



Attention and Memory in Deep Learning

Alex Graves

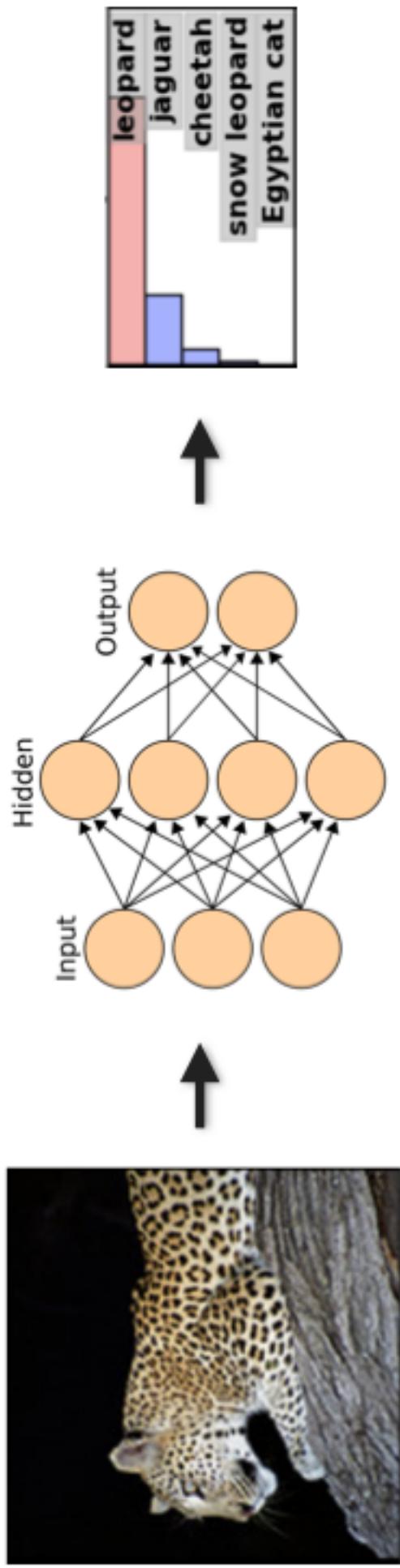
Attention, memory and Cognition

The ability to focus on one thing and ignore others has a vital role in guiding cognition.



Not only does this allow us to pick out salient information from noisy data (the cocktail party problem) it also allows us to pursue one thought at a time, remember one event rather than all events...

Neural Networks



Neural nets are parametric, nonlinear function approximations that can be fit to data to learn functions from input vectors (e.g. photographs) to output vectors (e.g. distributions over class labels)

What does that have to do with **attention**?

Implicit Attention in Neural Networks

What does that have to do with **attention**?

Implicit Attention in Neural Networks

Deep nets naturally learn a form of **implicit attention** where they respond more strongly to some parts of the data than others

To a first approximation, we can visualise this by looking at the network **Jacobian** — sensitivity of the network outputs with respect to the inputs

Neural Network Jacobian

Neural Network Jacobian

x = size k input vector

y = size m output vector

Jacobian J = m x k matrix

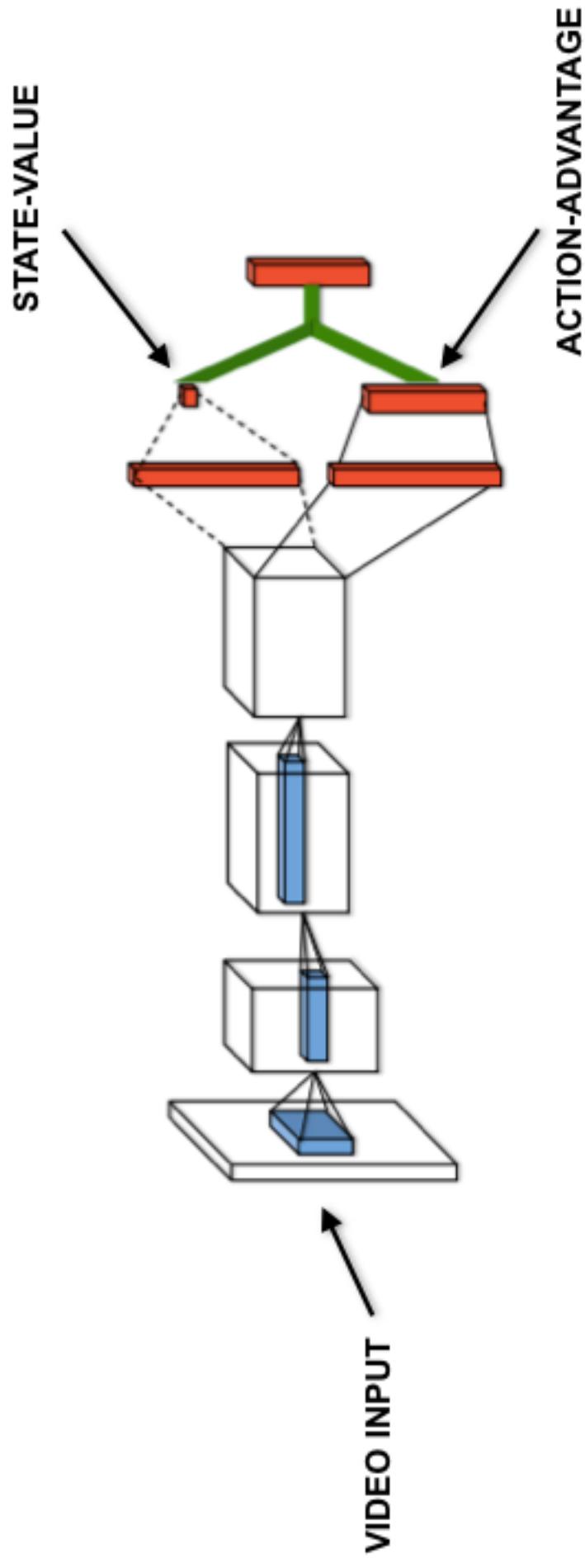
$$J = \begin{bmatrix} \frac{\partial y_1}{\partial x_1} & \cdots & \frac{\partial y_1}{\partial x_k} \\ \vdots & \ddots & \vdots \\ \frac{\partial y_m}{\partial x_1} & \cdots & \frac{\partial y_m}{\partial x_k} \end{bmatrix}$$
$$J_{ij} = \frac{\partial y_i}{\partial x_j}$$

Can compute with ordinary backdrop
(just set output 'errors' = output activations)

Jacobian in Action: Duelling Network

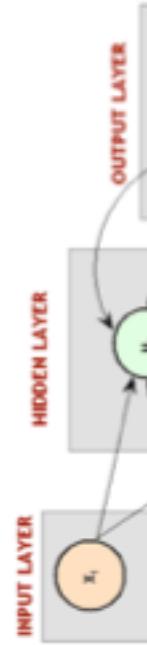
Just set output zeros – output activations)

Jacobian in Action: Duelling Network

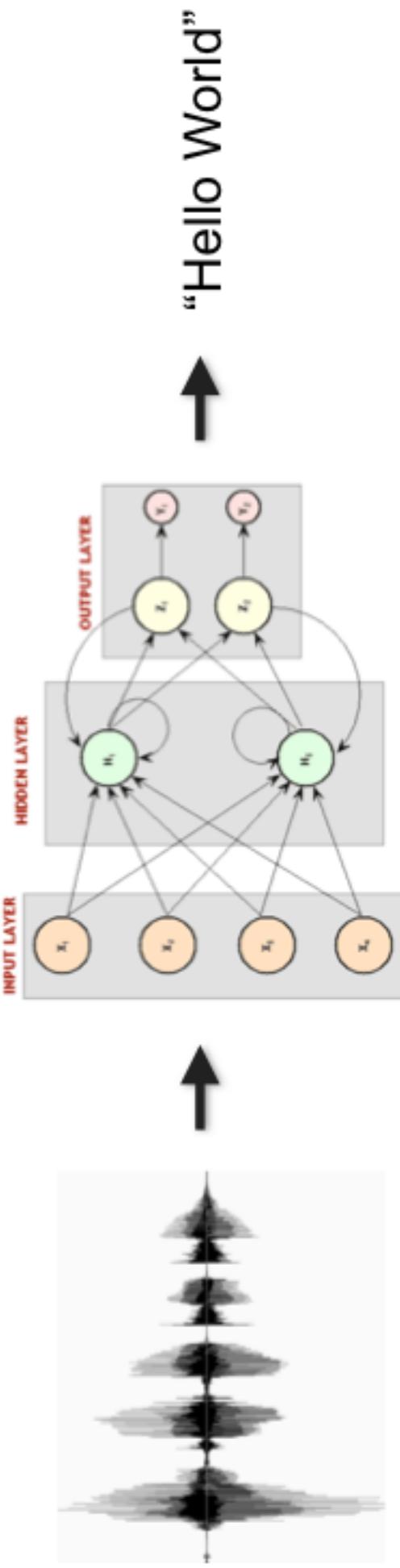


Dueling Network Architectures for Deep Reinforcement Learning, Wang et. al. (2015)

Attention and memory in Recurrent Networks (RNNs)

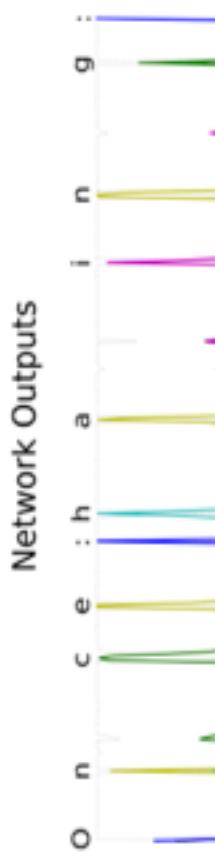


Attention and memory in Recurrent Networks (RNNs)

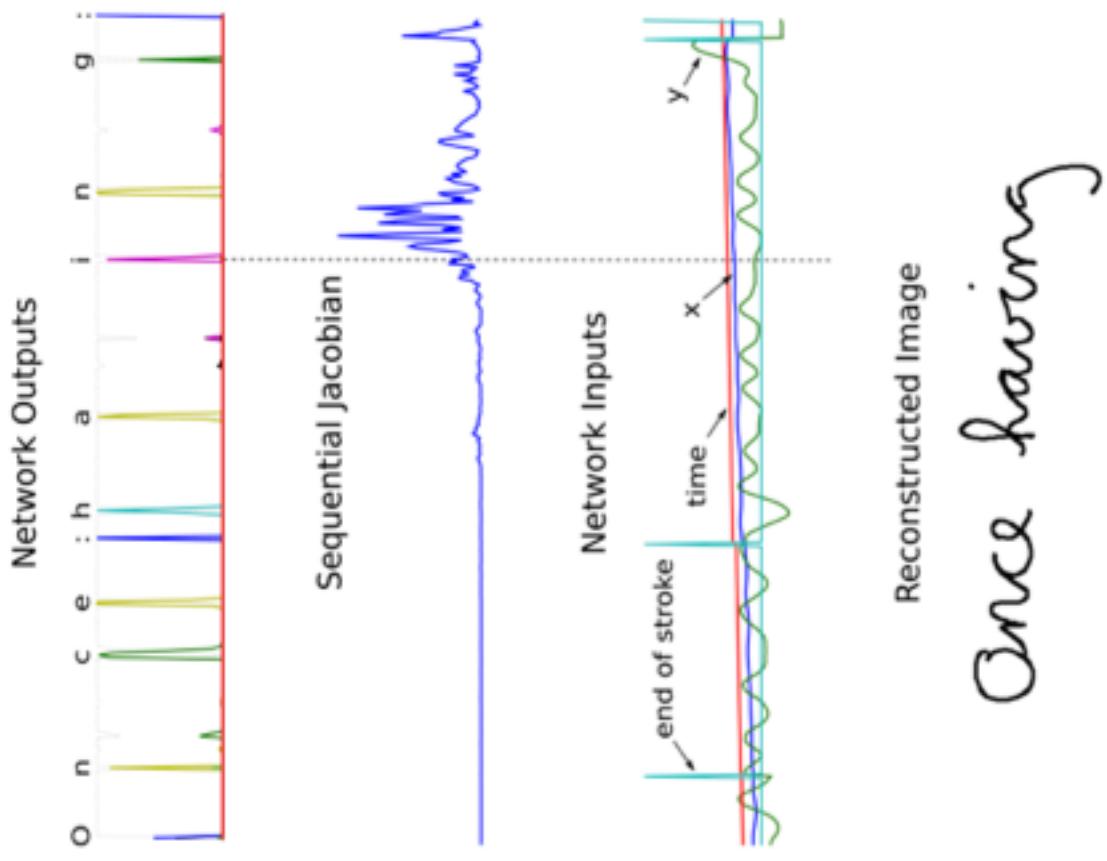


RNNs contain a recursive hidden state and learn functions from sequences of inputs (e.g. a speech signal) to sequences of outputs (e.g. words)

The **sequential Jacobian** shows which past inputs they **remember** when predicting current outputs.



► The **Sequential Jacobian** is the set of derivatives of one network output with respect



- The **Sequential Jacobian** is the set of derivatives of one network output with respect to all the inputs

$$J_k^t = \left(\frac{\partial y_k^t}{\partial x^1}, \frac{\partial y_k^t}{\partial x^2} \dots \right)$$

- It shows how the network responds to widely separated, but related, inputs, such as the **delayed dot** of the 'i' in 'having'

Implicit Attention
allows **reordering**
in machine

to
reach
the
official
residency



Implicit Attention
allows **reordering**
in machine
translation:

to reach the official residency of Prime Minister Nawaz Sharif.

“*to reach*” -> “*zu erreichen*”

Neural Machine Translation in Linear Time,
Kalchbrenner et. al. (2016)

Explicit Attention

Implicit attention is great, but there are still advantages to an **explicit attention** mechanism that limits the data presented to the network in

Explicit Attention

Implicit attention is great, but there are still advantages to an **explicit attention** mechanism that limits the data presented to the network in some way:

- Computational **efficiency**
- **Scalability** (e.g. fixed sized glimpse for any size image)
- **Sequential processing** of static data (e.g. moving gaze)
- **Easier to interpret**

Neural Attention Models

The **network** receives **input** and produces **output** as usual.



It also produces an extra output of something.

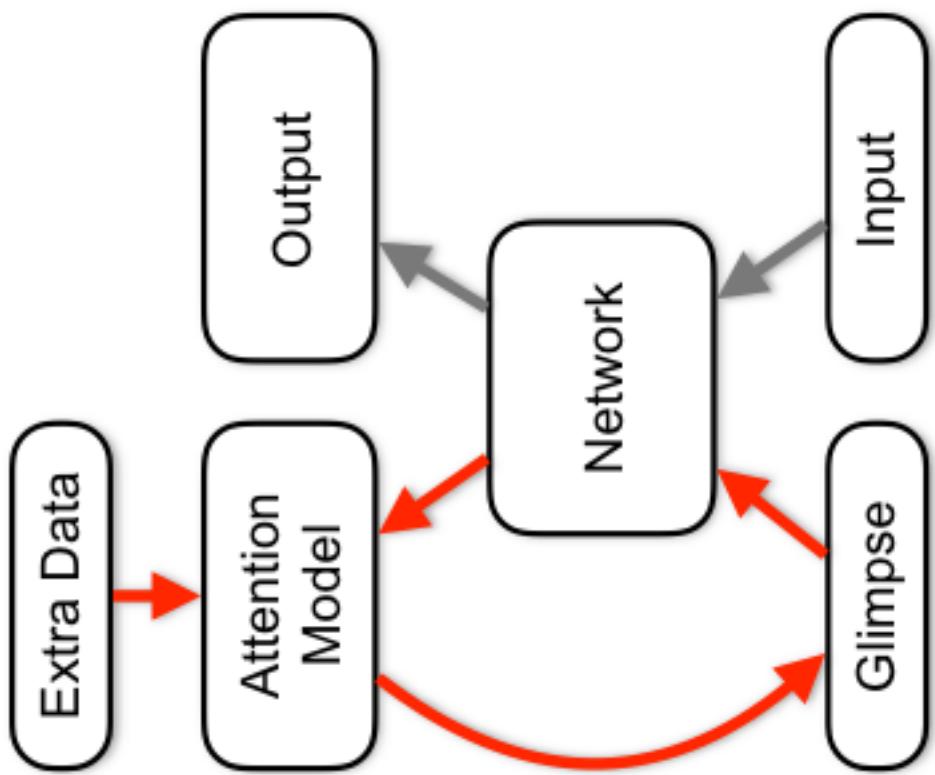
Neural Attention Models

The **network** receives **input** and produces **output** as usual.

