WELCOME TO THE

# UCL x DeepMind lecture series

In this lecture series, leading research scientists from leading AI research lab, DeepMind, will give 12 lectures on an exciting selection of topics in Deep Learning, ranging from the fundamentals of training neural networks via advanced ideas around memory, attention, and generative modelling to the important topic of responsible innovation.

Please join us for a deep dive lecture series into Deep Learning!

**#UCLxDeepMind**

# General information

**Exits:**

At the back, the way you came in

**Wifi:**

UCL guest

TODAY'S SPEAKER

# Alex Graves

Alex Graves completed a BSc in Theoretical Physics at the University of Edinburgh, Part III Maths at the University of Cambridge and a PhD in artificial intelligence at IDSIA with Jürgen Schmidhuber, followed by postdocs at the Technical University of Munich and with Geoffrey Hinton at the University of Toronto. He is now a research scientist at DeepMind. His contributions include the Connectionist Temporal Classification algorithm for sequence labelling (widely used for commercial speech and handwriting recognition), stochastic gradient variational inference, and the Neural Turing Machine / Differentiable Neural Computer architectures

TODAY'S LECTURE

# Attention and Memory in Deep Learning

Attention and memory have emerged as two vital new components of deep learning over the last few years. This lecture covers a broad range of attention mechanisms, including the implicit attention present in any deep network, as well as both discrete and differentiable variants of explicit attention. It then discusses networks with external memory and explains how attention provides them with selective recall. It briefly reviews transformers, a particularly successful type of attention network, and lastly looks at variable computation time, which can be seen as a form of 'attention in time'.

DeepMind

# Attention and Memory in Deep Learning

Alex Graves

UCL x DeepMind Lectures

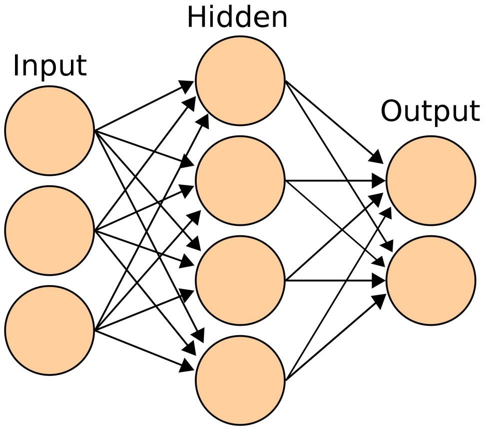
# 1 Introduction

# 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 (**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

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

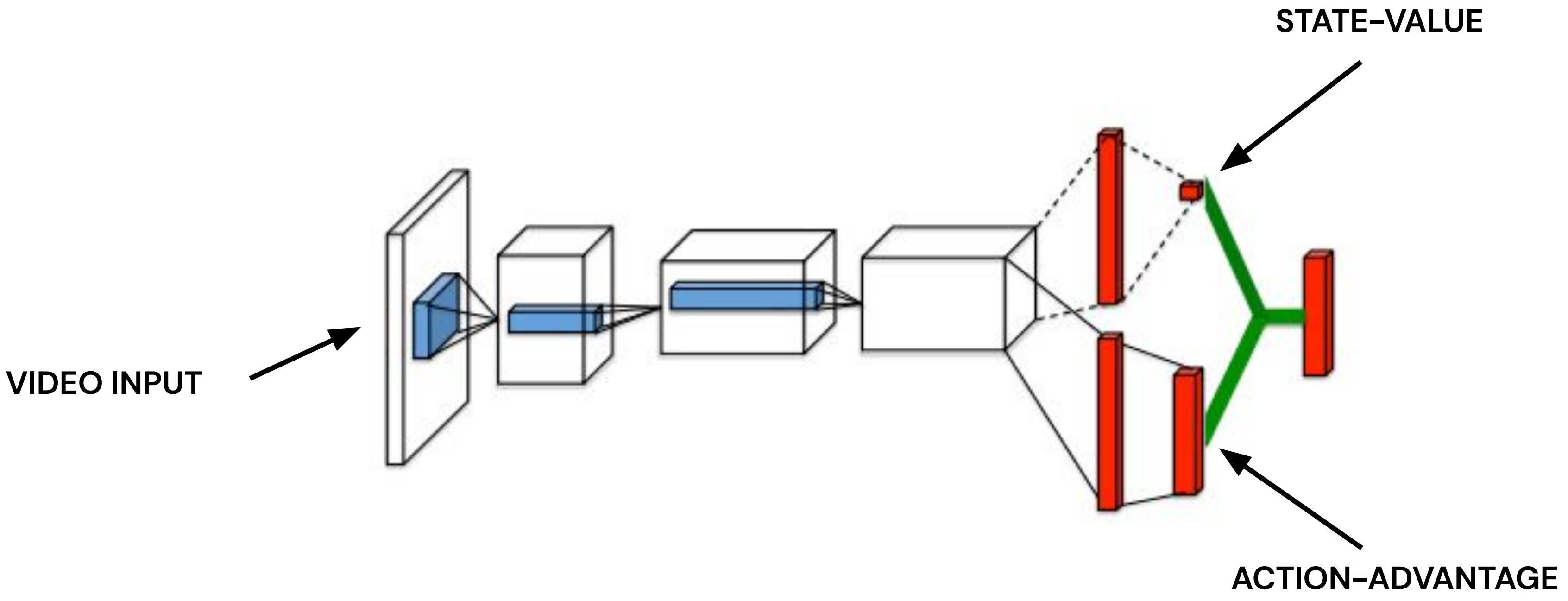
x = size k input vector
y = size m output vector
Jacobian J = m x k matrix

