Article: **Hybrid Computing Using a Neural Network with Dynamic External Memory**

Speaker: Valeria Bubnova

Prepare to me amazed…

1) What’s the idea?
2) Does it even work?
3) Details (you’ll need them!)

Neural Networks

Store weights

Learn from data

Learn some distributions

Weak with algorithms & data structures

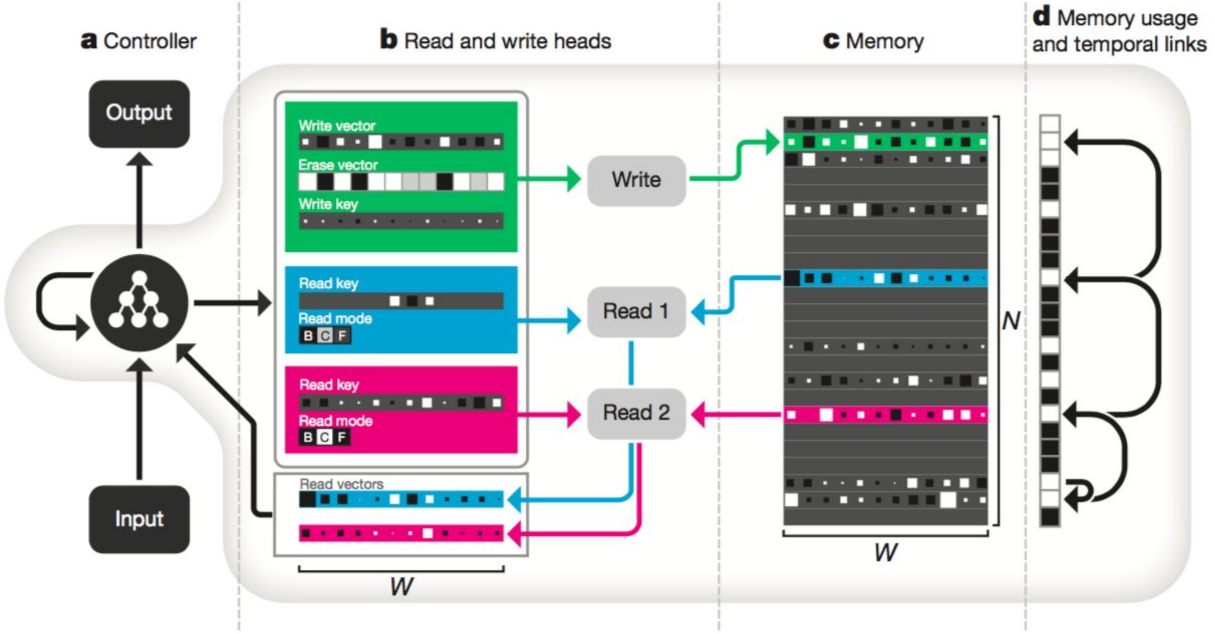
Neural Networks	Computers
Store weights	Processor and RAM
Learn from data	Passively store data
Learn some distributions	All data stored equally
Weak with algorithms & data structures	Can process data structures

Differentiable Neural Computers

Store weights	~~Processor and RAM~~ Controller and Memory matrix
Learn from data	~~Passively~~ store data
Learn some distributions	~~All data stored equally~~ Data is stored with some distribution
~~Weak with algorithms & data structures~~	Can process data structures

#NeuralTuringMachine

How?

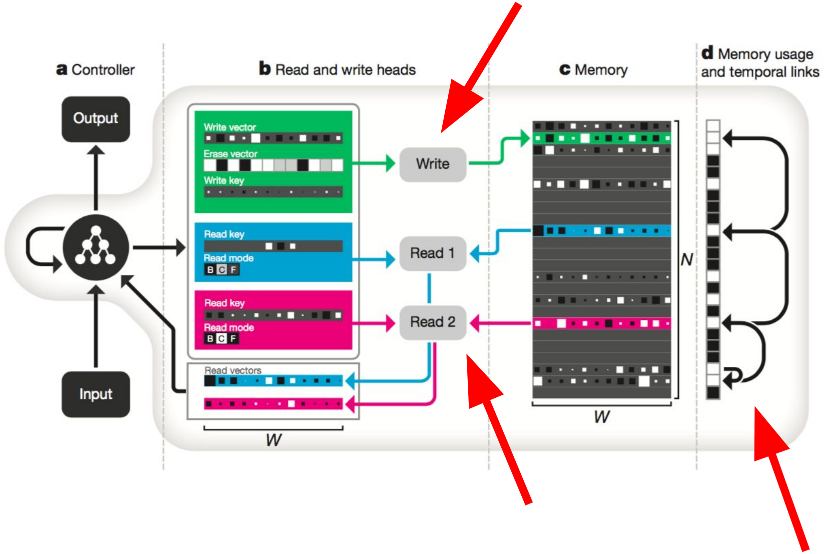

a Controller
b Read and write heads
c Memory
d Memory usage and temporal links
Output
Input
Write vector
Erase vector
Write key
Read key
Read mode
B C F
Read key
Read mode
B C F
Read vectors
Write
Read 1
Read 2
N
W
W

Differentiable everything!

Yes, and memory as well....

3 forms of <differentiable> attention

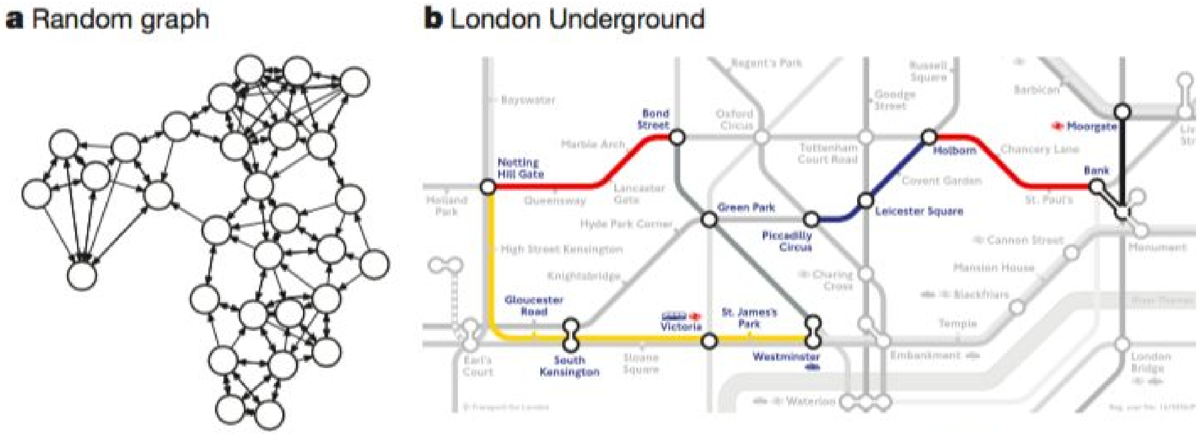
- Content Lookup
- Temporal Linking
- Memory Allocation