It also produces an extra set of outputs used to parameterise an **attention model**

The attention model then operates on some **extra data** (image, audio sample, text to be translated...) to create a fixed-size "**glimpse**" that is passed to the network as an extra input at the next time step

The complete system is **recurrent**, even if the network isn't



Glimpse Distribution

Attention models generally work by defining a probability distribution over glimpses g of the data x and some set of attention outputs a from the network:



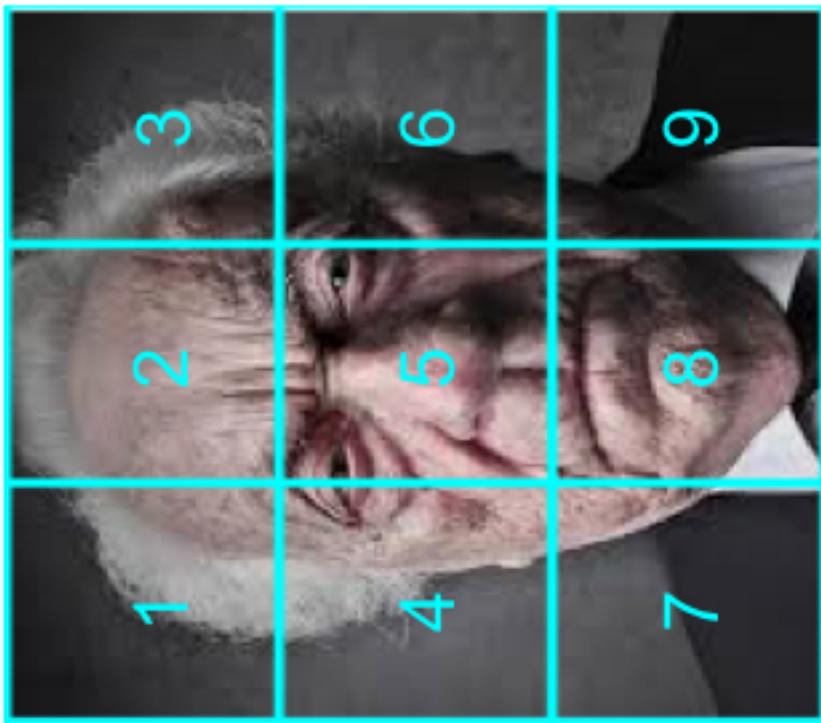
Glimpse Distribution

Attention models generally work by defining a probability distribution over glimpses \mathbf{g} of the data \mathbf{x} and some set of attention outputs \mathbf{a} from the network:

$$\Pr(\mathbf{g}|\mathbf{a})$$

simplest case: \mathbf{a} just assigns probabilities to a set of discrete glimpses:

$$\Pr(\mathbf{g}_k|\mathbf{a}) = \frac{\exp(a_k)}{\sum_{k'} \exp(a_{k'})}$$



Attention with RL

We can treat the distribution over glimpses \mathbf{g} as a **stochastic policy** $\pi_{\mathbf{a}}$, sample from it, and use **RL** techniques (with reward R = task loss L induced by the glimpse) to train the attention model

Attention with RL

We can treat the distribution over glimpses \mathbf{g} as a **stochastic policy** $\pi_{\mathbf{a}}$, sample from it, and use **RL** techniques (with reward R = task loss L induced by the glimpse) to train the attention model

$$\pi_{\mathbf{a}} = \Pr(\mathbf{g}_k | \mathbf{a})$$

$$R = \mathbb{E}_{\mathbf{g} \sim \pi_{\mathbf{a}}} [\log \pi_{\mathbf{a}} L(\mathbf{g})]$$

$$\nabla_{\mathbf{a}} R = \mathbb{E}_{\mathbf{g} \sim \pi_{\mathbf{a}}} [\nabla_{\mathbf{a}} \log \pi_{\mathbf{a}} L(\mathbf{g})]$$

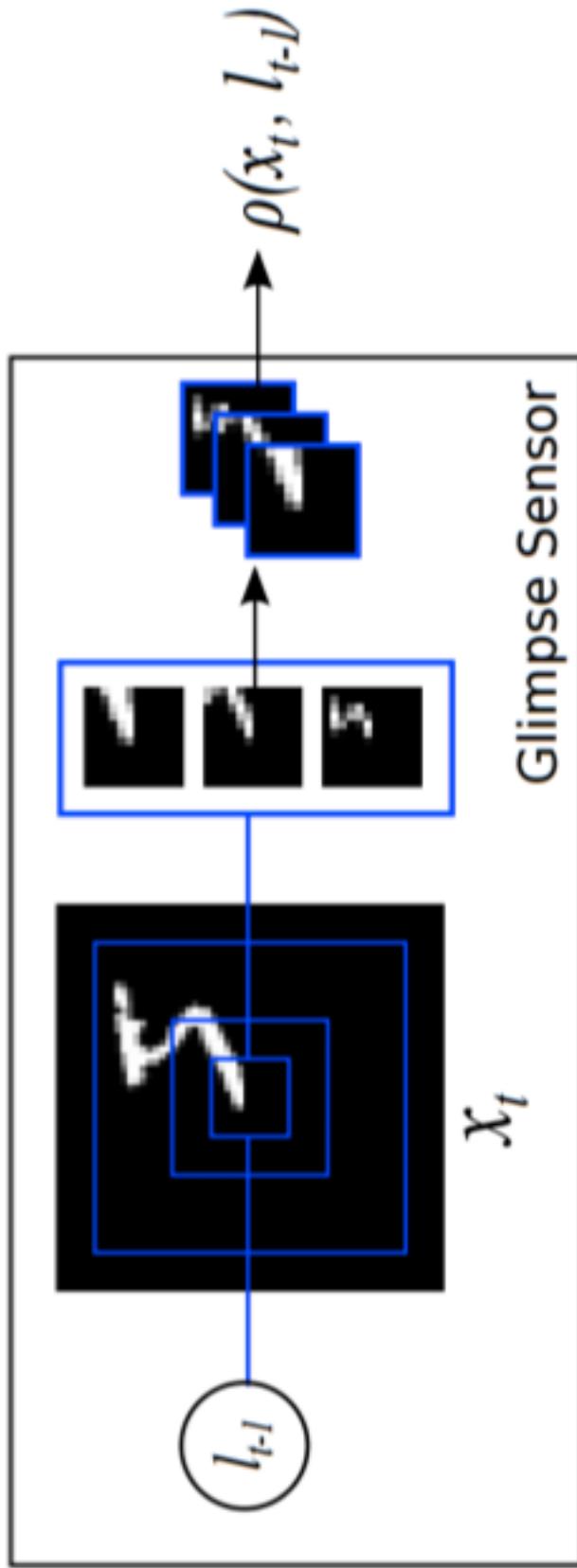
In general we can use RL methods for supervised tasks any time some module in the network is **non-differentiable**

Complex Glimpses

Generally the glimpse distribution is more complex than just a softmax (e.g. Gaussian over co-ordinates, width, height...) and the glimpses are more complex than image tiles (e.g. foveal models)

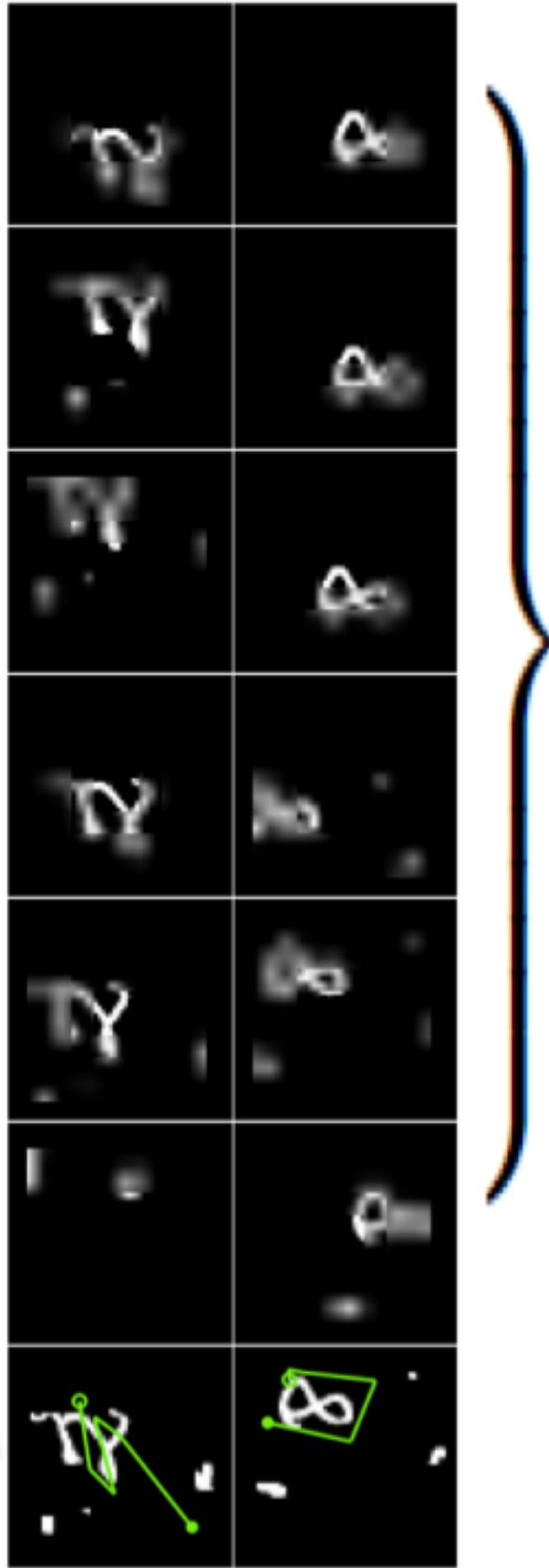
Complex Glimpses

Generally the glimpse distribution is more complex than just a softmax (e.g. Gaussian over co-ordinates, width, height...) and the glimpses are more complex than image tiles (e.g. foveal models)



6 point “Glimpse path” (in green) while trying to classify image



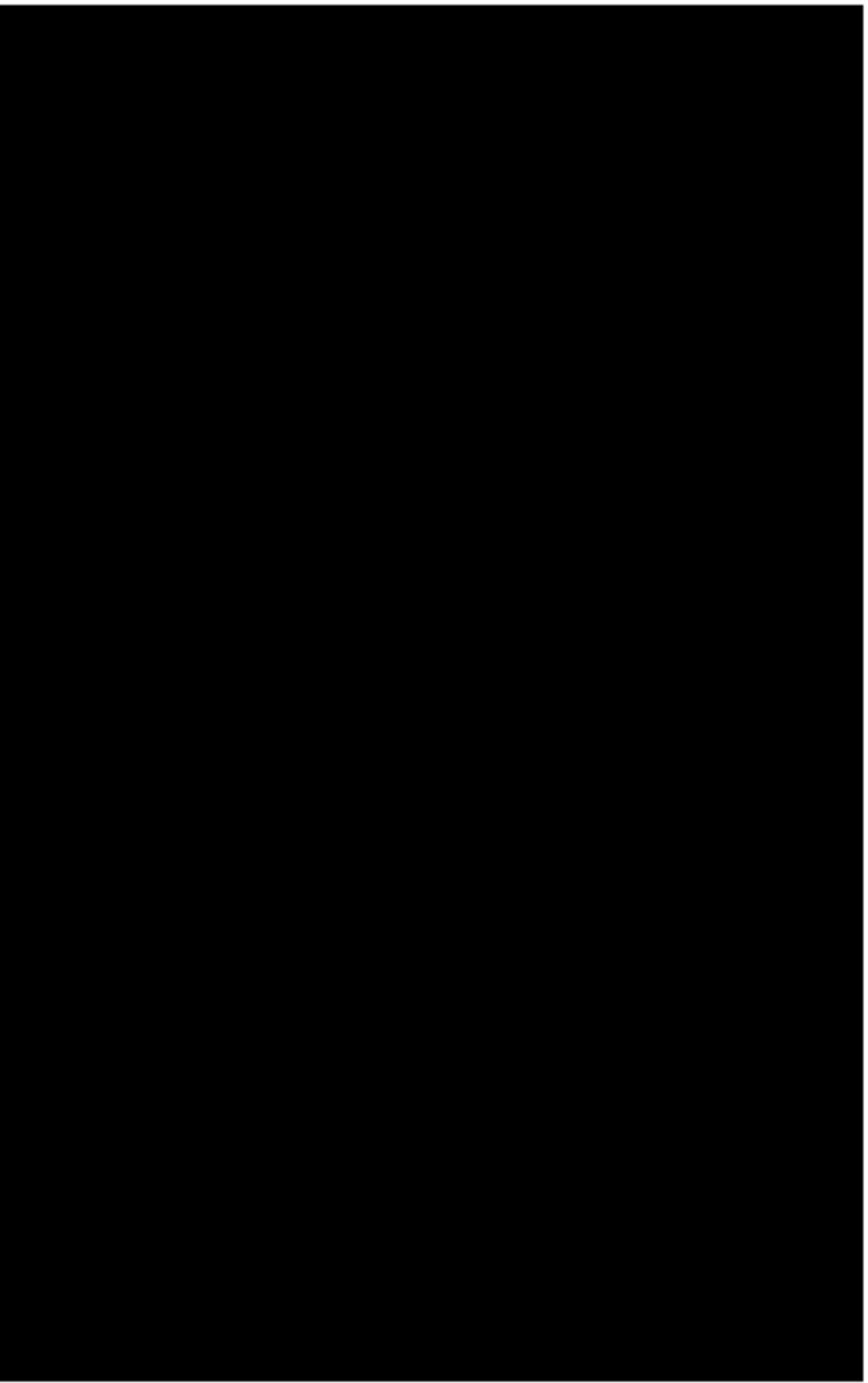


6 Foveal Glimpses seen by the network

Recurrent Models of Visual Attention, Mnih et. al. (2014)

Soft Attention

Multiple Object Recognition with Visual Attention, Ba et. al. (2014)



The last examples used **hard attention**: fixed size attention windows moved around the image, trained with RL techniques.

Robots have to look left or right, but in many cases attention doesn't need to be hard: we just want to focus more on certain regions and less on

The last examples used **hard attention**: fixed size attention windows moved around the image, trained with RL techniques.

Robots have to look left or right, but in many cases attention doesn't need to be hard: we just want to focus more on certain regions and less on others.