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

$$J_{ij} = \frac{\partial y_i}{\partial x_j}$$

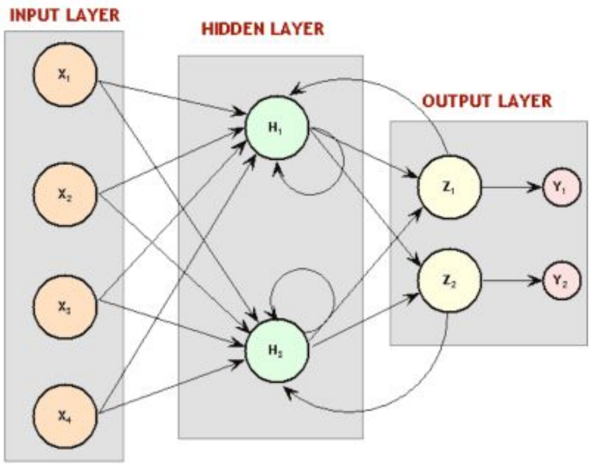
Can compute with ordinary backdrop
(just set output 'errors' = output activations)

# Jacobian in Action: Duelling Network




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

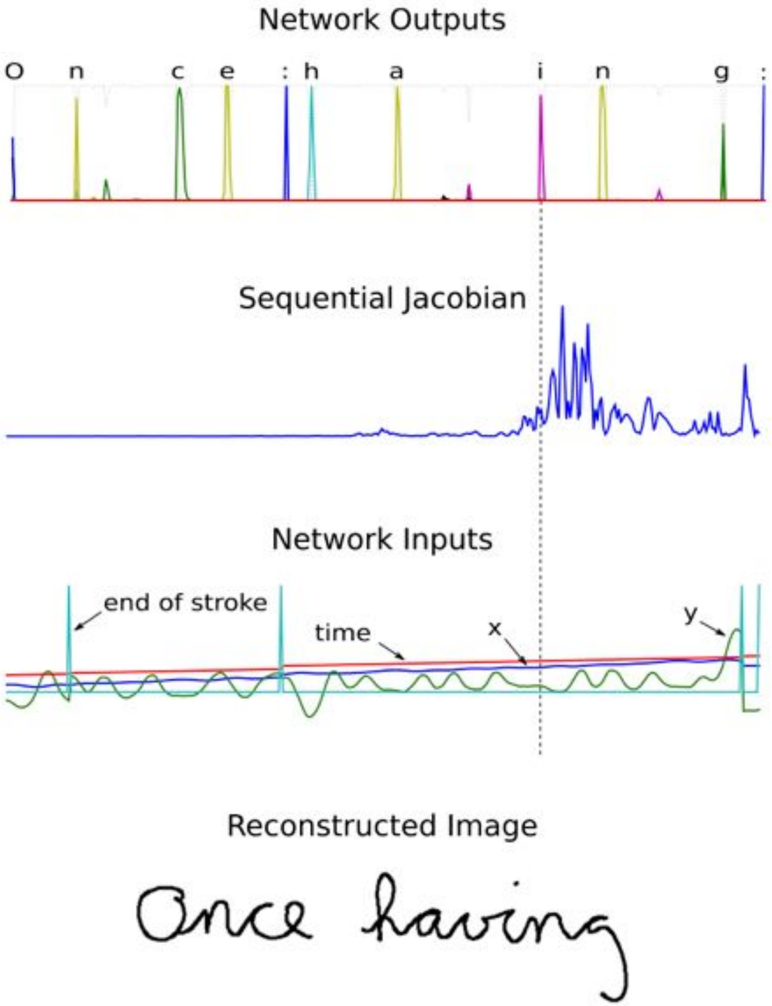
# Attention and memory in Recurrent Networks (RNNs)