#Hippocampus
#CA3_synapses
#CA1_synapses

3 forms of <differentiable> attention

Attention mechanism	Computational considerations
Content Lookup	Formation of associative data structures
Temporal Linking	Sequential retrieval of input sequences
Memory Allocation	Provides the write head with unused locations



Underground input	Traversal	Shortest-path

Underground input:
(OxfordCircus, TottenhamCtRd, Central)
(TottenhamCtRd, OxfordCircus, Central)
(BakerSt, Marylebone, Circle)
(BakerSt, Marylebone, Bakerloo)
(BakerSt, OxfordCircus, Bakerloo)
⋮
(LeicesterSq, CharingCross, Northern)
(TottenhamCtRd, LeicesterSq, Northern)
(OxfordCircus, PiccadillyCircus, Bakerloo)
(OxfordCircus, NottingHillGate, Central)
(OxfordCircus, Euston, Victoria)

84 edges in total

Traversal

Traversal question:
(BondSt, _, Central),
(_, _, Circle), (_, _, Circle),
(_, _, Circle), (_, _, Circle),
(_, _, Jubilee), (_, _, Jubilee),

Answer:
(BondSt, NottingHillGate, Central)
(NottingHillGate, GloucesterRd, Circle)
⋮
(Westminster, GreenPark, Jubilee)
(GreenPark, BondSt, Jubilee)

Shortest-path

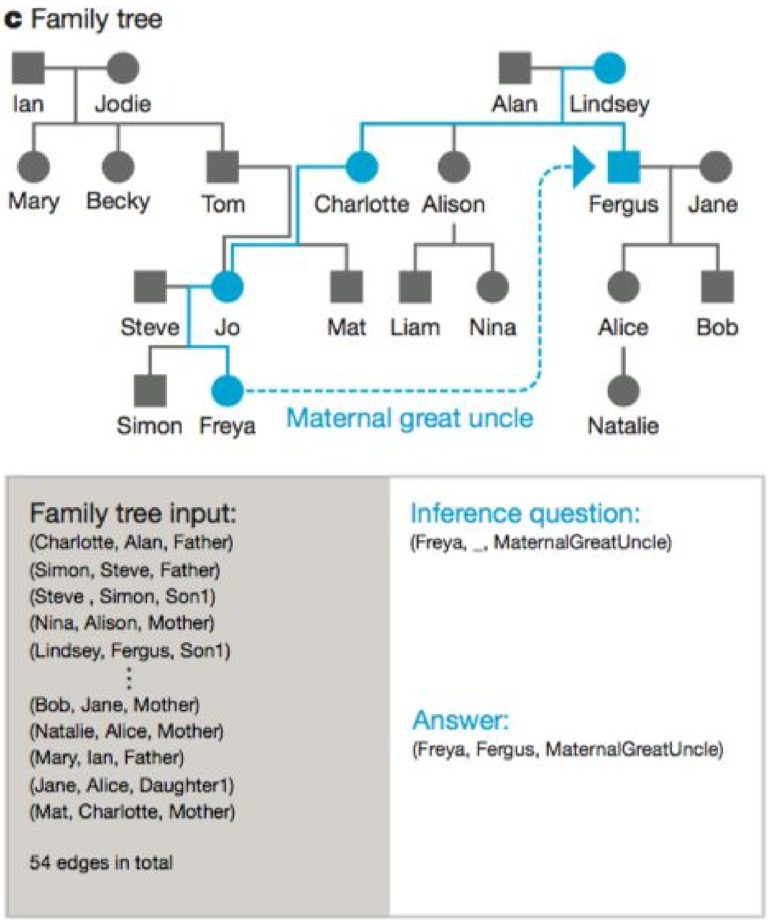
Shortest-path question:
(Moorgate, PiccadillyCircus, _)

Answer:
(Moorgate, Bank, Northern)
(Bank, Holborn, Central)
(Holborn, LeicesterSq, Piccadilly)
(LeicesterSq, PiccadillyCircus, Piccadilly)

LSTM: 37% accuracy after 2 million examples

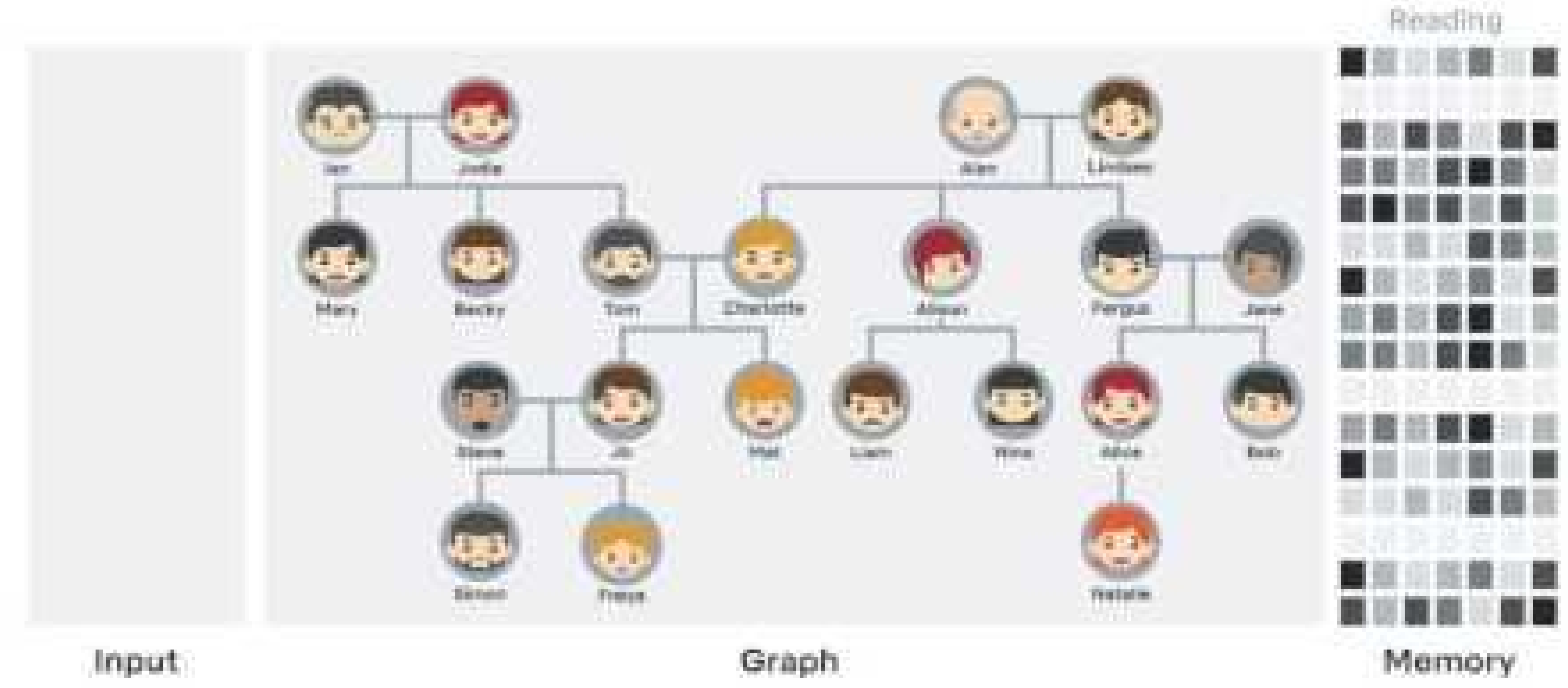
DNC: 98.8% accuracy after 1 million examples