If we do this in a differentiable way, we get **soft attention** which we can train **end-to-end** with **backprop**

Generally easier than using RL, but more expensive to compute

Soft Attention

Basic template: we use the attention parameters \mathbf{a} to determine a distribution $\text{Pr}(\mathbf{g}|\mathbf{a})$ as before, only now we take an **expectation** over all possible glimpses instead of a **sample**

Basic template: we use the attention parameters \mathbf{a} to determine a distribution $\Pr(\mathbf{g}|\mathbf{a})$ as before, only now we take an **expectation** over all possible glimpses instead of a **sample**

$$\mathbf{g} = \sum_{\mathbf{g}' \in \mathbf{x}} \mathbf{g}' \Pr(\mathbf{g}'|\mathbf{a})$$

This is differentiable w.r.t. \mathbf{a} as long as $\Pr(\mathbf{g}|\mathbf{a})$ is:

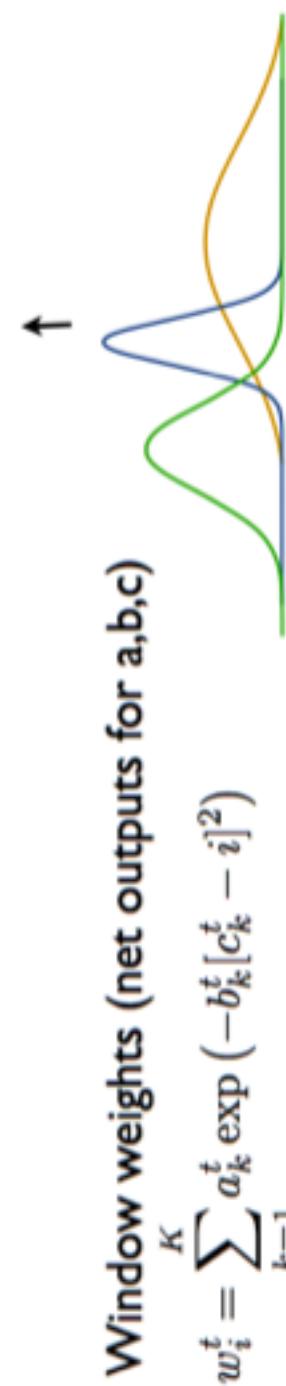
$$\nabla_{\mathbf{a}} \mathbf{g} = \sum_{\mathbf{g}' \in \mathbf{x}} \mathbf{g}' \nabla_{\mathbf{a}} \Pr(\mathbf{g}'|\mathbf{a})$$

Location Attention

Window vector (input to net)

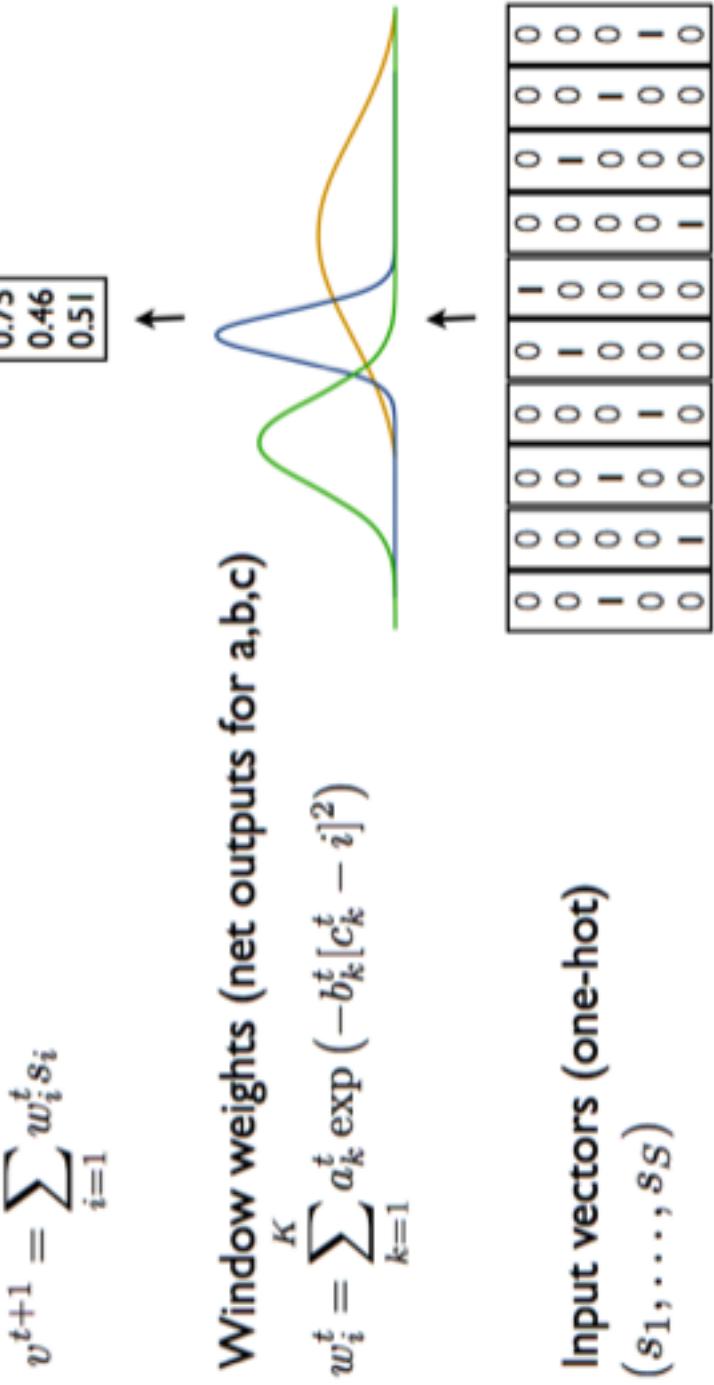
$$v^{t+1} = \sum_{i=1}^S w_i^t s_i$$

1.01
1.84
0.75
0.46
0.51



Attention

$$v^{t+1} = \sum_{i=1} w_i^t s_i$$



Generating Sequences with Recurrent Neural Networks, Graves (2013)

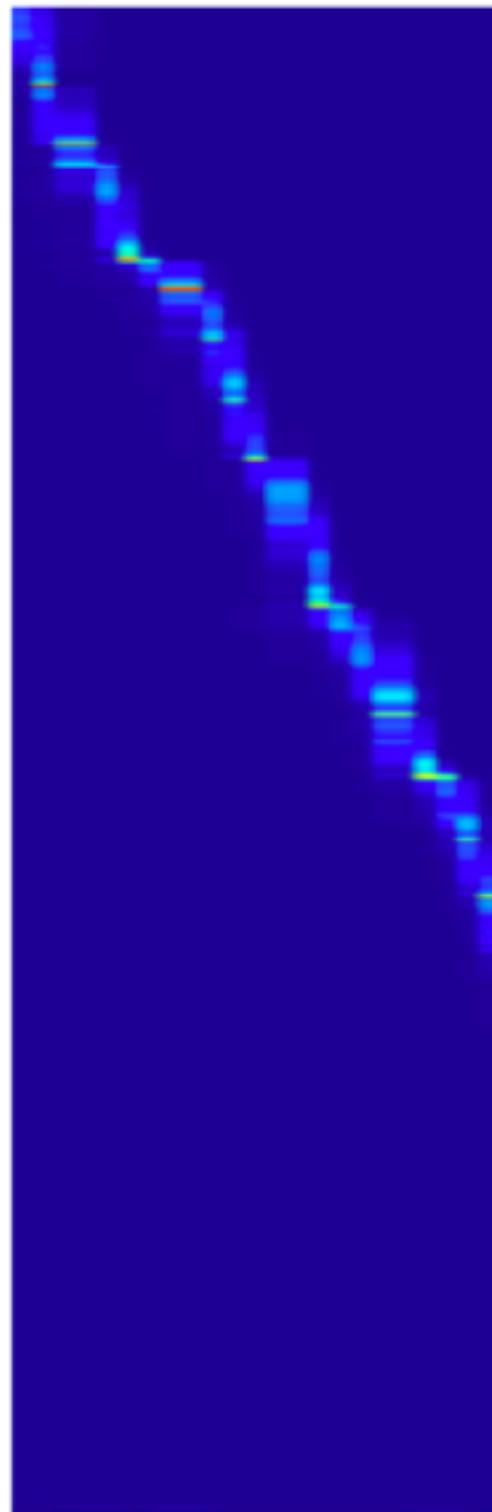
Handwriting Synthesis With Soft Reading

These sequences were generated by
picking samples at every step

These sequences were generated by
picking samples at every star

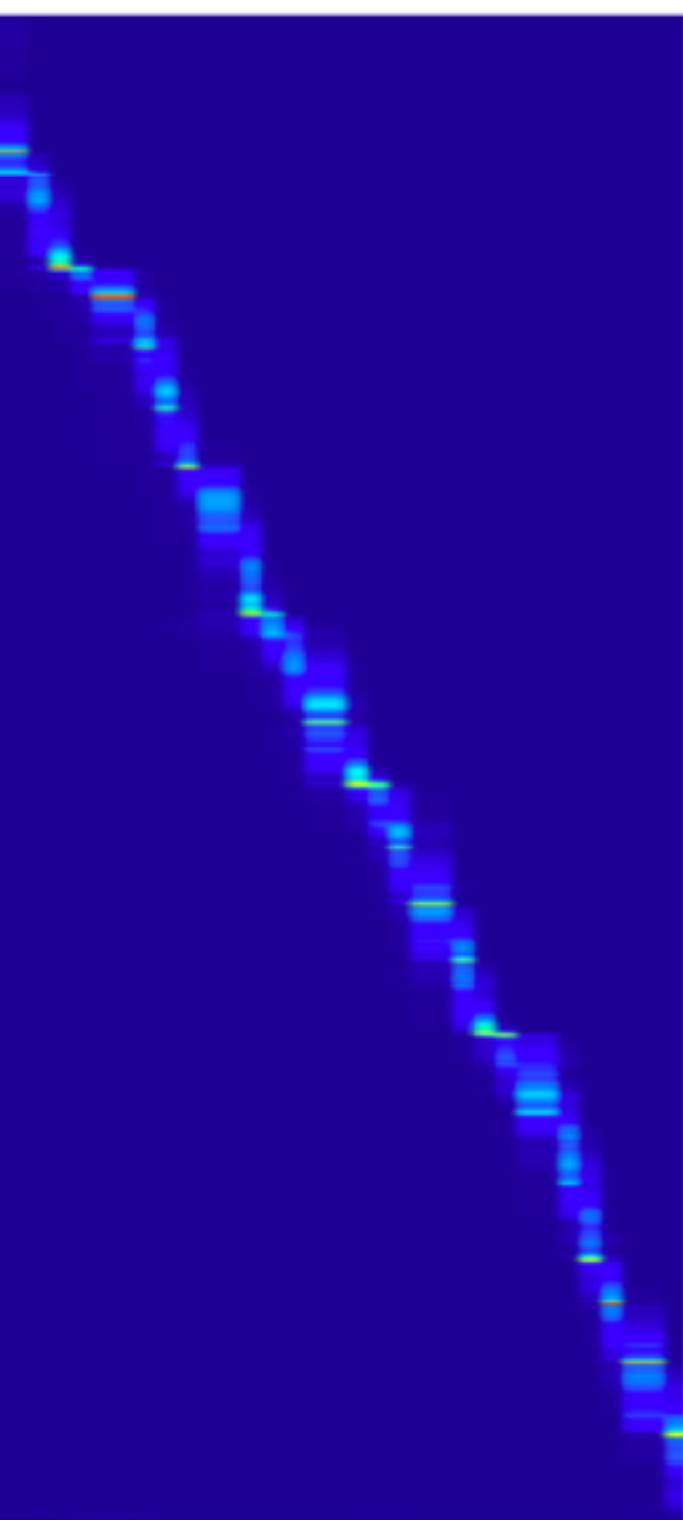
every line is a different style
yes, real people write this badly

Alignment



at the master from

Thought that the muster fric



+ thought that the muster from

Associative Attention

Instead of attending by position, we can attend by **content**. In this setting \mathbf{a} = a **key vector** that is compared to *all* glimpses \mathbf{g} using some **similarity function** S . The scores are normalised and used to define $\text{Pr}(\mathbf{g}|\mathbf{a})$

$$\text{Pr}(\mathbf{g}|\mathbf{a}) = \frac{\exp(S(\mathbf{g}, \mathbf{a}))}{\sum_{\mathbf{g}' \in \mathbf{x}} \exp(S(\mathbf{g}', \mathbf{a}))}$$

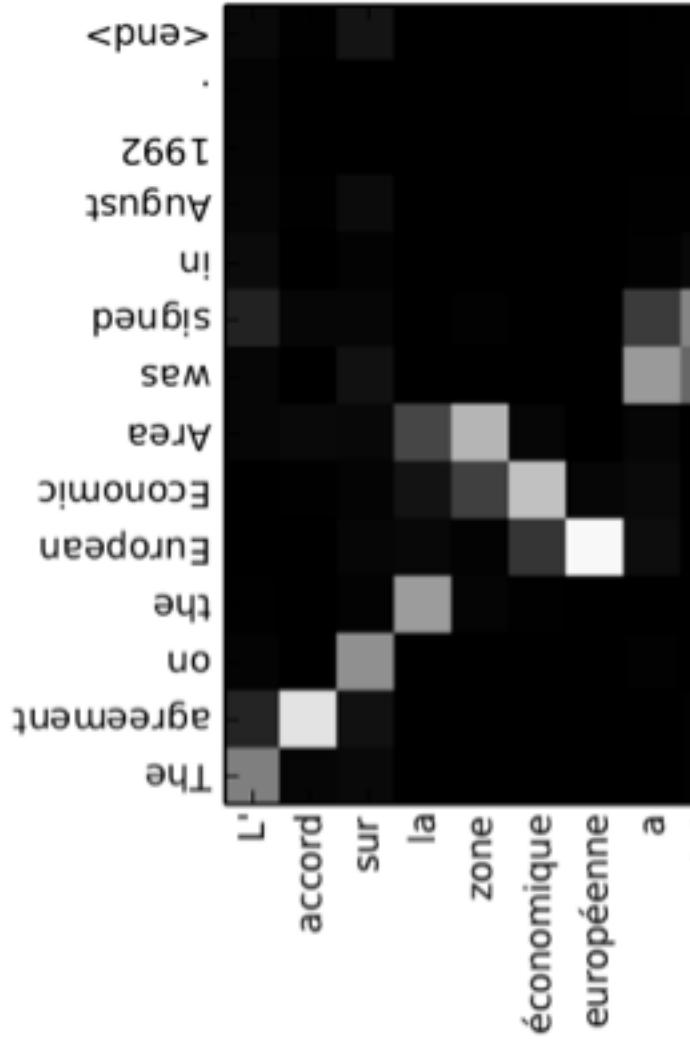
a = a **key vector** that is compared to *all* glimpses **g** using some **similarity function S**. The scores are normalised and used to define $\text{Pr}(\mathbf{g}|\mathbf{a})$

$$\text{Pr}(\mathbf{g}|\mathbf{a}) = \frac{\exp(S(\mathbf{g}, \mathbf{a}))}{\sum_{\mathbf{g}' \in \mathbf{x}} \exp(S(\mathbf{g}', \mathbf{a}))}$$

S can be learned (MLP, linear operator...) or fixed (dot product, cosine similarity...). Yields a **Multidimensional, feature-based** lookup: natural way to search data

Neural Machine Translation by Jointly Learning to Align and Translate, Bahdanau et. al. (2014)