"Hello World"

**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.



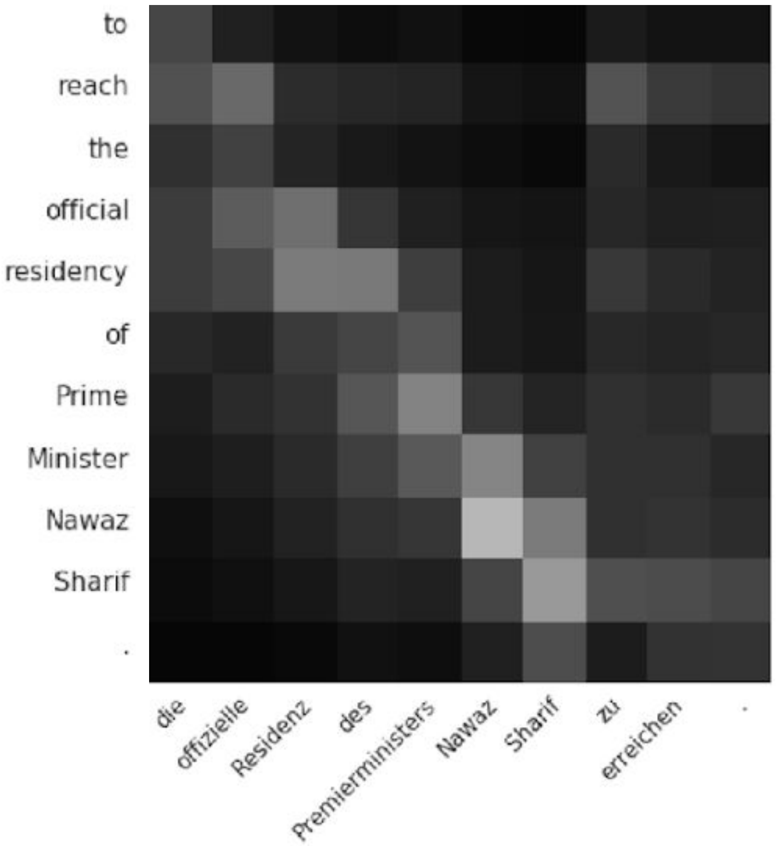

- The Sequential Jacobian is the set of derivatives of one network output with respect to all the inputs

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

- 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 translation:

*“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 some way:

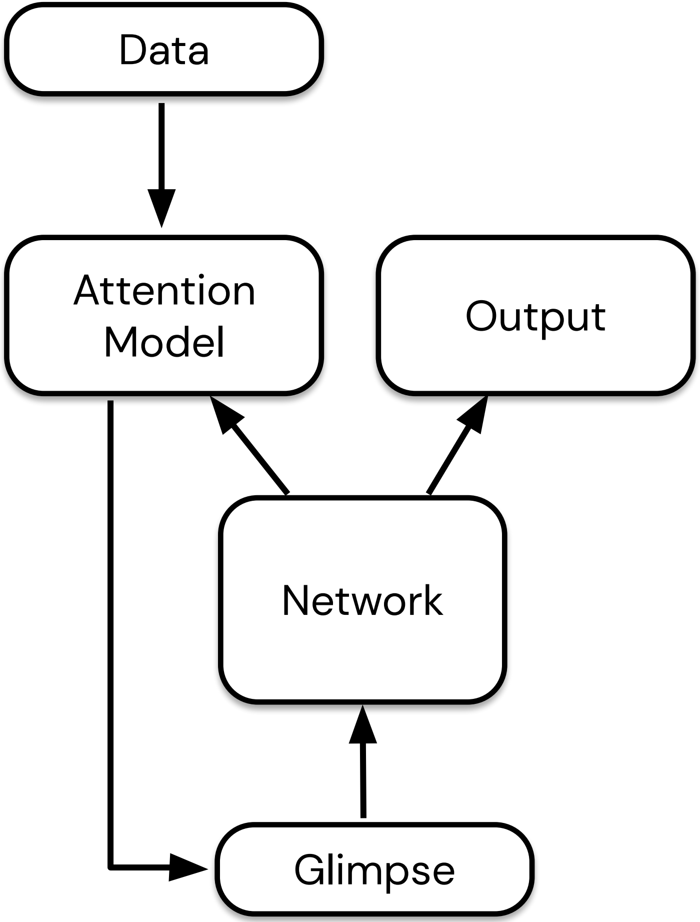
- Computational **efficiency**
- **Scalability** (e.g. fixed sized glimpse for any size image)
- **Sequential processing** of static data (e.g. moving gaze)
- **Easier to interpret**

# Neural Attention Models

The **network** produces and extra output vector used to parameterise an **attention model**

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

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



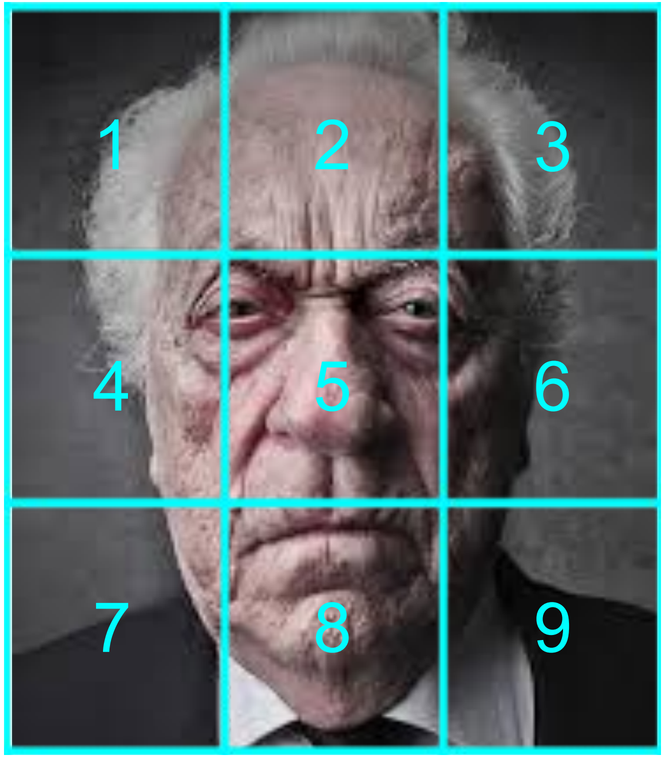
# Glimpse Distribution

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

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

simplest case: **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 **g** as a **stochastic policy** $\pi_a$, sample from it, and use **REINFORCE** (with reward $R$ = task loss $L$ induced by the glimpse) to train the attention model