Results



DNC: 81.8% accuracy
on four-step relations

Results

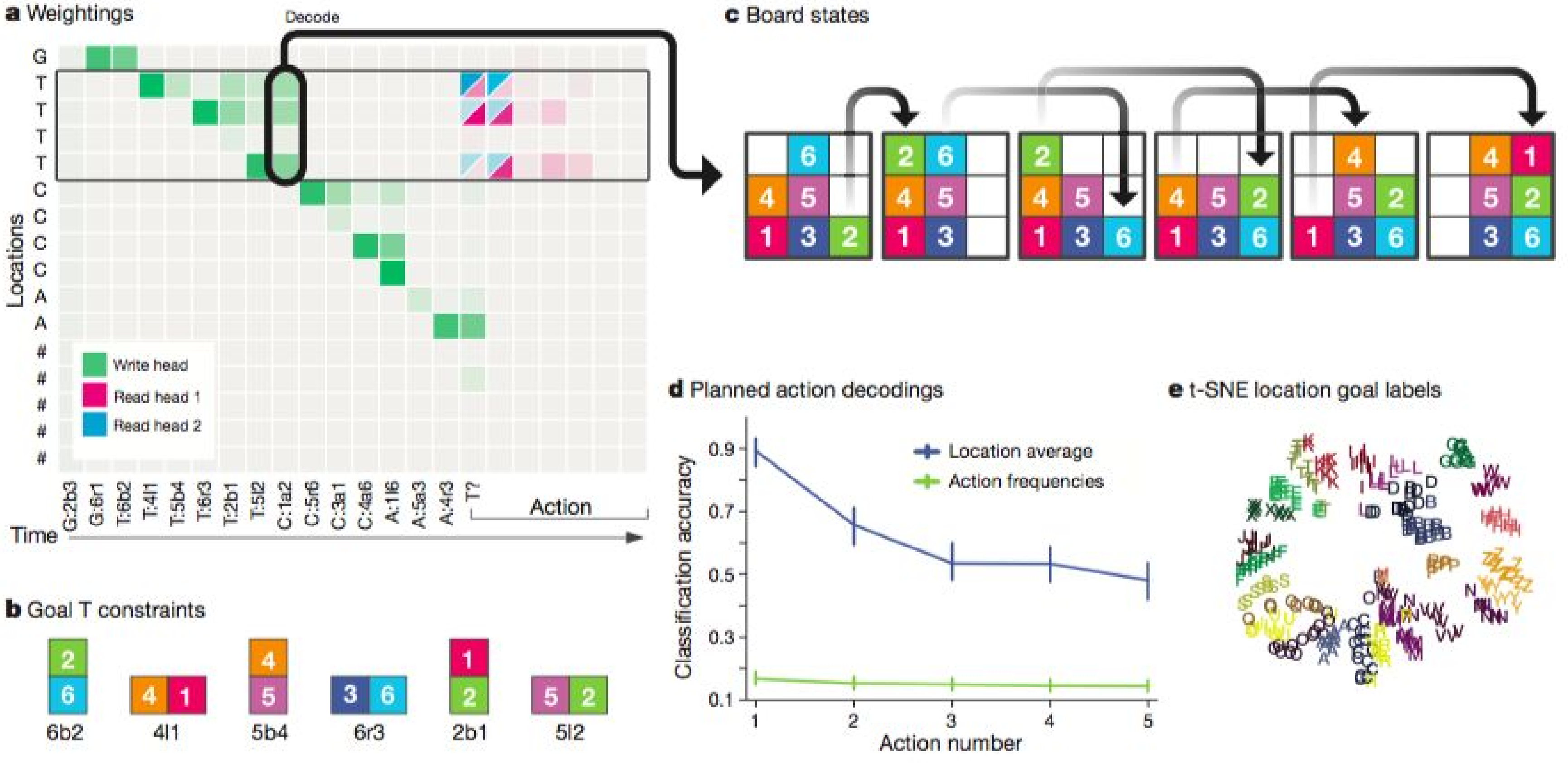


Sheep are afraid of wolves.
Gertrude is a sheep.
Mice are afraid of cats.

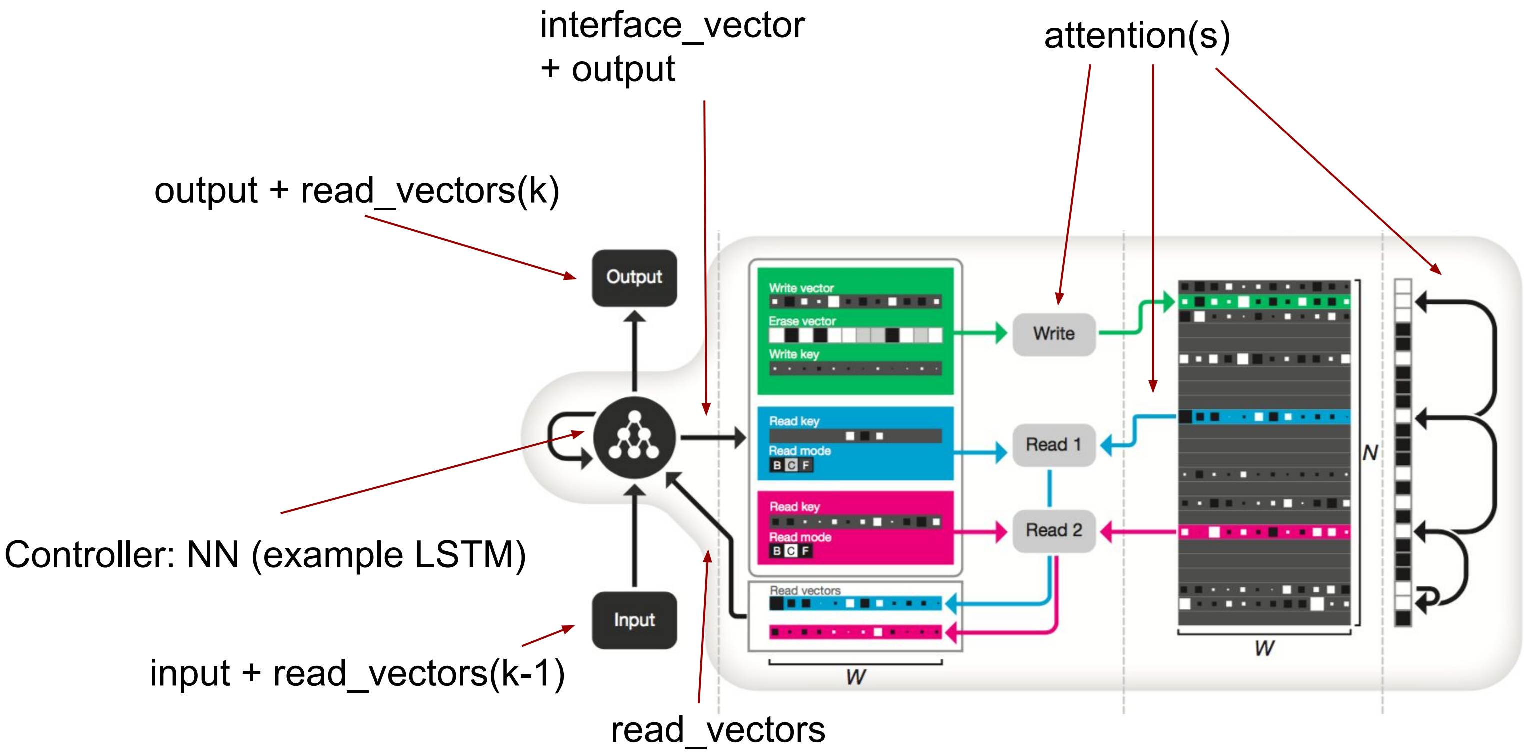
What is Gertrude afraid of?

