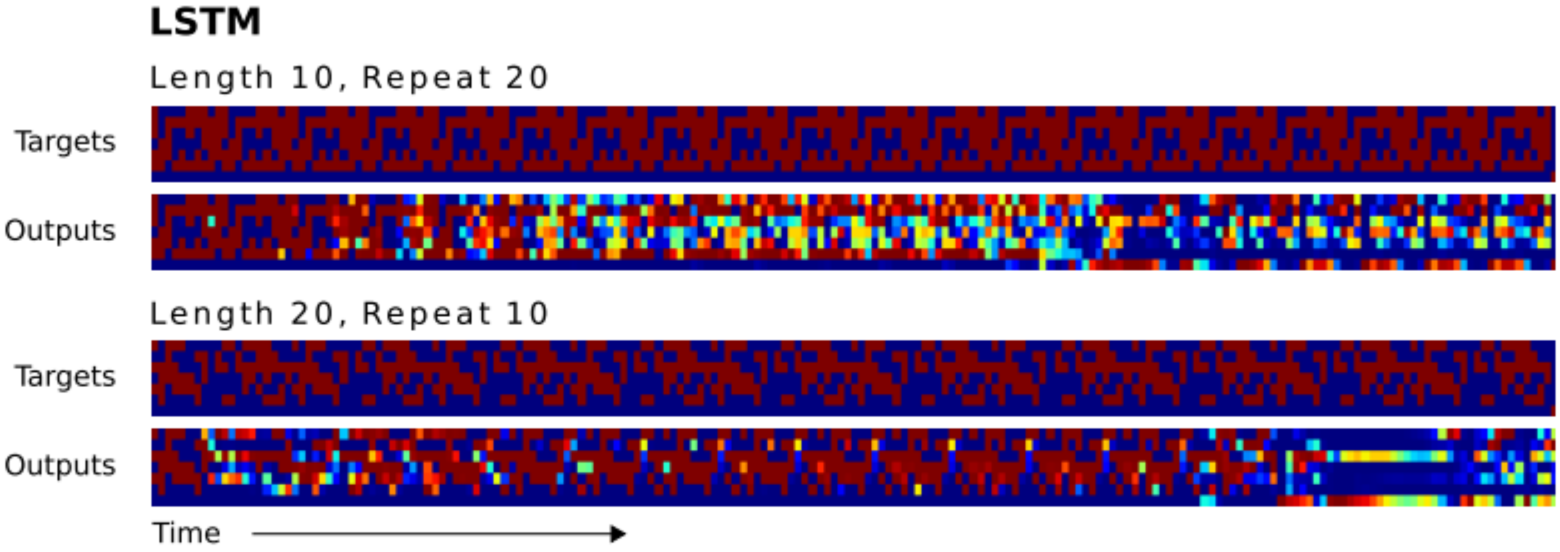
Neural Turing Machines - copy



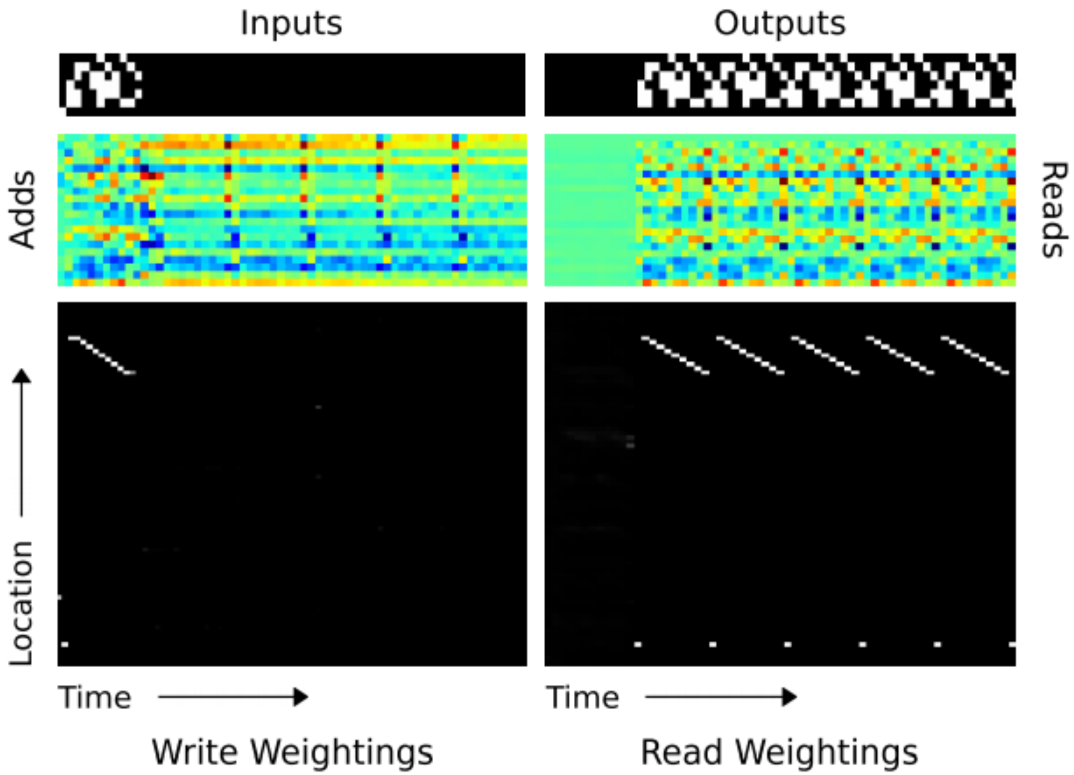
Neural Turing Machines - mult copy



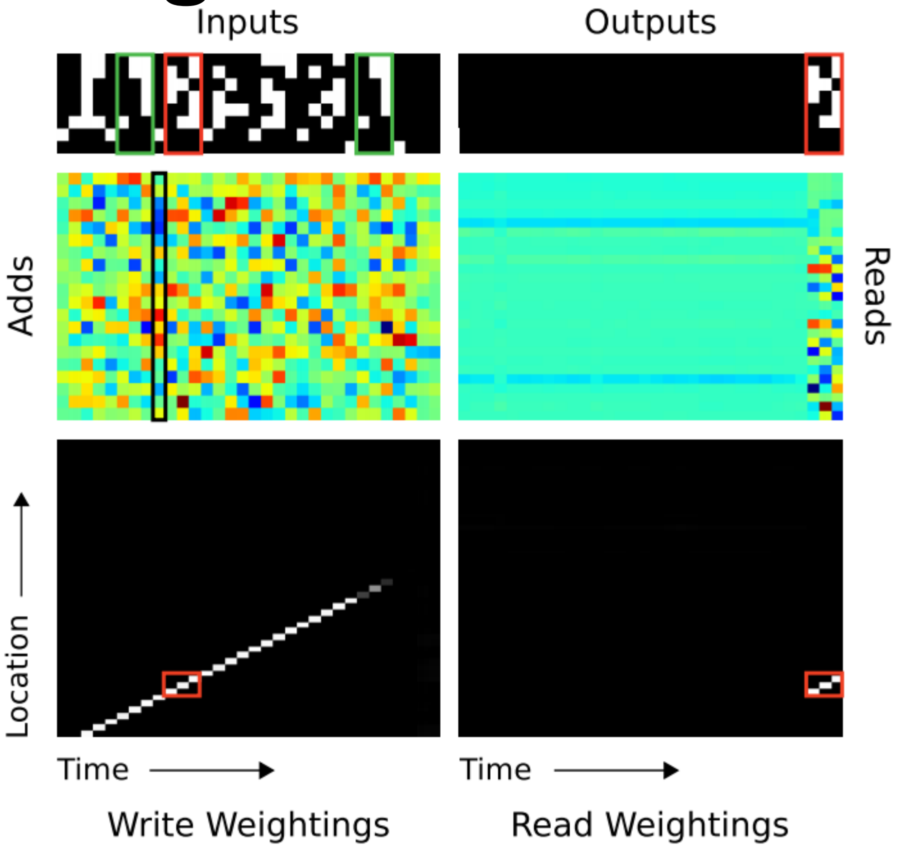
Neural Turing Machines - mult copy



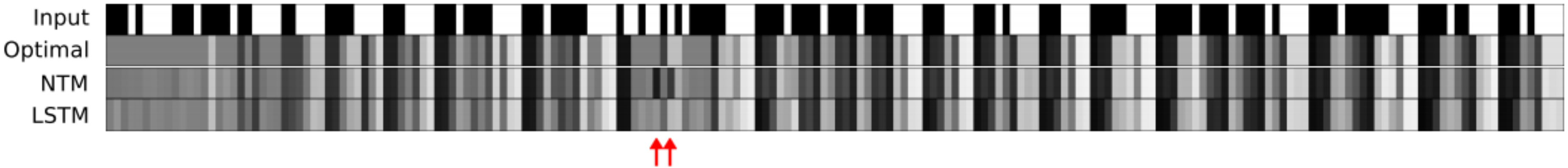
Neural Turing Machines - mult. copy



Neural Turing Machines - ass. recall



Neural Turing Machines - examples



Neural Turing Machines - examples