Reordering in machine translation using associative attention



associative attention

accord

sur

la

zone

économique

européenne

a

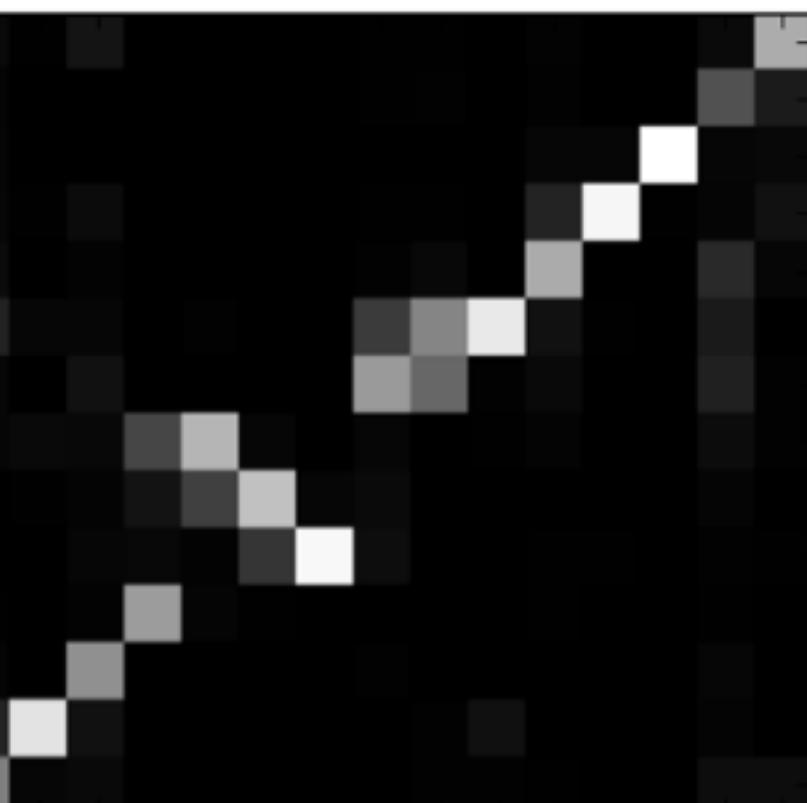
été

signé

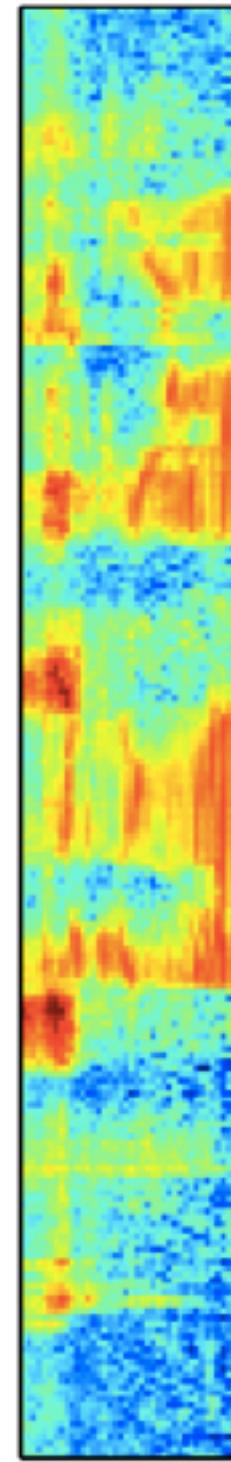
en

août

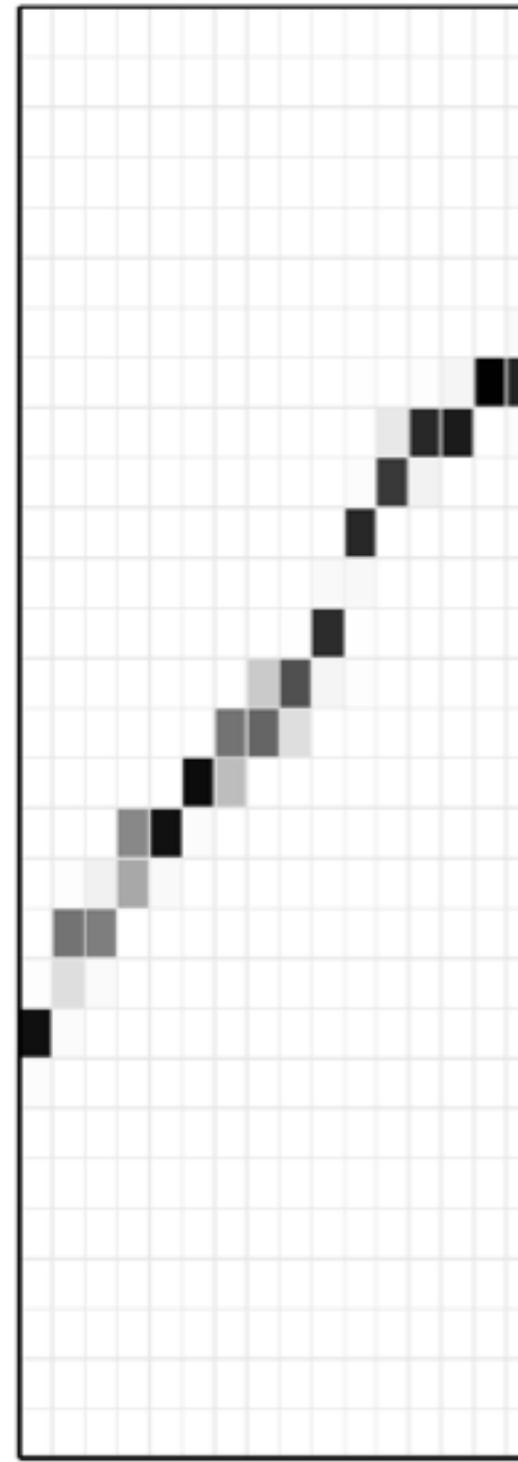
1992



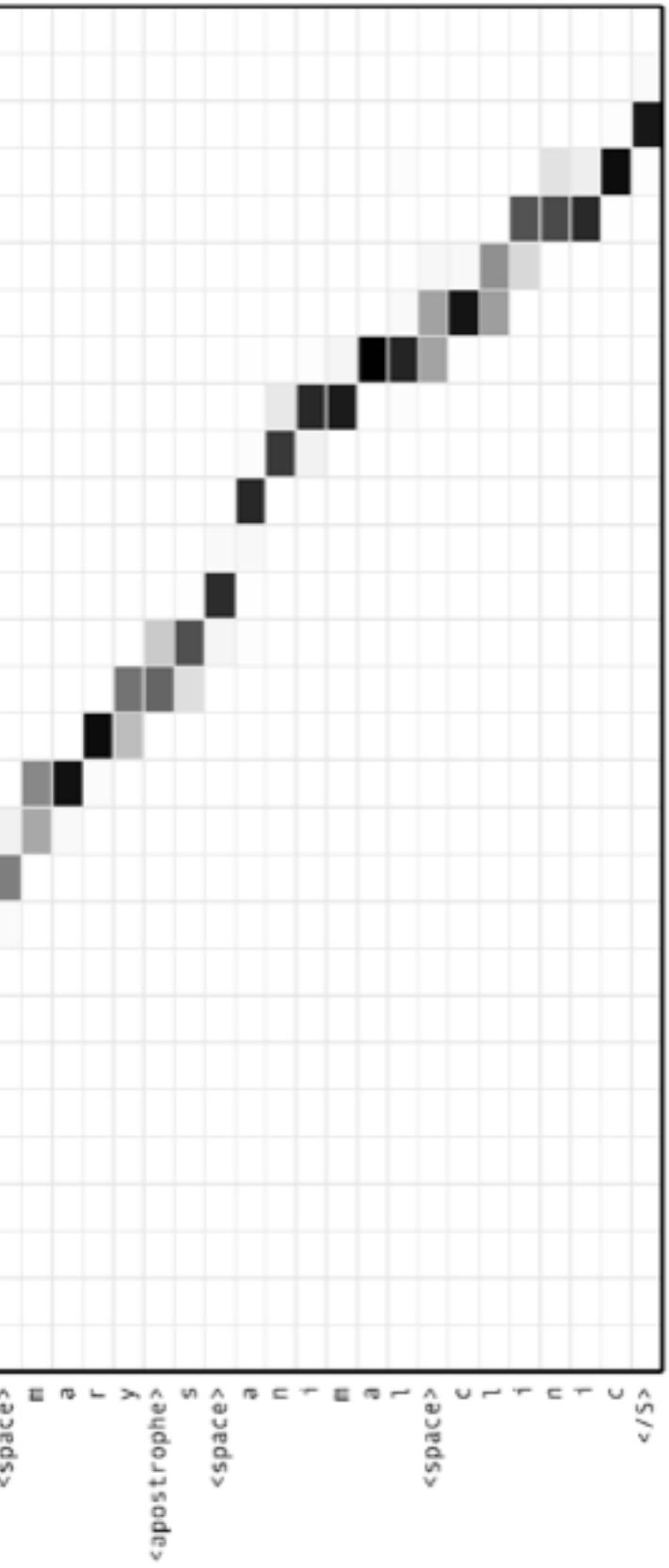
*Neural Machine Translation by Jointly Learning
to Align and Translate*, Bahdanau et. al. (2014) <end>



Audio



s t e a r y
<space> s
<apostrophe> s
<space> a n t e a



Listen, Attend and Spell, Chan et. al. (2015)

by ent270 ,ent223 updated 9:35 am et ,mon march 2 ,2015
(ent223) ent63 went familial for fall at its fashion show in
ent231 on sunday ,dedicating its collection to `` mamma ''
with nary a pair of `` mom jeans "in sight .ent164 and ent21 ,
who are behind the ent196 brand ,sent models down the
runway in decidedly feminine dresses and skirts adorned
with roses ,lace and even embroidered doodles by the
designers 'own nieces and nephews .many of the looks
featured saccharine needlework phrases like `` i love you ,
...

by ent423 ,ent261 correspondent updated 9:49 pm et ,thu
march 19 ,2015 (ent261) a ent114 was killed in a parachute
accident in ent45 ,ent85 ,near ent312 ,a ent119 official told
ent261 on wednesday .he was identified thursday as
special warfare operator 3rd class ent23 ,29 ,of ent187 ,
ent265 . `` ent23 distinguished himself consistently
throughout his career .he was the epitome of the quiet
professional all facets of his life ,and he leaves an
inspiring legacy of natural tenacity and focused
...

ent207 on wednesday . he was identified on tuesday as
special warfare operator 3rd class ent23 , 29 , of ent187 ,
ent265 . `` ent23 distinguished himself consistently

throughout his career . he was the epitome of the quiet
professional in all facets of his life , and he leaves an
inspiring legacy of natural tenacity and focused
... .

ent119 identifies deceased sailor as X , who leaves behind
a wife

X dedicated their fall fashion show to moms

Teaching Machines to Read and Comprehend, Hermann et. al. (2015)

Introspective Attention

So far we have looked at attention to **external data**

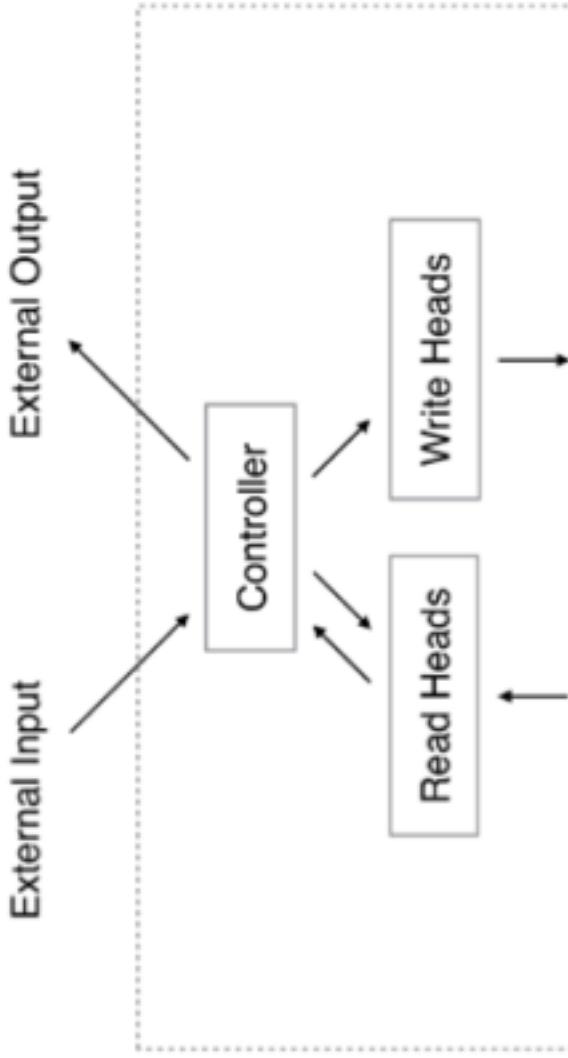
Also useful to selectively attend to the network's internal state or
memory: **introspective attention**

With internal information we can do selective **writing** as well as

Also useful to selectively attend to the network's internal state or memory: **introspective attention**

With internal information we can do selective **writing** as well as **reading**, allowing the network to **iteratively modify** its state

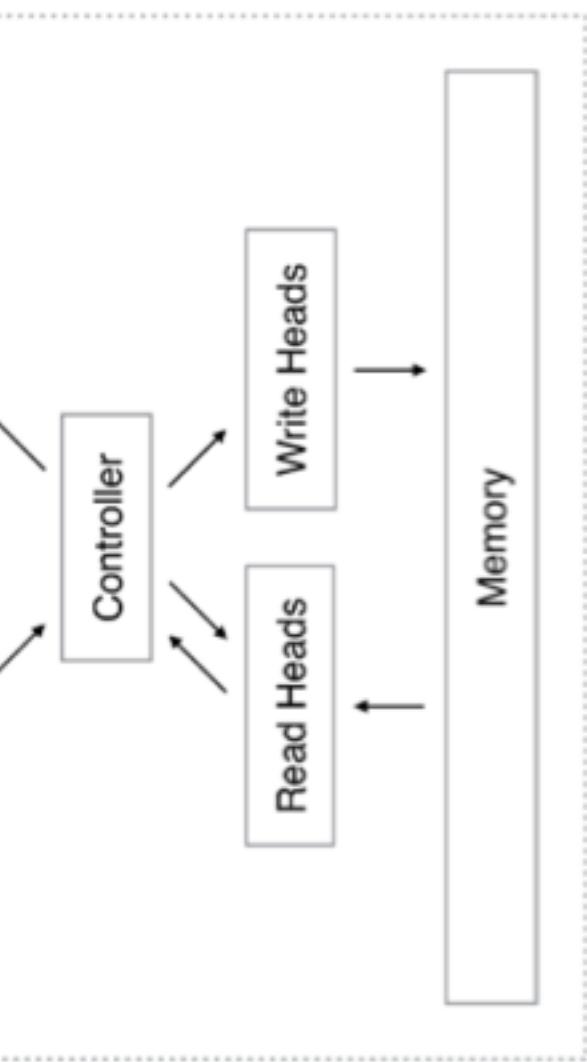
Neural Turing Machines



The **Controller** is a neural network
(recurrent or feedforward)

The **Heads** select portions of the
memory and **read** or **write** to them

The **Controller** is a **neural network**
(recurrent or feedforward)



Neural Turing Machines, Graves et. al. (2014)

Addressing by Content

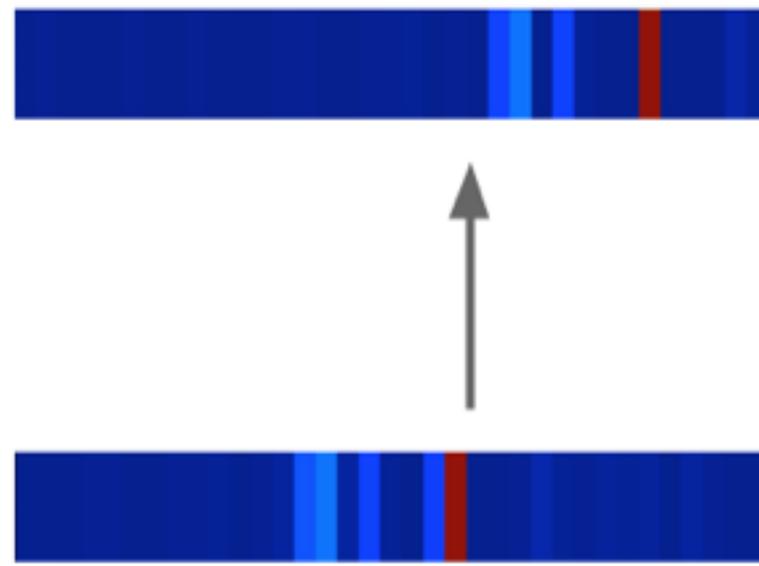
A **key vector** \mathbf{k} is emitted by the controller and compared to the content of each memory location $\mathbf{M}[i]$ using a similarity measure $S(\cdot, \cdot)$ (e.g. **cosine distance**) then normalised with a **softmax**. A '**sharpness**' β is used to narrow the focus. Finds the memories '**closest**' to the key

$$\exp(\beta S(\mathbf{k}, \mathbf{M}[i]))$$

coseine distance) then normalised with a **softmax**. A '**sharpness**' β is used to narrow the focus. Finds the memories '**closest**' to the key

$$\mathbf{w}[i] = \frac{\exp(\beta S(\mathbf{k}, \mathbf{M}[i]))}{\sum_j \exp(\beta S(\mathbf{k}, \mathbf{M}[j]))}$$

Addressing by Location

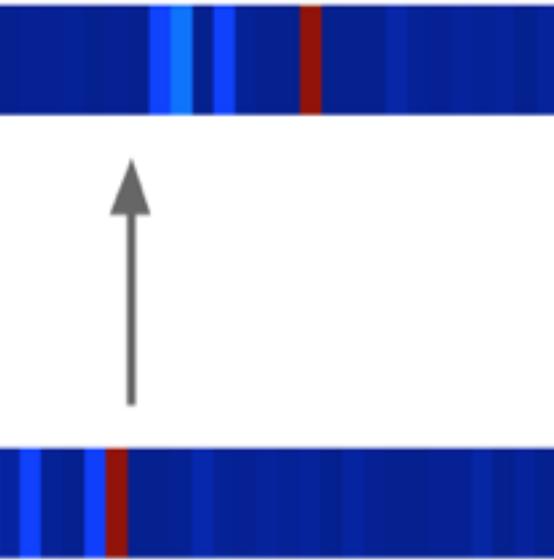


The controller outputs a **shift kernel \mathbf{s}** (e.g. a softmax on $[-n, n]$) which is convolved with a weighting \mathbf{w} to produce a shifted weighting $\hat{\mathbf{w}}$.

$$\hat{\mathbf{w}}[i] = \sum \mathbf{w}[j] \mathbf{s}(i - j)$$

softmax on [-n,n]) which is convolved with a weighting \mathbf{w} to produce a shifted weighting $\hat{\mathbf{w}}$.

$$\hat{\mathbf{w}}[i] = \sum_j \mathbf{w}[j] \mathbf{s}(i - j)$$



Data Structure and Accessors

The combination of addressing mechanisms allows the controller to interact with the memory in several distinct modes, corresponding to different **data structures** and **accessors**.