LSTM: 7.5% mean test error

DNC: 3.8% mean test error

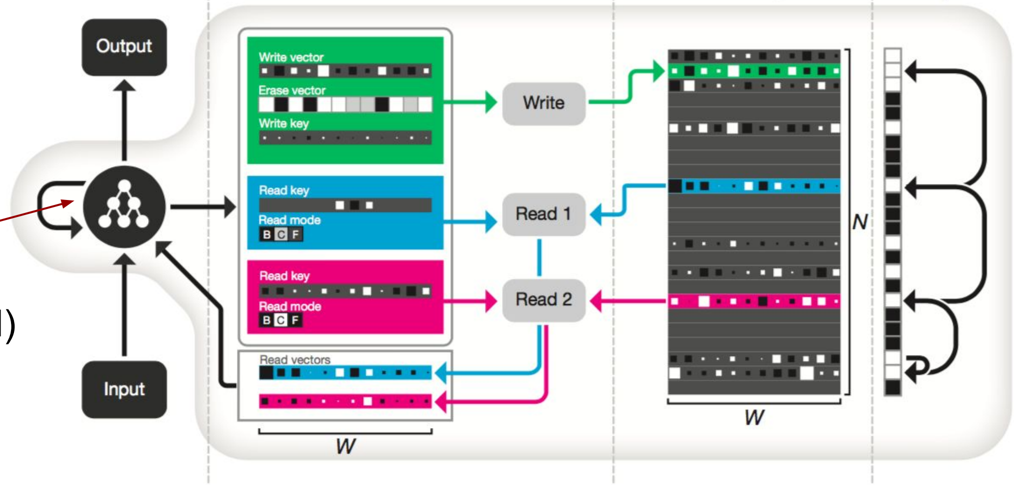

a Weightings
Decode
Locations
Write head
Read head 1
Read head 2
Time
Action
b Goal T constraints
6b2
4l1
5b4
6r3
2b1
5l2
c Board states
d Planned action decodings
Classification accuracy
Action number
Location average
Action frequencies
e t-SNE location goal labels

DETAILS


interface_vector
+ output
attention(s)
output + read_vectors(k)
Output
Write vector
Erase vector
Write key
Write
Read key
Read mode
B C F
Read 1
Read key
Read mode
B C F
Read 2
N
Controller: NN (example LSTM)
Input
Read vectors
W
W
input + read_vectors(k-1)
read_vectors

DETAILS

Controller: NN (example LSTM)



Details

input $$\boldsymbol{i}_t^l = \sigma(W_{\boldsymbol{i}}^l[\chi_t; \boldsymbol{h}_{t-1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_{\boldsymbol{i}}^l)$$

forget $$\boldsymbol{f}_t^l = \sigma(W_{\boldsymbol{f}}^l[\chi_t; \boldsymbol{h}_{t-1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_{\boldsymbol{f}}^l)$$

state $$\boldsymbol{s}_t^l = \boldsymbol{f}_t^l \boldsymbol{s}_{t-1}^l + \boldsymbol{i}_t^l \tanh(W_{\boldsymbol{s}}^l[\chi_t; \boldsymbol{h}_{t-1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_{\boldsymbol{s}}^l)$$

output gate activation $$\boldsymbol{o}_t^l = \sigma(W_{\mathbf{o}}^l[\chi_t; \boldsymbol{h}_{t-1}^l; \boldsymbol{h}_t^{l-1}] + \boldsymbol{b}_{\mathbf{o}}^l)$$

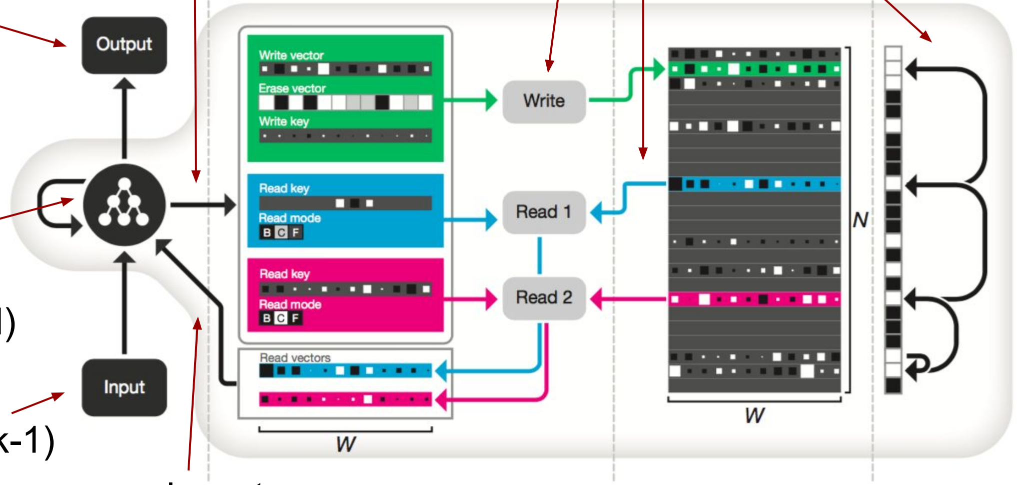
hidden $$\boldsymbol{h}_t^l = \boldsymbol{o}_t^l \tanh(\boldsymbol{s}_t^l)$$

DETAILS

interface_vector
+ output

attention(s)

output + read_vectors(k)

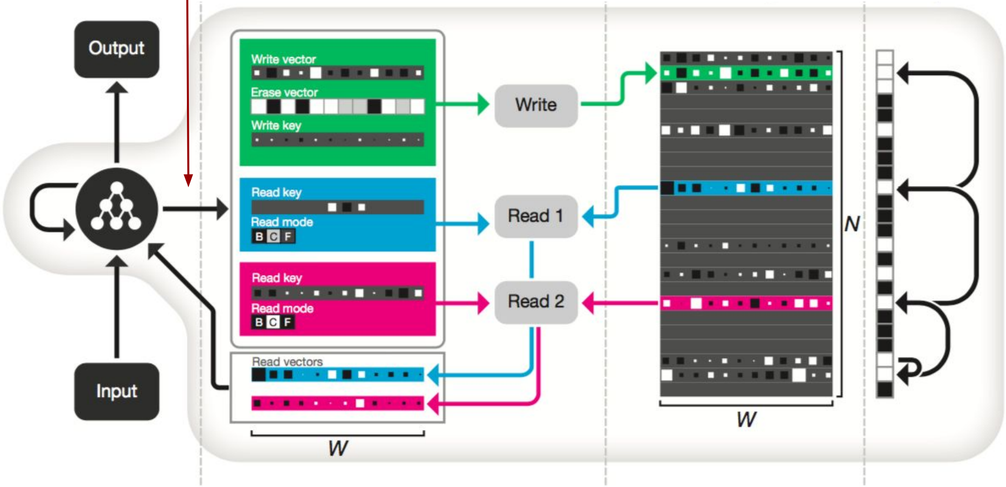


Controller: NN (example LSTM)

input + read_vectors(k-1)

read_vectors

interface_vector
+ output



interface_vector

Interface parameters. Before being used to parameterize the memory interactions, the interface vector $\boldsymbol{\xi}_t$ is subdivided as follows:

$$\boldsymbol{\xi}_t = \left[\boldsymbol{k}_t^{\mathrm{r},1};\ldots;\boldsymbol{k}_t^{\mathrm{r},R};\hat{\beta}_t^{\mathrm{r},1};\ldots;\hat{\beta}_t^{\mathrm{r},R};\boldsymbol{k}_t^{\mathrm{w}};\hat{\beta}_t^{\mathrm{w}};\hat{\boldsymbol{e}}_t;\boldsymbol{v}_t;\hat{f}_t^1;\ldots;\hat{f}_t^R;\hat{g}_t^{\mathrm{a}};\hat{g}_t^{\mathrm{w}};\hat{\boldsymbol{\pi}}_t^1;\ldots;\hat{\boldsymbol{\pi}}_t^R\right]$$

- the write key $\boldsymbol{k}_t^{\mathrm{w}} \in \mathbb{R}^W$;
- the write strength $\beta_t^{\mathrm{w}} = \mathrm{oneplus}(\hat{\beta}_t^{\mathrm{w}}) \in [1,\infty)$;
- the erase vector $\boldsymbol{e}_t = \sigma(\hat{\boldsymbol{e}}_t) \in [0,1]^W$;
- the write vector $\boldsymbol{v}_t \in \mathbb{R}^W$;
- R free gates $\{f_t^i = \sigma(\hat{f}_t^i) \in [0,1]; 1 \leq i \leq R\}$;
- the allocation gate $g_t^{\mathrm{a}} = \sigma(\hat{g}_t^{\mathrm{a}}) \in [0,1]$;
- the write gate $g_t^{\mathrm{w}} = \sigma(\hat{g}_t^{\mathrm{w}}) \in [0,1]$; and
- R read modes $\{\boldsymbol{\pi}_t^i = \mathrm{softmax}(\hat{\boldsymbol{\pi}}_t^i) \in \mathcal{S}_3; 1 \leq i \leq R\}$.

interface_vector

Interface parameters. Before being used to parameterize the memory interactions, the interface vector ξ_t is subdivided as follows:

$$\xi_t = \left[\boldsymbol{k}_t^{\mathrm{r},1};\ldots;\boldsymbol{k}_t^{\mathrm{r},R};\hat{\beta}_t^{\mathrm{r},1};\ldots;\hat{\beta}_t^{\mathrm{r},R};\boldsymbol{k}_t^{\mathrm{w}};\hat{\beta}_t^{\mathrm{w}};\hat{\boldsymbol{e}}_t;\boldsymbol{v}_t;\hat{f}_t^{1};\ldots;\hat{f}_t^{R};\hat{g}_t^{\mathrm{a}};\hat{g}_t^{\mathrm{w}};\hat{\boldsymbol{\pi}}_t^{1};\ldots;\hat{\boldsymbol{\pi}}_t^{R}\right]$$

$$\upsilon_t = W_y[\boldsymbol{h}_t^1;\ldots;\boldsymbol{h}_t^L]$$

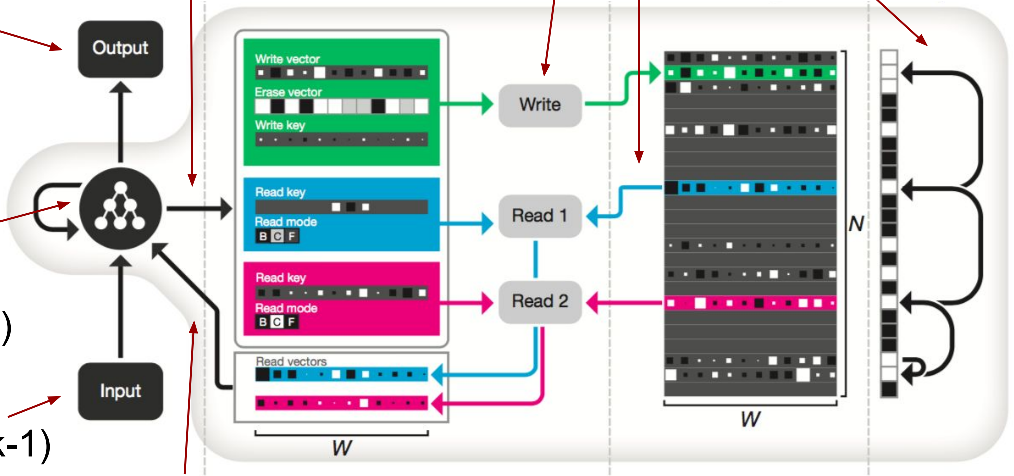
$$\xi_t = W_\xi[\boldsymbol{h}_t^1;\ldots;\boldsymbol{h}_t^L]$$