Content key only — memory is accessed like an **associative map**

Content and location — key finds an **array**, shift **indexes** into it

Content key only — memory is accessed like an **associative map**

Content and location — key finds an **array**, shift **indexes** into it

Location only — shift **iterates** from the last focus

Reading and Writing

Once the weightings are defined, each **read head** returns a **read vector** \mathbf{r} as input to the controller at the next timestep

$$\mathbf{r} = \sum_i \mathbf{w}[i] \mathbf{M}[i]$$

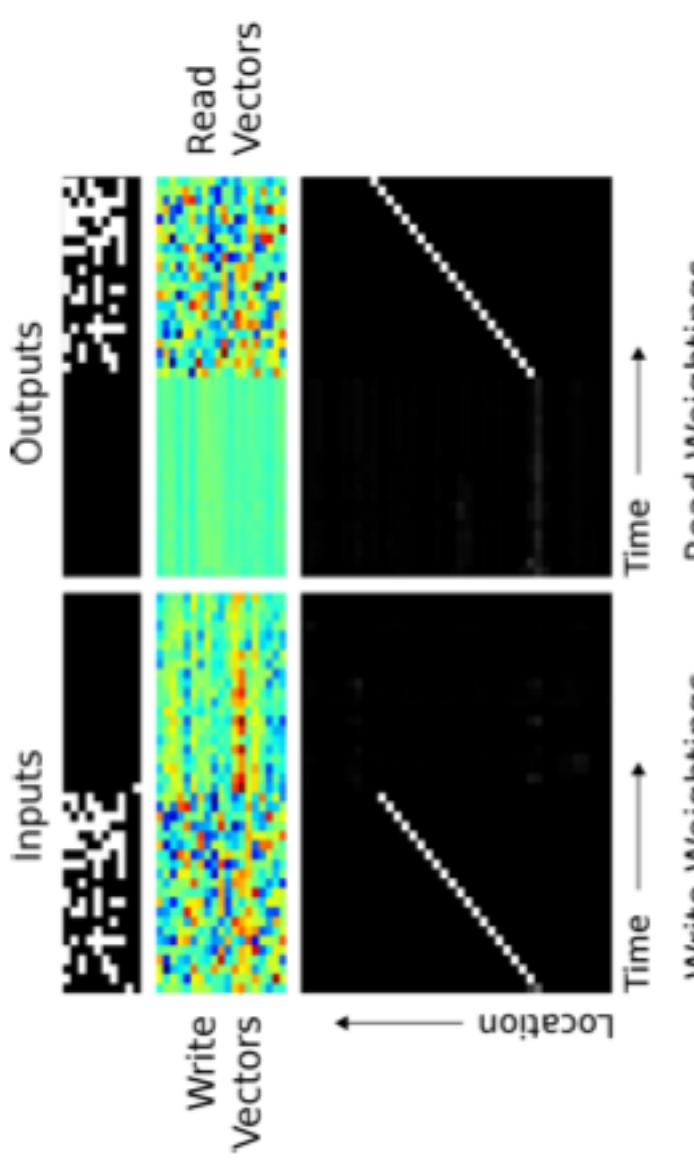
Each **write head** receives an **erase vector** \mathbf{e} and an **add vector** \mathbf{a} from the controller and resets then writes to modify the memory (like **LSTM**)

$$\mathbf{r} = \sum_i \mathbf{w}[i] \mathbf{M}[i]$$

Each **write head** receives an **erase vector** \mathbf{e} and an **add vector** \mathbf{a} from the controller and resets then writes to modify the memory (like LSTM)

$$\mathbf{M}[i] \leftarrow \mathbf{M}[i](1 - \mathbf{w}[i]\mathbf{e}) + \mathbf{w}[i]\mathbf{a}$$

The NTM Copy Algorithm



```
initialize: move head to start location
while input delimiter not seen do
    receive input vector
    write input to head location
    increment head location by 1
end while
return head to start location
while true do
    read output vector from head location
    emit output
    increment head location by 1
end while
```

```
increment head location by 1
```

```
end while
```

```
return head to start location
```

```
while true do
```

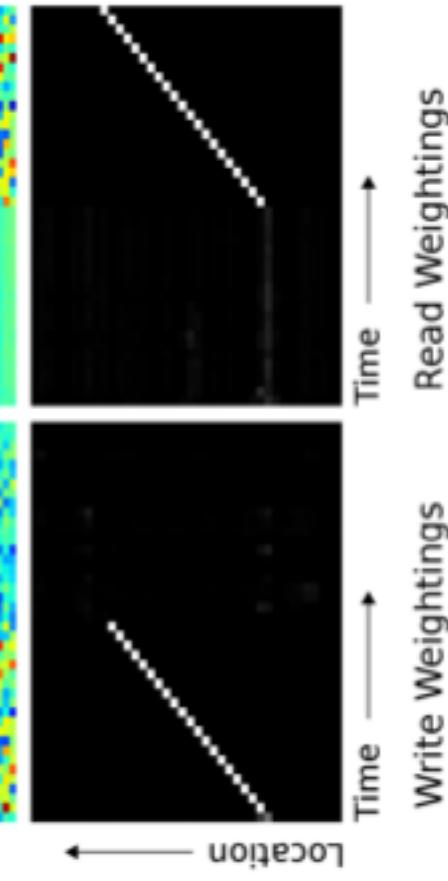
```
read output vector from head location
```

```
emit output
```

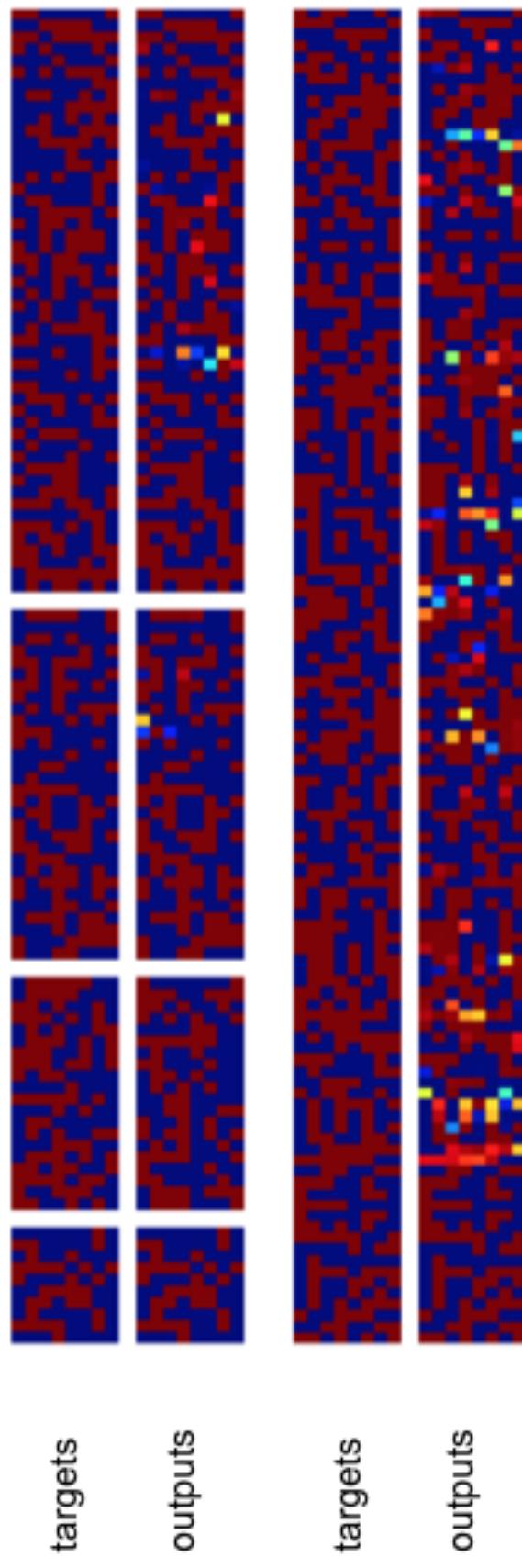
```
increment head location by 1
```

```
end while
```

pseudocode



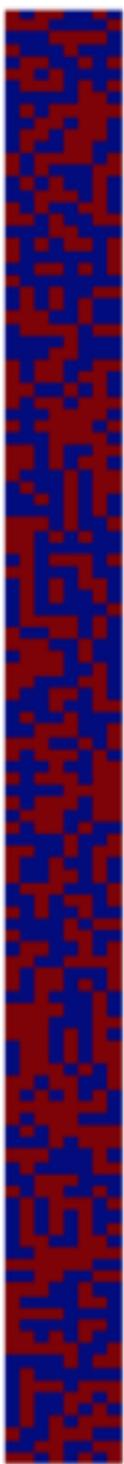
Copy Generalisation: length 10 to 120



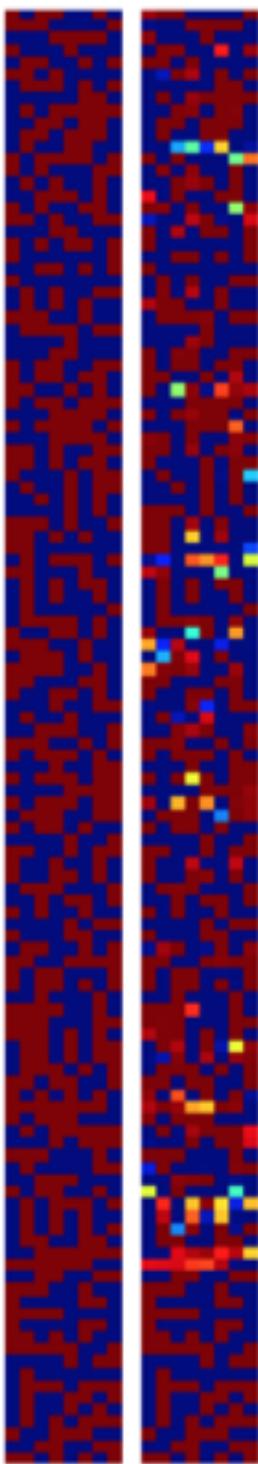
outputs



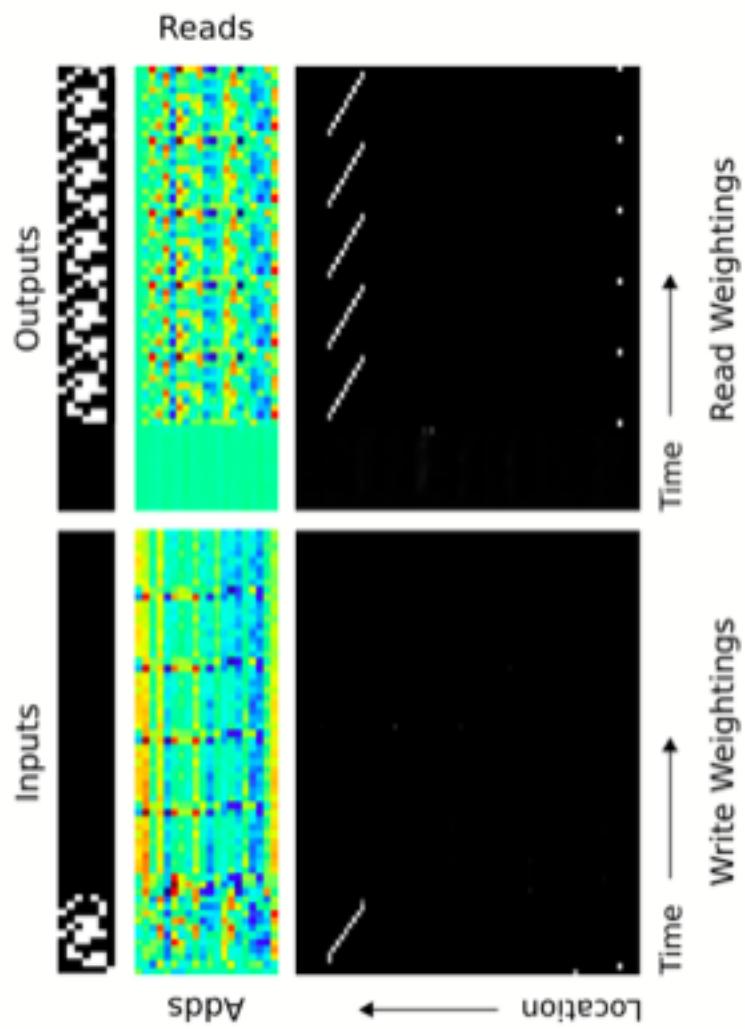
targets



outputs



Copy N Times

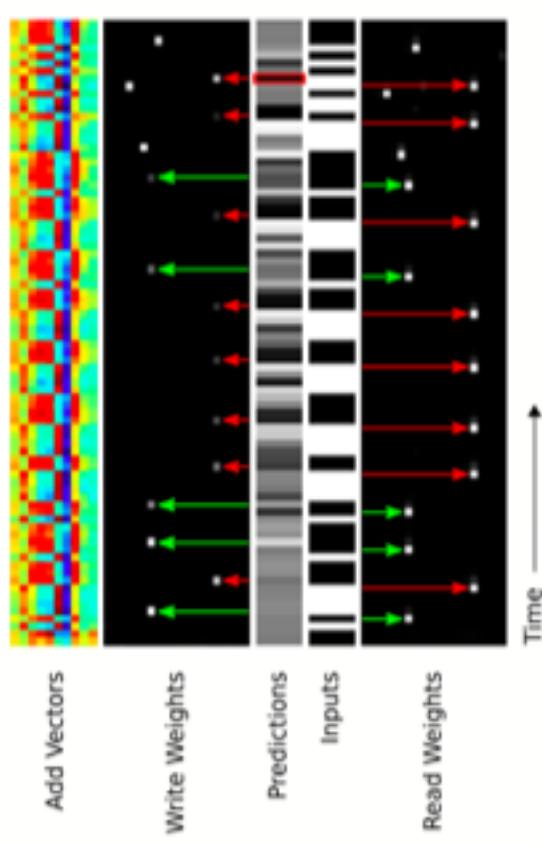
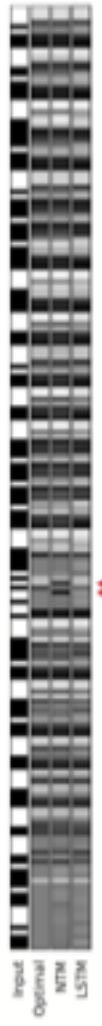


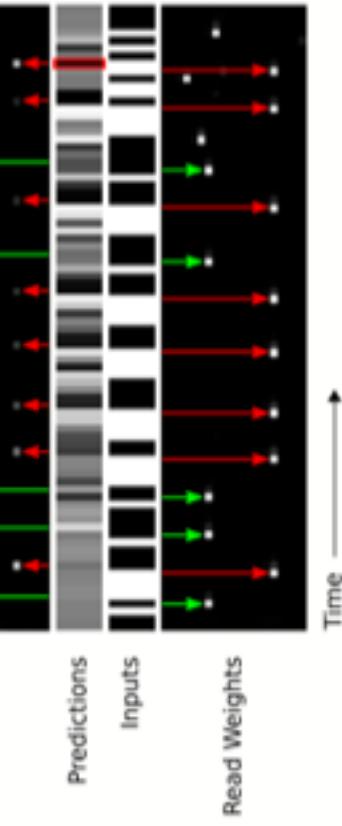


Time → Write Weightings Time → Read Weightings

NTM learns its first **for-loop**, using **content to jump, iteration to step**, and a **variable to count** to N .