DETAILS

interface_vector
+ output

attention(s)

output + read_vectors(k)

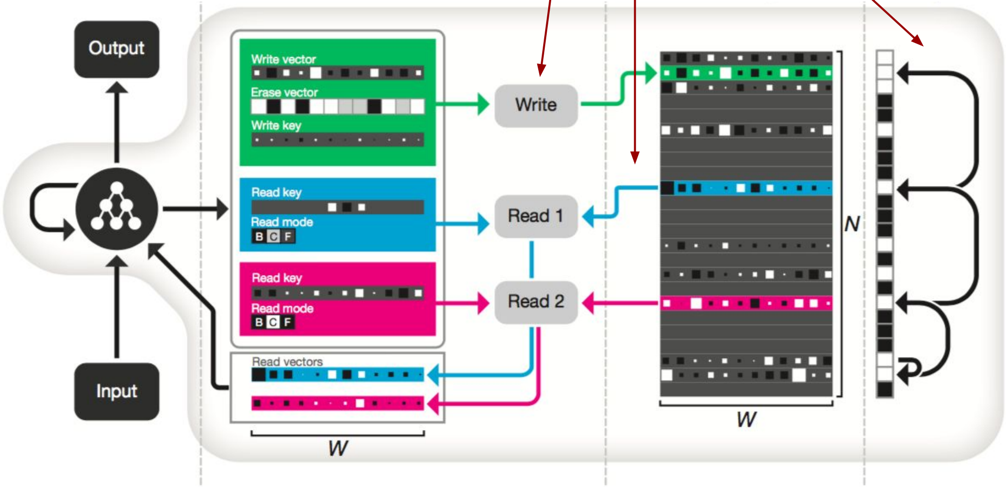


Controller: NN (example LSTM)

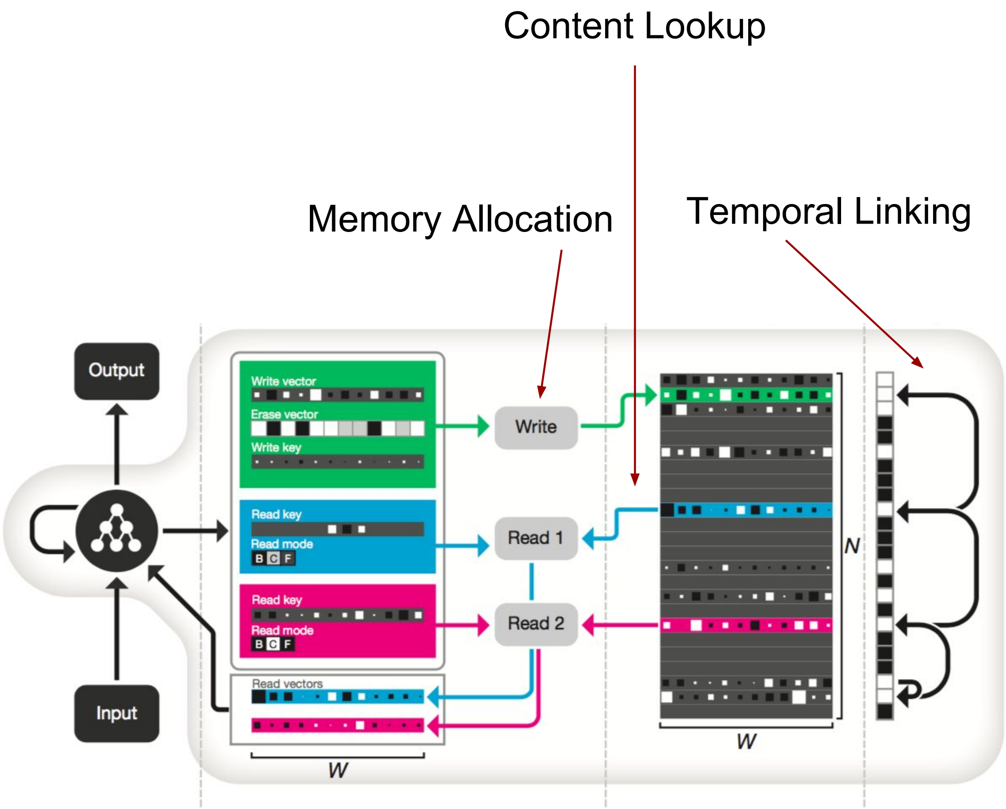
input + read_vectors(k-1)

read_vectors

attention(s)


Output
Input
Write vector
Erase vector
Write key
Read key
Read mode
B C F
Read key
Read mode
B C F
Read vectors
W
Write
Read 1
Read 2
N
W

Details



Content Lookup
Memory Allocation
Temporal Linking
Output
Input
Write vector
Erase vector
Write key
Read key
Read mode
B C F
Read key
Read mode
B C F
Read vectors
W
Write
Read 1
Read 2
N
W

Content lookup (attention 1)

$$\mathcal{C}(M, \boldsymbol{k}, \beta)[i] = \frac{\exp\{\mathcal{D}(\boldsymbol{k}, M[i, \cdot\,])\beta\}}{\sum_j \exp\{\mathcal{D}(\boldsymbol{k}, M[j, \cdot\,])\beta\}}$$

where $\boldsymbol{k} \in \mathbb{R}^W$ is a lookup key, $\beta \in [1, \infty)$ is a scalar representing key strength and $\mathcal{D}$ is the cosine similarity:

$$\mathcal{D}(\boldsymbol{u}, \boldsymbol{v}) = \frac{\boldsymbol{u} \cdot \boldsymbol{v}}{|\boldsymbol{u}||\boldsymbol{v}|}$$

Memory allocation (attention 2)

$$\psi_t = \prod_{i=1}^{R} \left(\mathbf{1} - f_t^i \mathbf{w}_{t-1}^{\mathrm{r},i}\right)$$

$$\boldsymbol{u}_t = (\boldsymbol{u}_{t-1} + \boldsymbol{w}_{t-1}^{\mathrm{w}} - \boldsymbol{u}_{t-1} \circ \boldsymbol{w}_{t-1}^{\mathrm{w}}) \circ \psi_t$$

$$\boldsymbol{a}_t[\phi_t[j]] = (1 - \boldsymbol{u}_t[\phi_t[j]]) \prod_{i=1}^{j-1} \boldsymbol{u}_t[\phi_t[i]]$$

Memory allocation (attention 2)

Free gates
(from interface vector)

Retention vector

$$\psi_t = \prod_{i=1}^{R} \left(\mathbf{1} - f_t^i \mathbf{w}_{t-1}^{\mathrm{r},i}\right)$$

Weights
(learnable)

Element-wise product

Memory usage vector

$$\boldsymbol{u}_t = (\boldsymbol{u}_{t-1} + \boldsymbol{w}_{t-1}^{\mathrm{w}} - \boldsymbol{u}_{t-1} \circ \boldsymbol{w}_{t-1}^{\mathrm{w}}) \circ \psi_t$$

Free list

$\phi_t[1]$ is the index of the least used location

$$\boldsymbol{a}_t[\phi_t[j]] = (1 - \boldsymbol{u}_t[\phi_t[j]]) \prod_{i=1}^{j-1} \boldsymbol{u}_t[\phi_t[i]]$$

Allocation weighting

Write weighting. The controller can write to newly allocated locations, or to locations addressed by content, or it can choose not to write at all. First, a write content weighting $\boldsymbol{c}_t^{\mathrm{w}} \in \mathcal{S}_N$ is constructed using the write key $\boldsymbol{k}_t^{\mathrm{w}}$ and write strength β_t^{w}:

$$\boldsymbol{c}_t^{\mathrm{w}} = \mathcal{C}(M_{t-1}, \boldsymbol{k}_t^{\mathrm{w}}, \beta_t^{\mathrm{w}})$$

$\boldsymbol{c}_t^{\mathrm{w}}$ is interpolated with the allocation weighting $\boldsymbol{a}_t$ defined in equation (1) to determine a write weighting $\boldsymbol{w}_t^{\mathrm{w}} \in \Delta_N$:

$$\boldsymbol{w}_t^{\mathrm{w}} = g_t^{\mathrm{w}}\left[g_t^{\mathrm{a}}\boldsymbol{a}_t + (1 - g_t^{\mathrm{a}})\boldsymbol{c}_t^{\mathrm{w}}\right] \quad (2)$$

where $g_t^{\mathrm{a}} \in [0,1]$ is the allocation gate governing the interpolation and $g_t^{\mathrm{w}} \in [0,1]$ is the write gate. If the write gate is 0, then nothing is written, regardless of the other write parameters; it can therefore be used to protect the memory from unnecessary modifications.

Temporal linking (attention 3)

$$p_0 = \mathbf{0}$$

$$p_t = \left(1 - \sum_i w_t^{\mathrm{w}}[i]\right) p_{t-1} + w_t^{\mathrm{w}}$$

Precedence weights (writing order)

$$L_0[i,j] = 0 \quad \forall\, i,j$$

$$L_t[i,i] = 0 \quad \forall\, i$$

$$L_t[i,j] = (1 - w_t^{\mathrm{w}}[i] - w_t^{\mathrm{w}}[j]) L_{t-1}[i,j] + w_t^{\mathrm{w}}[i] p_{t-1}[j]$$

Temporal linking matrix

Read weighting. Each read head i computes a content weighting $\mathbf{c}_t^{r,i} \in \Delta_N$ using a read key $\boldsymbol{k}_t^{r,i} \in \mathbb{R}^W$:

$$\boldsymbol{c}_t^{r,i} = \mathcal{C}(M_t, \boldsymbol{k}_t^{r,i}, \beta_t^{r,i})$$

Each read head also receives a read mode vector $\boldsymbol{\pi}_t^i \in \mathcal{S}_3$, which interpolates among the backward weighting $\boldsymbol{b}_t^i$, the forward weighting $\boldsymbol{f}_t^i$ and the content read weighting $\boldsymbol{c}_t^{r,i}$, thereby determining the read weighting $\boldsymbol{w}_t^{r,i} \in \mathcal{S}_3$:

$$\boldsymbol{w}_t^{r,i} = \boldsymbol{\pi}_t^i[1]\boldsymbol{b}_t^i + \boldsymbol{\pi}_t^i[2]\boldsymbol{c}_t^{r,i} + \boldsymbol{\pi}_t^i[3]\boldsymbol{f}_t^i$$

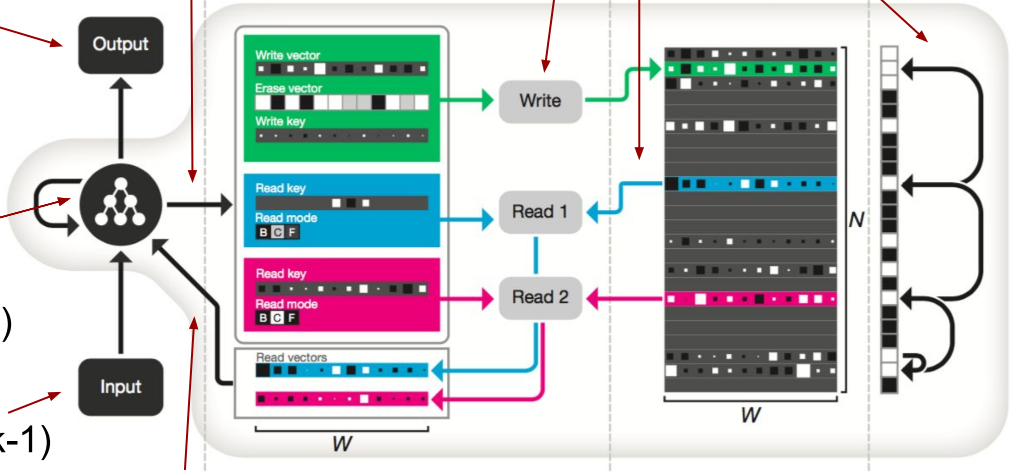
If $\boldsymbol{\pi}_t^i[2]$ dominates the read mode, then the weighting reverts to content lookup using $\boldsymbol{k}_t^{r,i}$. If $\boldsymbol{\pi}_t^i[3]$ dominates, then the read head iterates through memory locations in the order they were written, ignoring the read key. If $\boldsymbol{\pi}_t^i[1]$ dominates, then the read head iterates in the reverse order.

DETAILS

interface_vector
+ <part of> output

attention(s)

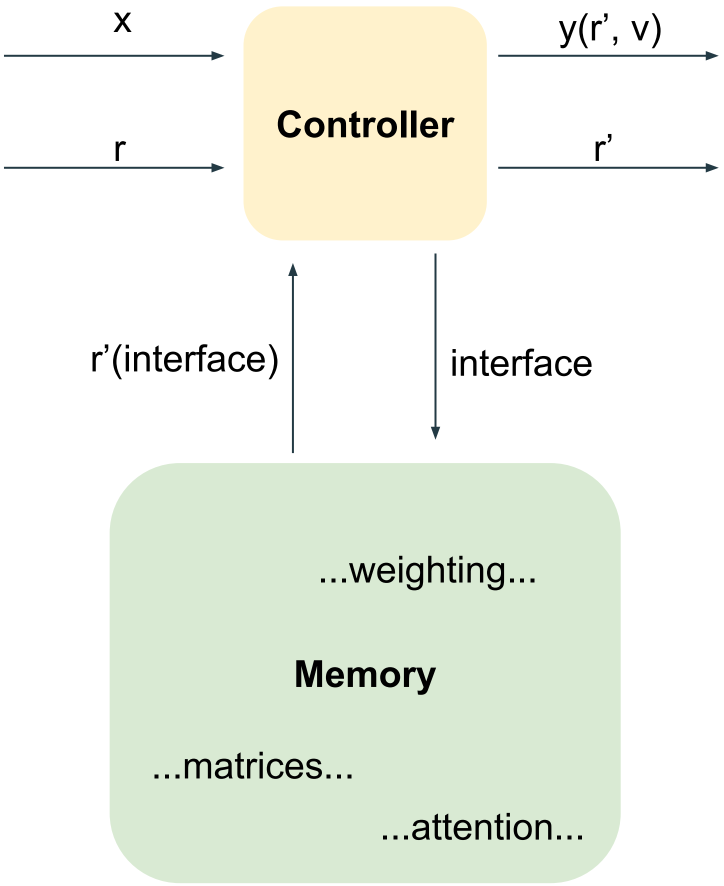
output + read_vectors(k)