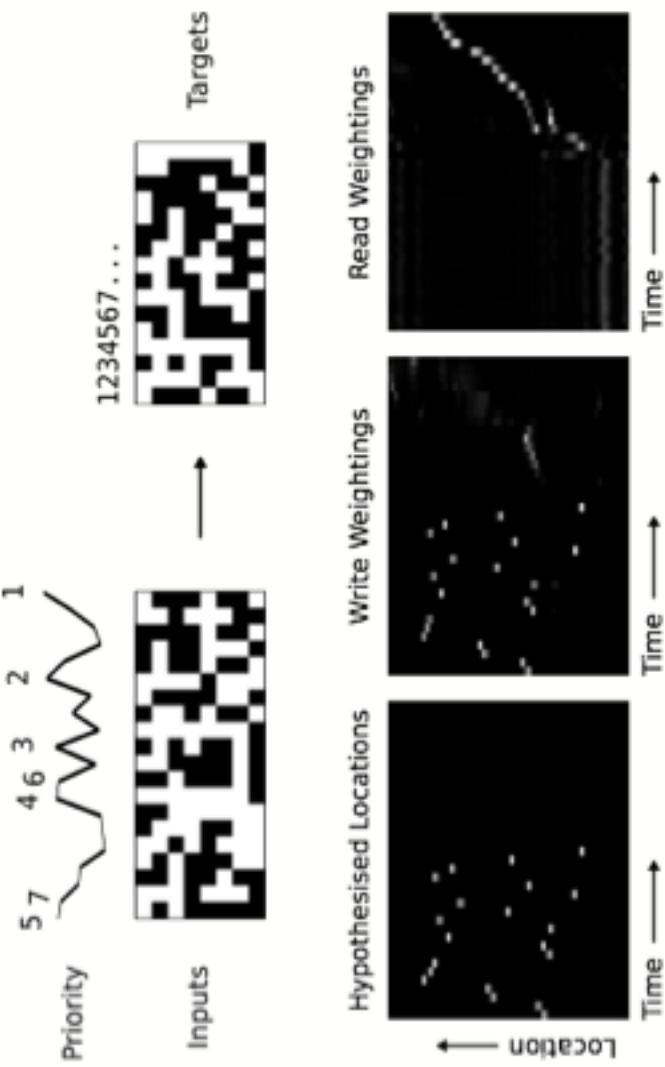
N-Gram Inference



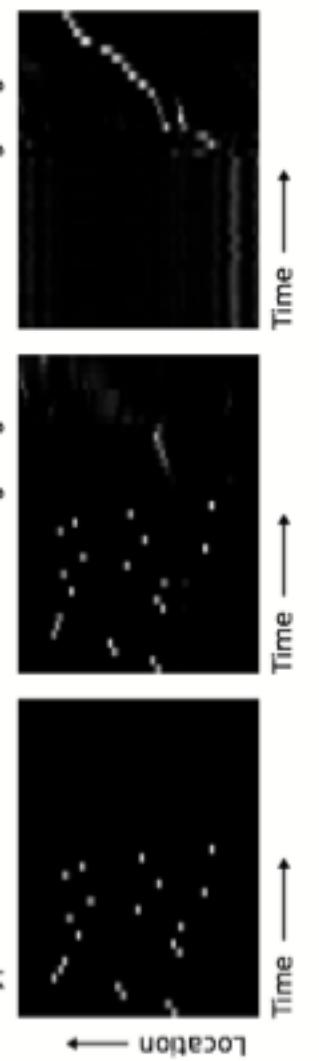


Specific memory locations store **variables** that **count the occurrences** of particular N-Grams

Priority Sort



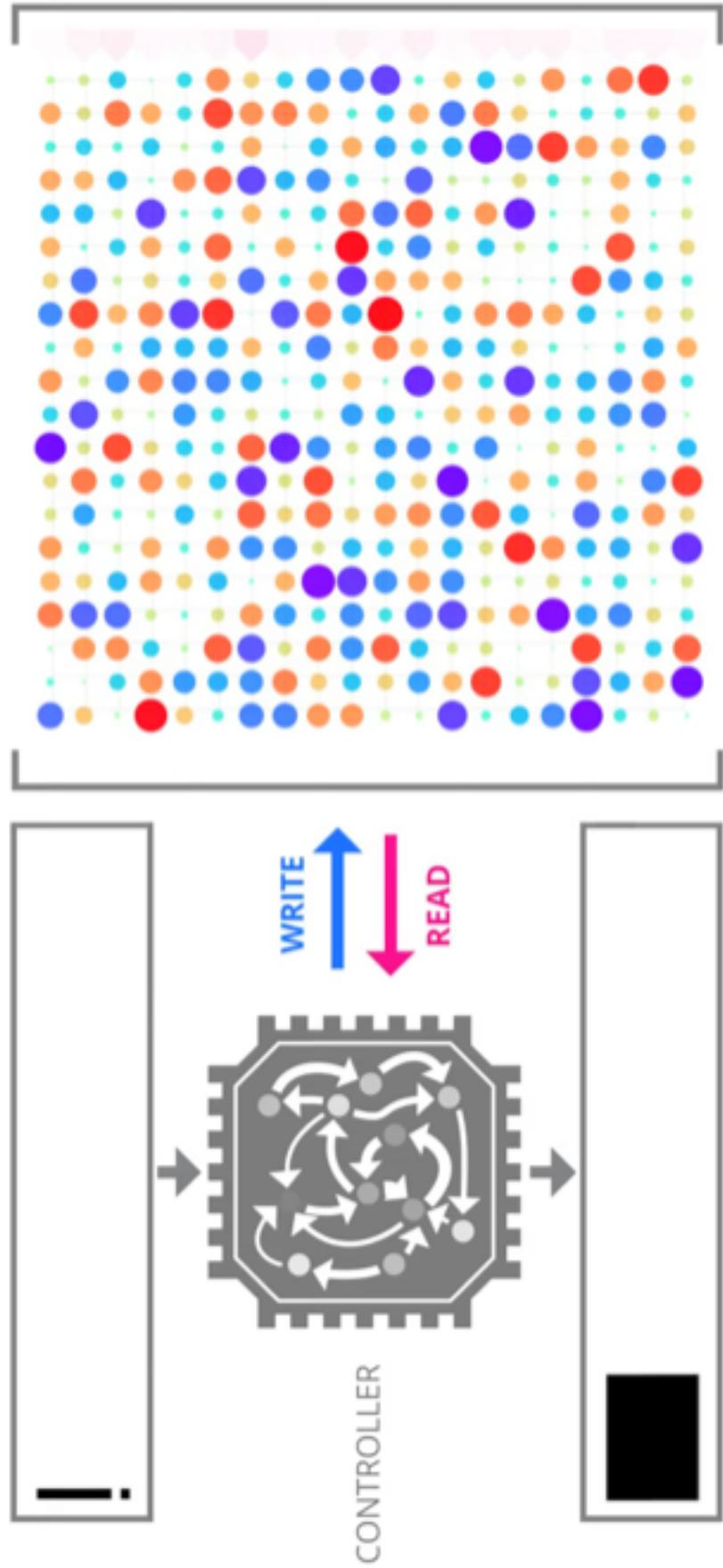
The network maps from **priorities** to **write locations**

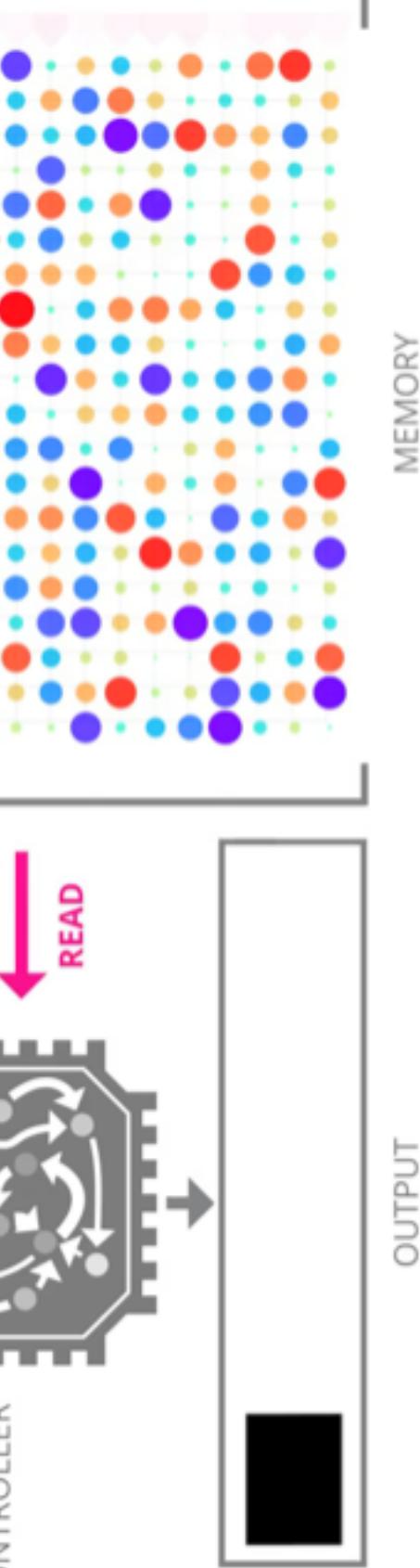


The network maps from **priorities** to **write locations**,
then **iterates** through the memory to return the sorted list

TRAINING

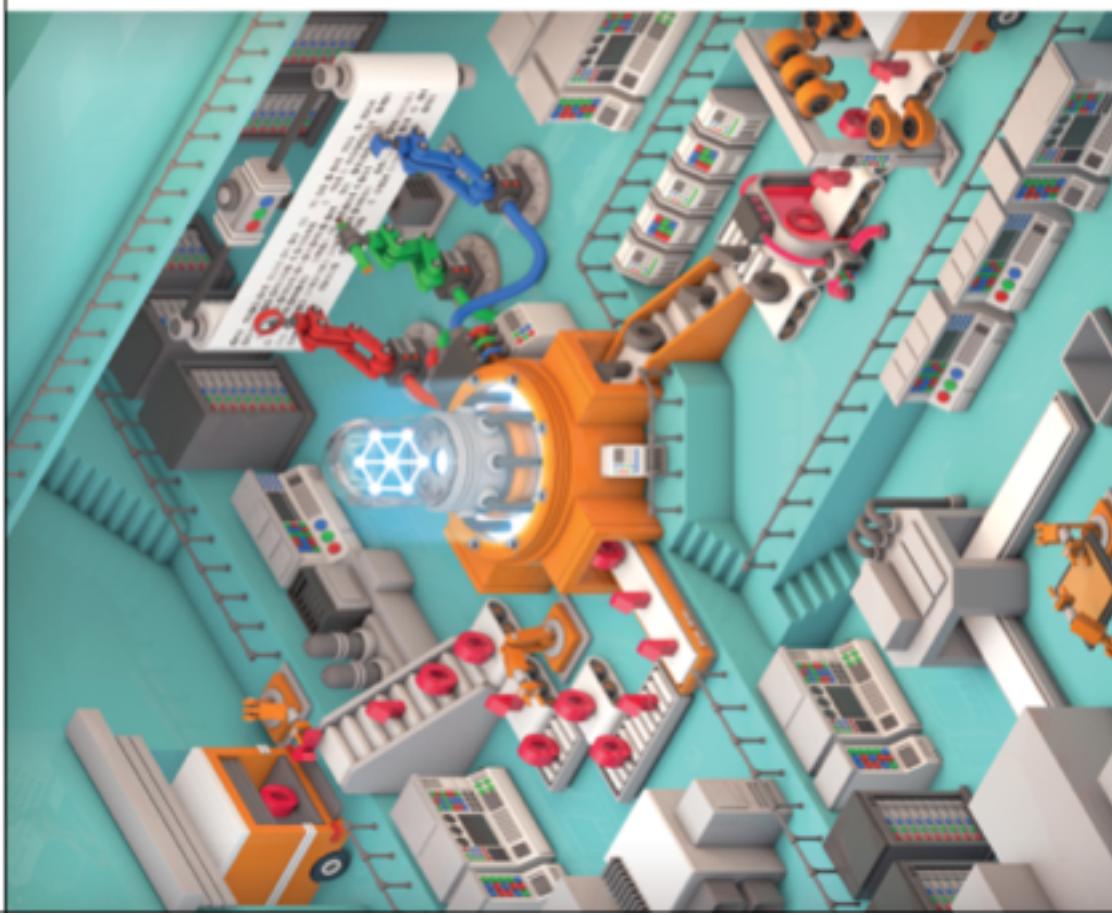
INPUT





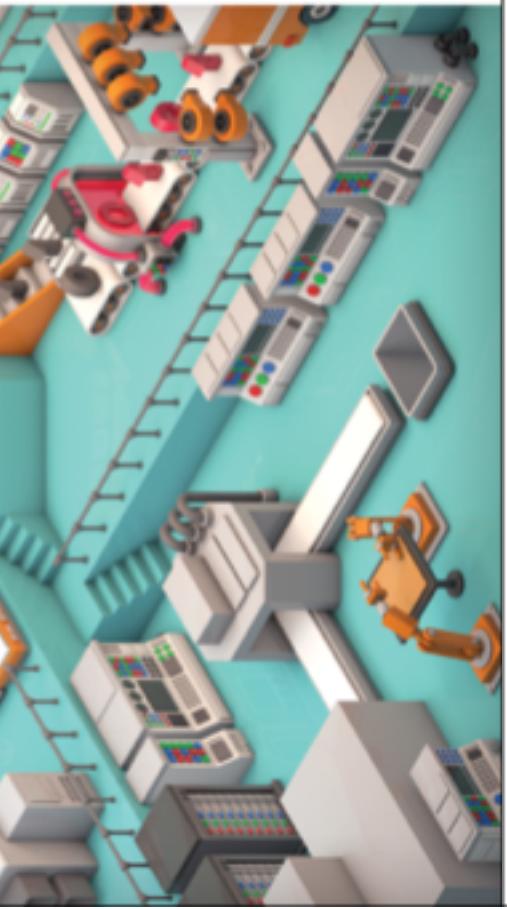
Differentiable Neural Computers

DNC is a successor architecture to
Neural Turing Machines with new
attention mechanisms for memory
access

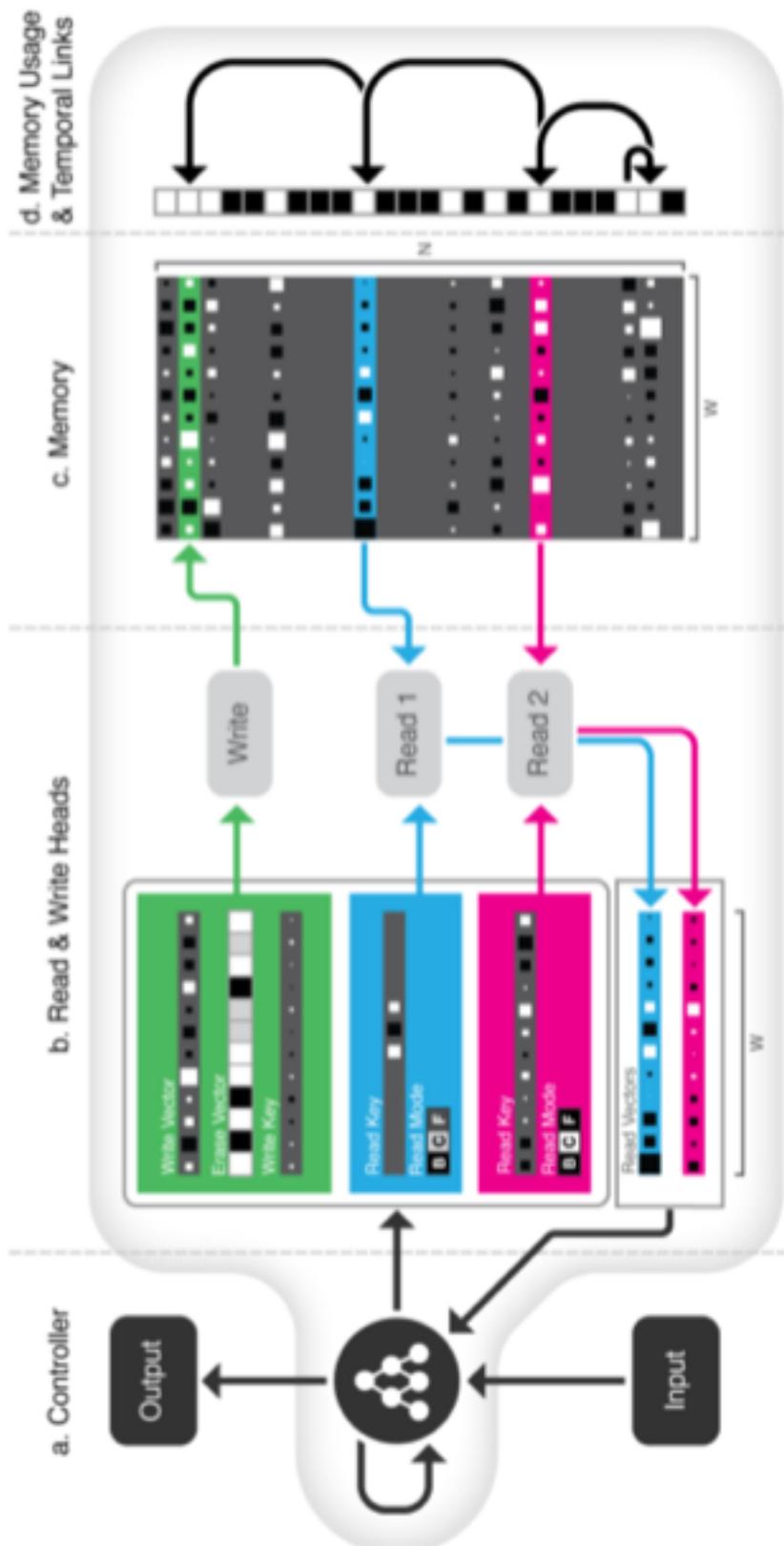


hybrid computing using a neural network with dynamic external memory, Graves et. al. (2016)