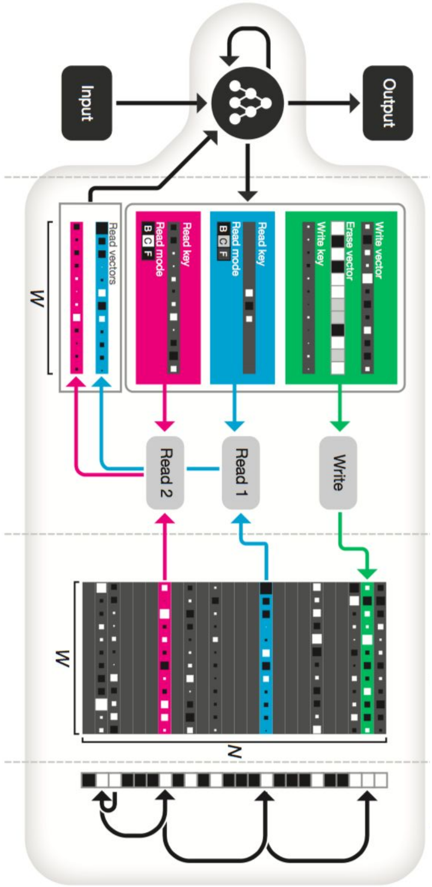


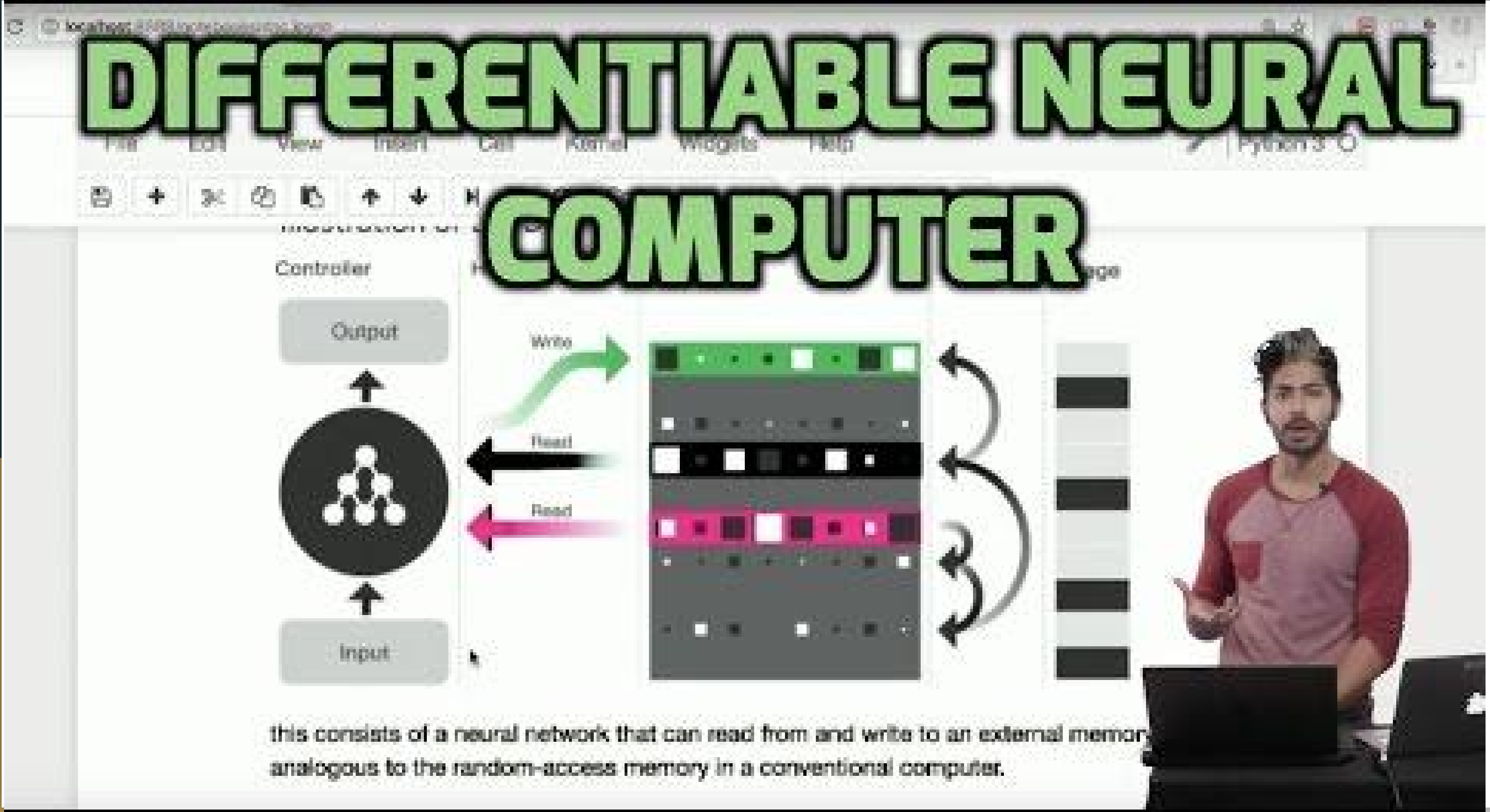
Controller: NN (example LSTM)

input + read_vectors(k-1)

read_vectors


x
r
Controller
y(r’, v)
r’
r’(interface)
interface
...weighting...
Memory
...matrices...
...attention...


Input
Output
Read vectors
Read mode
Read key
Write key
Erase vector
Write vector
Read 2
Read 1
Write
W
N


DIFFERENTIABLE NEURAL
COMPUTER
File Edit View Insert Cell Kernel Widgets Help
Python 3
Controller
Output
Write
Read
Read
Input
this consists of a neural network that can read from and write to an external memor
analogous to the random-access memory in a conventional computer.