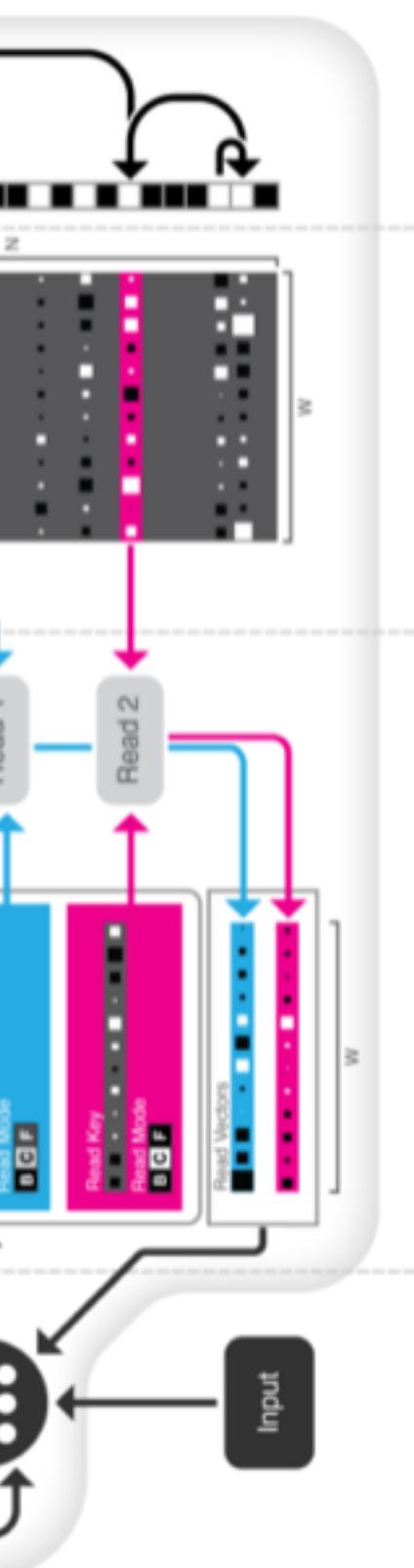
access



Overall Architecture



hybrid computing using a neural network with dynamic external memory, Graves et. al. (2016)



Allocating Memory

- NTM could only 'allocate' memory in contiguous blocks, leading to memory management problems
- DNC defines a differentiable *free list* tracking the ***usage*** (\mathbf{u}_t) of each memory location
- Usage is automatically *increased* after each write (\mathbf{w}^w_t) and optionally *decreased* after each read (\mathbf{w}^r_t) by ***free gates*** (f_t^i)
- The controller then uses an ***allocation gate*** (g^a_t) to interpolate between writing to a newly allocated (\mathbf{a}_t) location, or an existing one found by content (\mathbf{c}^w_t)

$$\mathbf{u}_t = (\mathbf{u}_{t-1} + \mathbf{w}_{t-1}^w - \mathbf{u}_{t-1} \circ \mathbf{w}_{t-1}^w) \circ \prod_{i=1}^R (1 - f_t^i \mathbf{w}_{t-1}^{r,i})$$

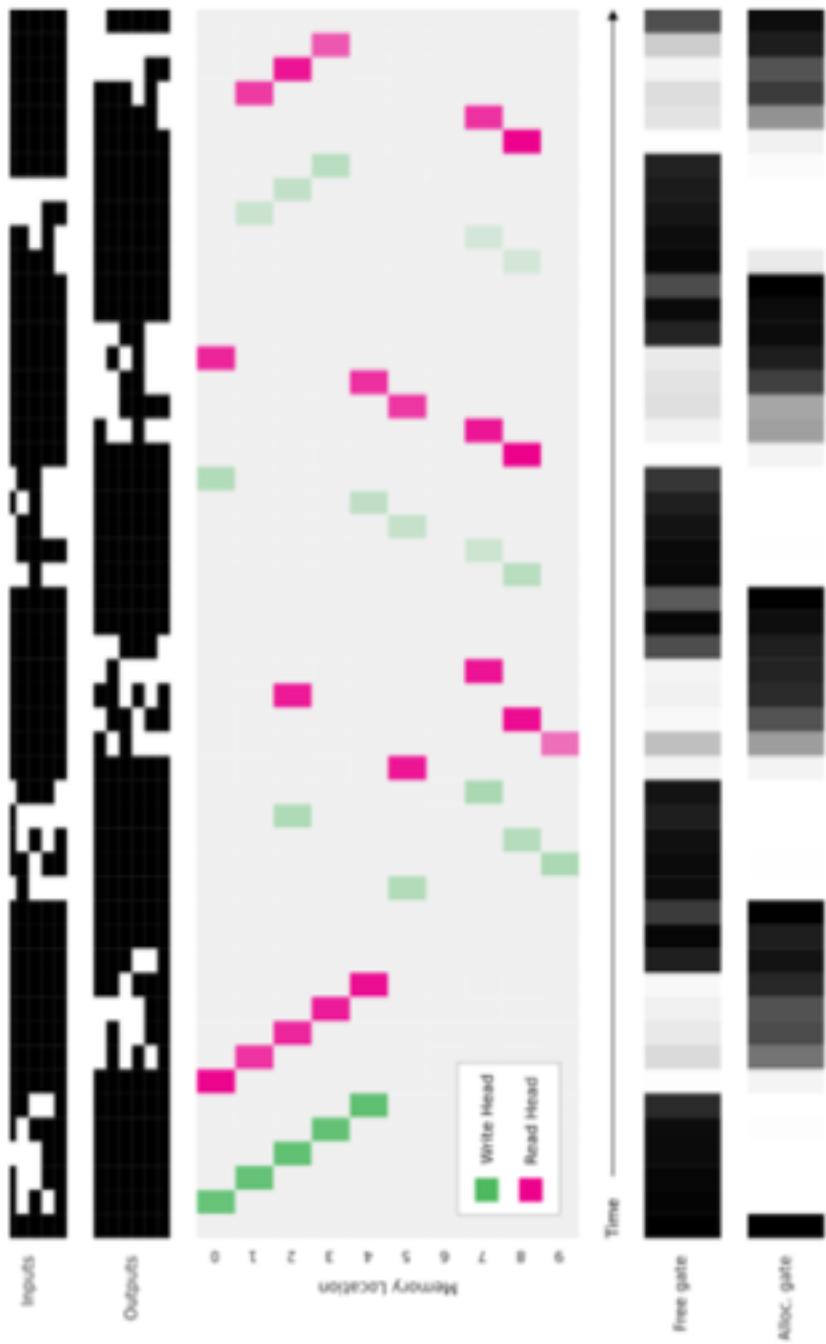
$$\mathbf{w}_t^w = g_t^w (g_t^a \mathbf{a}_t + (1 - g_t^a) \mathbf{c}_t^w)$$

$$\mathbf{u}_t = (\mathbf{u}_{t-1} + \mathbf{w}_{t-1}^{\text{W}} - \mathbf{u}_{t-1} \circ \mathbf{w}_{t-1}^{\text{W}}) \circ \prod_{i=1}^K (1 - f_t^i \mathbf{w}_{t-1}^{\text{r},i})$$

- The controller then uses an **allocation gate** (g_t^a) to interpolate between writing to a newly allocated (\mathbf{a}_t) location, or an existing one found by content (\mathbf{c}_t^w)

$$\mathbf{w}_t^w = g_t^w (g_t^a \mathbf{a}_t + (1 - g_t^a) \mathbf{c}_t^w)$$

Memory Allocation Test





Searching By Time

- NTM was able to retrieve memories in order of their *index* but not in order in which they were written
- Preserving temporal order is necessary for many tasks (e.g. sequence of instructions) and appears to play an important role in human cognition
- We wanted DNC to be able to iterate through memories in the order they were written (or rewritten)
- A **precedence weighting** (\mathbf{p}_t) keeps track of which locations were most recently written to:

$$\mathbf{p}_t = \left(1 - \sum_i \mathbf{w}_t^w[i] \right) \mathbf{p}_{t-1} + \mathbf{w}_t^w$$

- A **precedence weighting** (\mathbf{p}_t) keeps track of which locations were most recently written to:

$$\mathbf{p}_t = \left(1 - \sum_i \mathbf{w}_t^{\text{w}}[i] \right) \mathbf{p}_{t-1} + \mathbf{w}_t^{\text{w}}$$

Searching By Time

\mathbf{p}_t is then used to update a **temporal link matrix** (L_t), defining the order locations were written in:

$$L_t[i, j] = (1 - \mathbf{w}_t^{\text{w}}[i] - \mathbf{w}_t^{\text{w}}[j]) L_{t-1}[i, j] + \mathbf{w}_t^{\text{w}}[i] \mathbf{p}_{t-1}[j]$$

The controller can use L_t to retrieve the write before (b'_i) or after (f'_i) the last read location ($\mathbf{w}^{r,i}_{t-1}$), allowing it to iterate forwards or backwards in time

$$\mathbf{b}_t^i = \hat{L}_t^\top \mathbf{w}_{t-1}^{r,i} \quad f_t^i = \hat{L}_t \mathbf{w}_{t-1}^{r,i}$$

Finally three-way gates ($\boldsymbol{\pi}^i$) are used to interpolate among iterating forwards, backwards, or reading by content:

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

$$\mathbf{b}_t^i = \hat{L}_t \mathbf{w}_{t-1}^{r,i} \quad f_t^i = \hat{L}_t \mathbf{w}_{t-1}^{r,i}$$

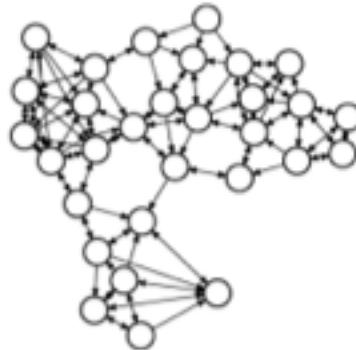
Finally three-way gates (π_t^i) are used to interpolate among iterating forwards, backwards, or reading by content:

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

Graph Experiments

Training Data

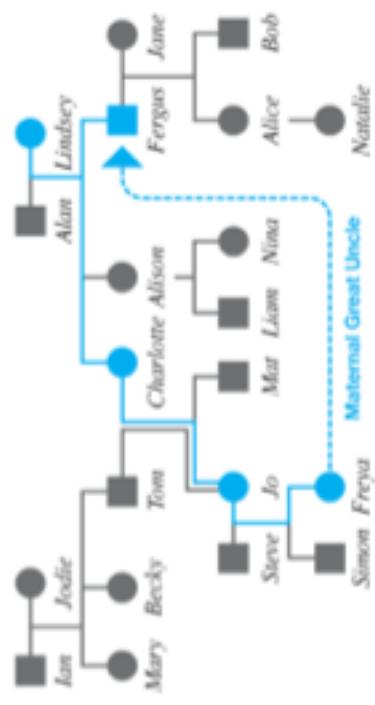
a. Random Graph



b. London Underground



c. Family Tree



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, NorthernHillGate, Central)
(OxfordCircus, Euston, Victoria)

- 84 edges in total

Shortest Path Question:

(Moorgate, PiccadillyCircus,)
(Moorgate, Bank, Northern)
(Bank, Holborn, Central)
(Holborn, LeicesterSq, Piccadilly)
(LeicesterSq, PiccadillyCircus, Piccadilly)
...
(OxfordCircus, NottingHillGate, Central)
(NottingHillGate, Paddington, Circle)
...
(Embankment, Waterloo, Bakerloo)
(Waterloo, GreenPark, Jubilee)

Answer:

(Fergus, ... MaternalGreatUncle)
(Fergus, ... MaternalUncle)

Answer:

(Fergus, Alice, Mother)
(Alice, Ian, Father)
(Ian, Alice, Daughter)
(Mat, Charlotte, Mother)

Family Tree Input:

(Charlotte, Alan, Father)
(Simon, Steve, Father)
(Steve, Simon, Son1)
(Melanie, Alison, Mother)
(Lindsey, Fergus, Son1)
...
(Bob, Jane, Mother)
(Natalie, Alice, Mother)
(Mary, Ian, Father)
(Lane, Alice, Daughter)
(Mat, Charlotte, Mother)
- 54 edges in total

Inference Question:

{Fergus, ... MaternalGreatUncle}

Answer:

(Fergus, Alice, Mother)
(Alice, Ian, Father)
(Ian, Alice, Daughter)

How DNC Reads Graphs

Answer:
 (OxfordCircus, CharingCross, Northern)
 (TottenhamCtRd, LeicesterSq, Northern)
 (OxfordCircus, PiccadillyCircus, Bakerloo)
 (OxfordCircus, Northern/HGate, Central)
 (BakerSt, Marybone, Circle)
 (BakerSt, Marylebone, Bakerloo)
 (BakerSt, OxfordCircus, Bakerloo)
 ...
 (LeicesterSq, CharingCross, Northern)
 (TottenhamCtRd, LeicesterSq, Northern)
 (OxfordCircus, PiccadillyCircus, Bakerloo)
 (OxfordCircus, Northern/HGate, Central)
 (BakerSt, Marybone, Circle)
 (BakerSt, Marylebone, Bakerloo)
 (BakerSt, OxfordCircus, Bakerloo)
 ...
 - 84 edges in total!

Answer:
 (OxfordCircus, NottingHGate, Central)
 (NottingHGate, Paddington, Circle)
 (BakerSt, Victoria, Victoria)
 (Victoria, Bakerloo, Circle)
 (Victoria, Jubilee, Jubilee)

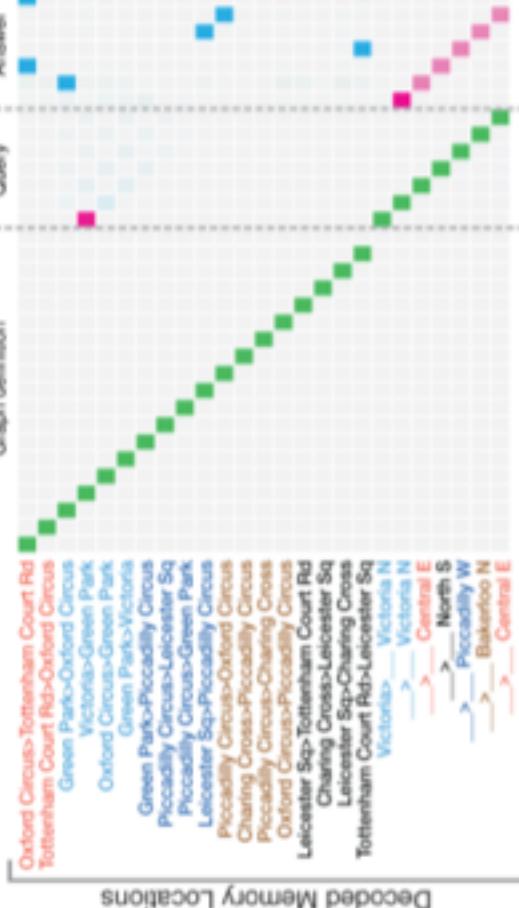
Answer:
 (Bob, Jane, Mother)
 (Natalie, Alice, Mother)
 (Mary, Ian, Father)
 (Liam, Alice, Daughter 1)
 (Melanie, Alison, Mother)
 (Lindsey, Ferguson, Son 1)
 ...
 (Fergie, ... Maternal/GreatUncle)

Answer:
 (Moorgate, Bank, Northern)
 (Bank, Holborn, Central)
 (Holborn, LeicesterSq, Piccadilly)
 (LeicesterSq, Piccadilly/Circus, Piccadilly)
 ...
 - 54 edges in total!

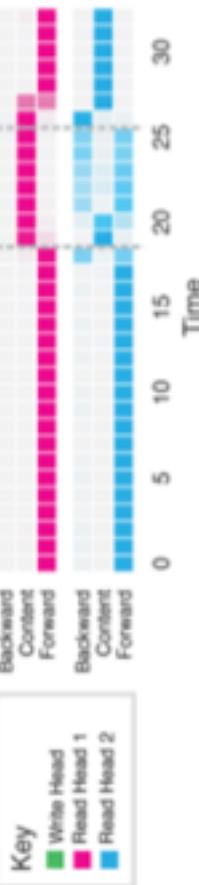
Answer:
 (OxfordCircus, NottingHGate, Central)
 (NottingHGate, Paddington, Circle)

Answer:
 (Fergie, Ferguson, Maternal/GreatUncle)

a. Read and Write Weightings



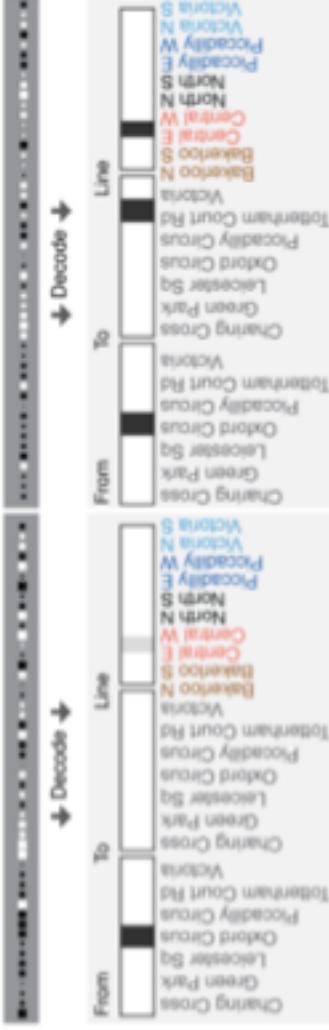
b. Read Mode



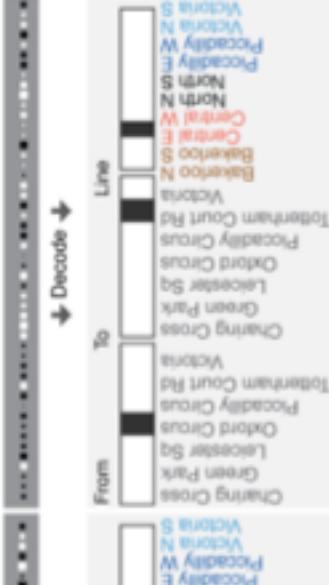
c. London Underground Map

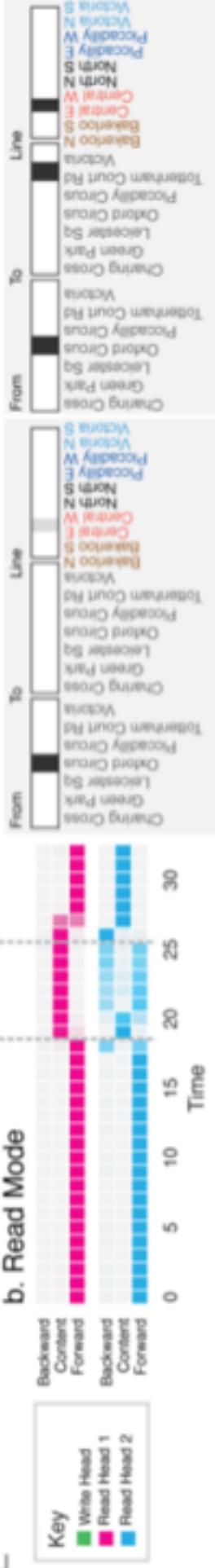


d. Read Key



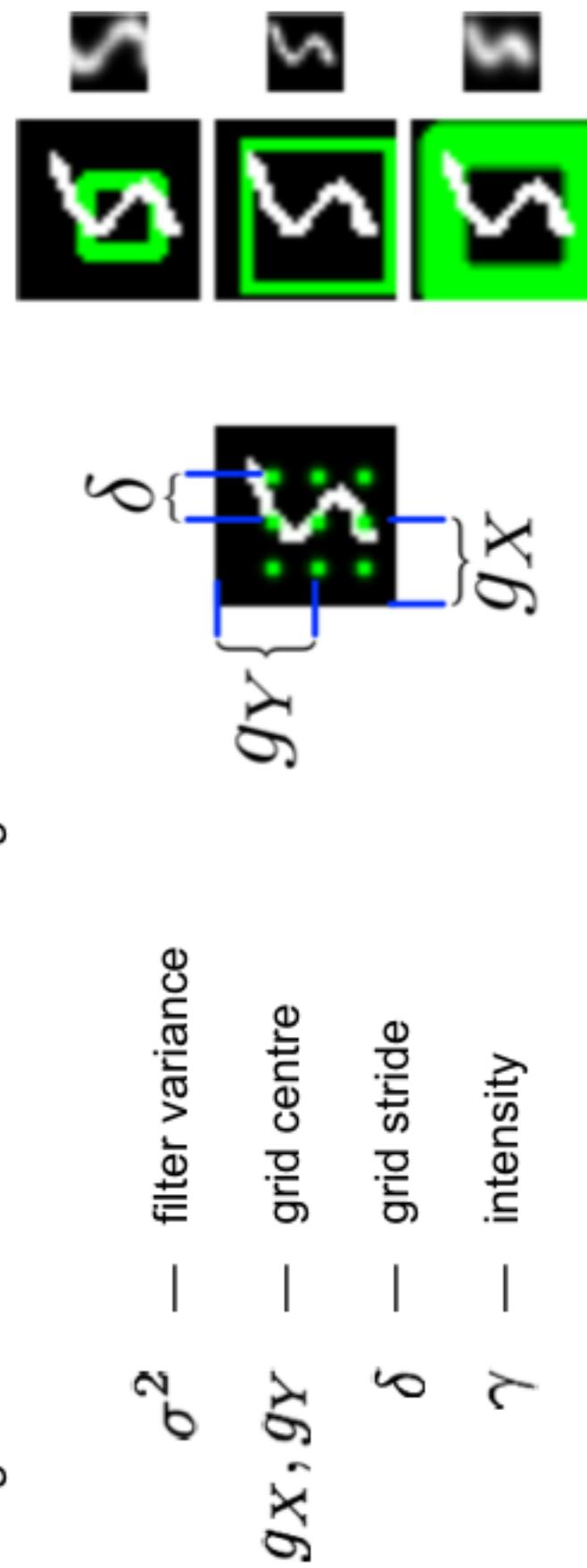
e. Location Content





Differentiable Visual Attention

DRAW (Gregor et. al. 2015) uses a grid of Gaussian filters to **read** from input images and **draw** to a **canvas** image:



Reading MNIST

γ — intensity

$\tilde{g_X}$

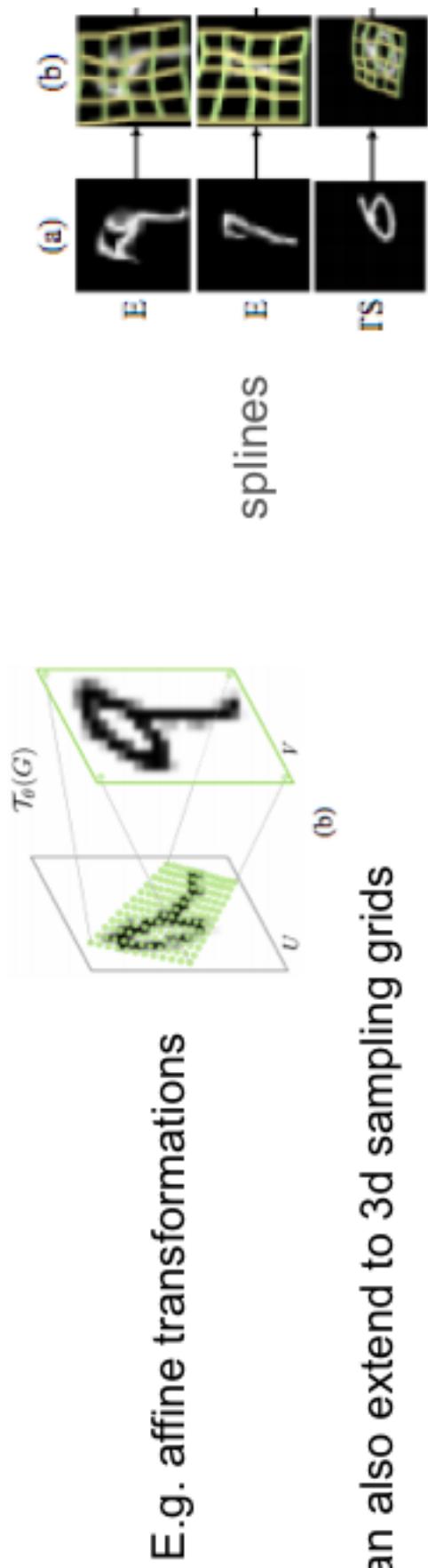


5

Reading MNIST

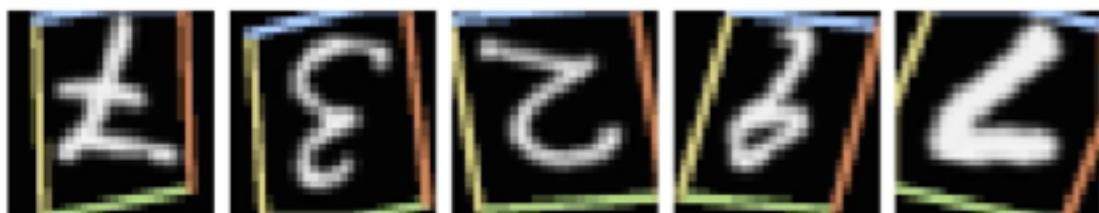
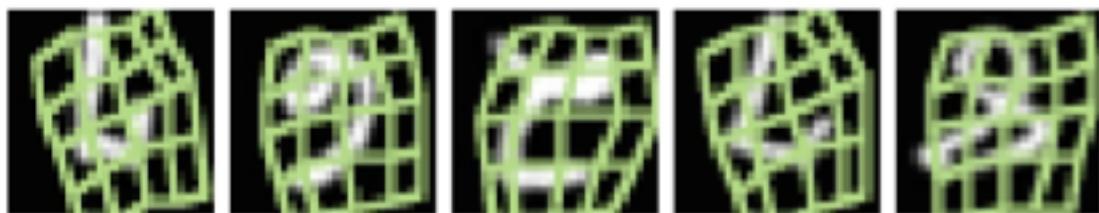
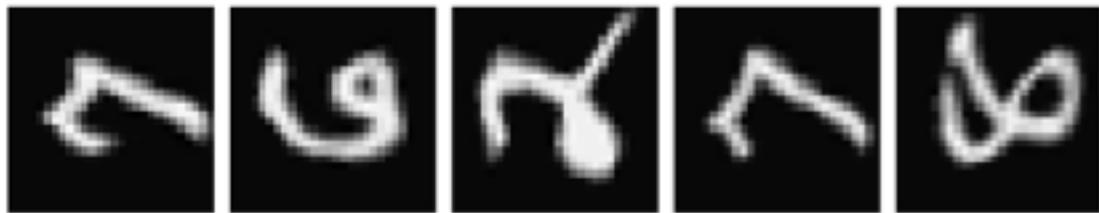
Spatial Transformers

The attention mechanism in **Spatial Transformer Networks** (Jaderberg et. al. 2015) extends DRAW by allowing more complex transformations of the **sampling grid**

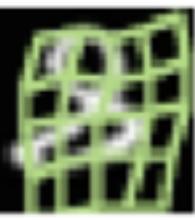


Can also extend to 3d sampling grids

RS → σ



8

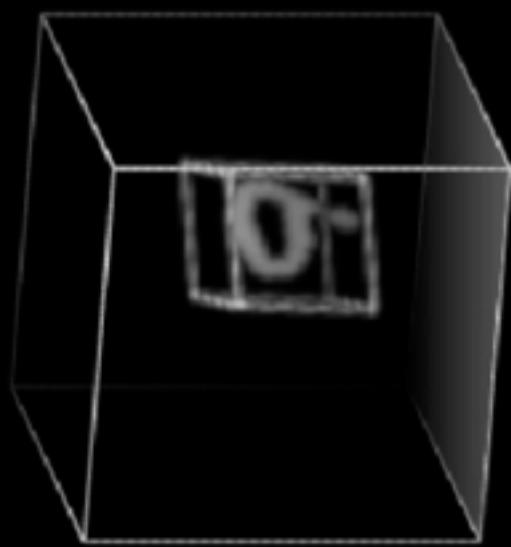
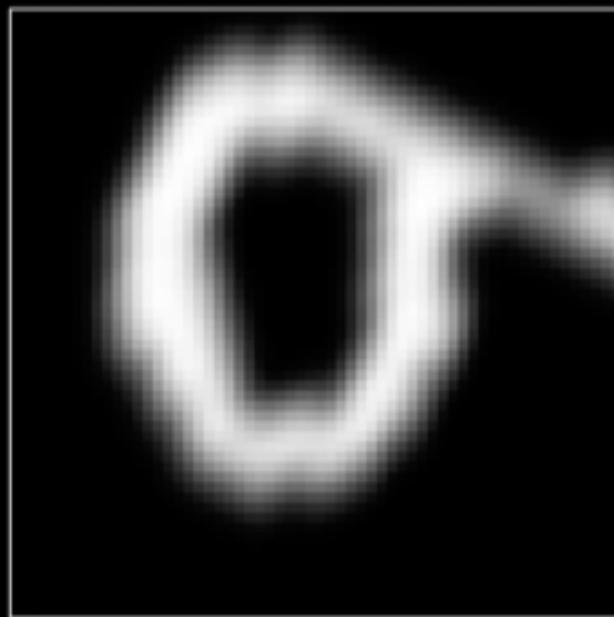
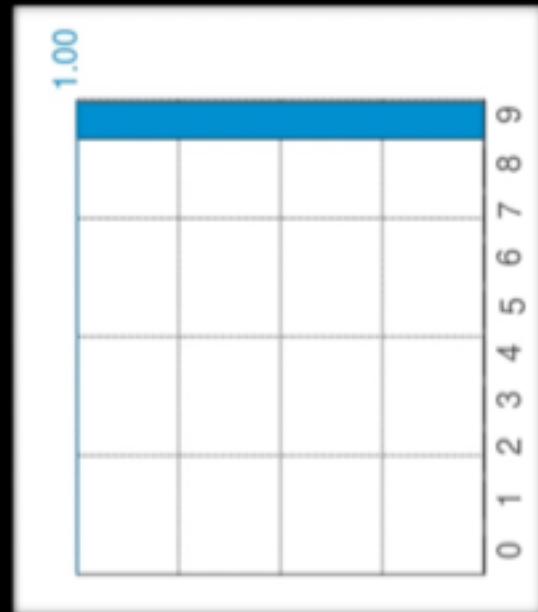


9

7



6



Summary

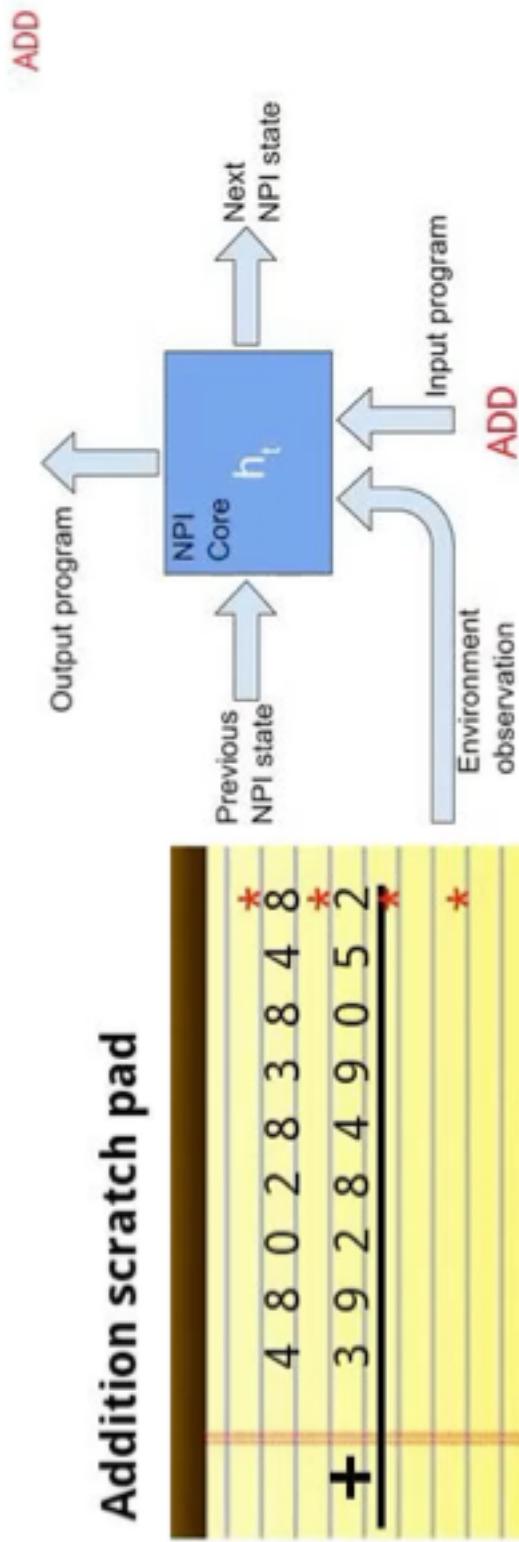
Summary

- Selective attention appears to be as useful for deep learning as it is for people
- Neural nets always have **implicit attention**, but we can also add **explicit attention** mechanisms
- These can be **stochastic** and trained with **reinforcement learning**
- Or **differentiable** and trained with ordinary **backprop**
- We can use attention to decide where to **write**, as well as where to **read**
- Many types of neural attention model (**content**, **location**, **visual**, **temporal**...) have been defined, and more are appearing all the time

been defined, and more are appearing all the time

NPI inference

Generated commands



Neural Programmer-Interpreters, Reed and de Freitas (2015)