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

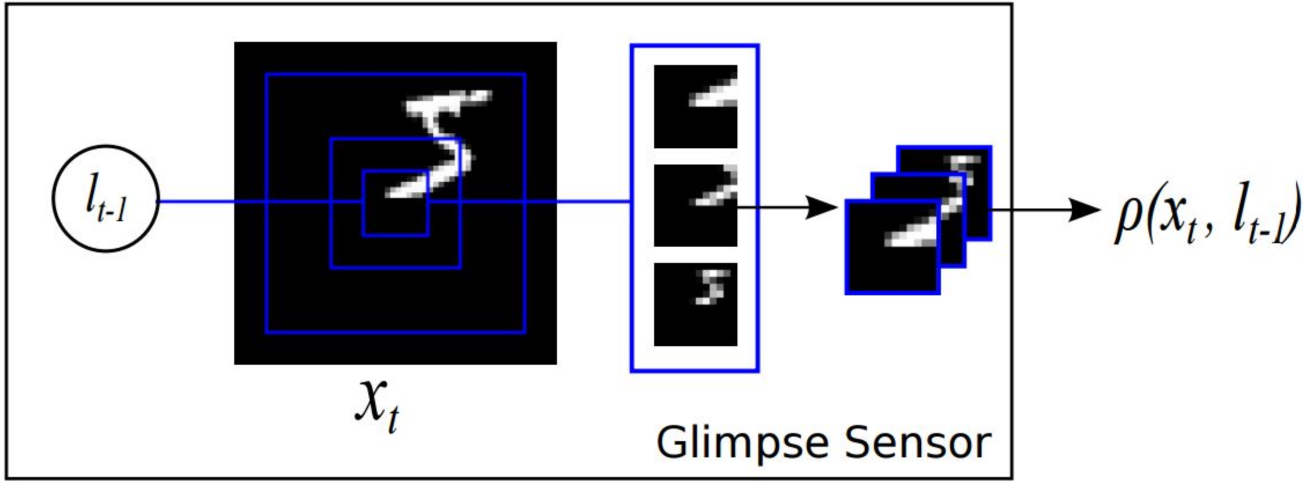
$$R = \mathbb{E}_{\mathbf{g} \sim \pi_{\mathbf{a}}} \left[ \log \pi_{\mathbf{a}} L(\mathbf{g}) \right]$$

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

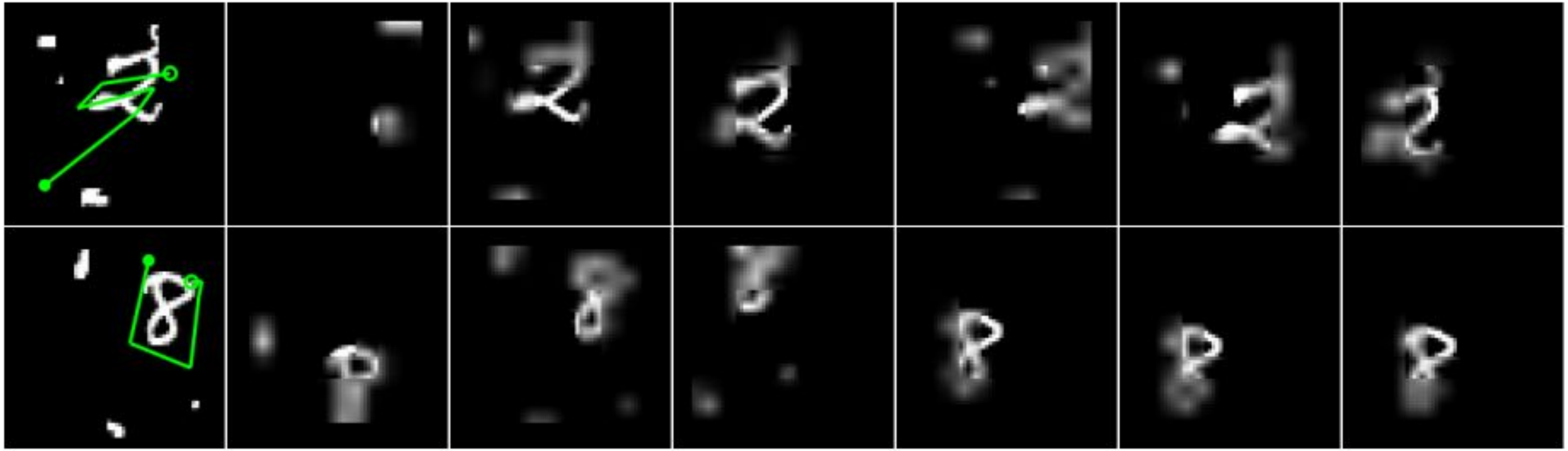
In general we can use **RL** methods for supervised tasks any time some module in the network is **non-differentiable**

# 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)

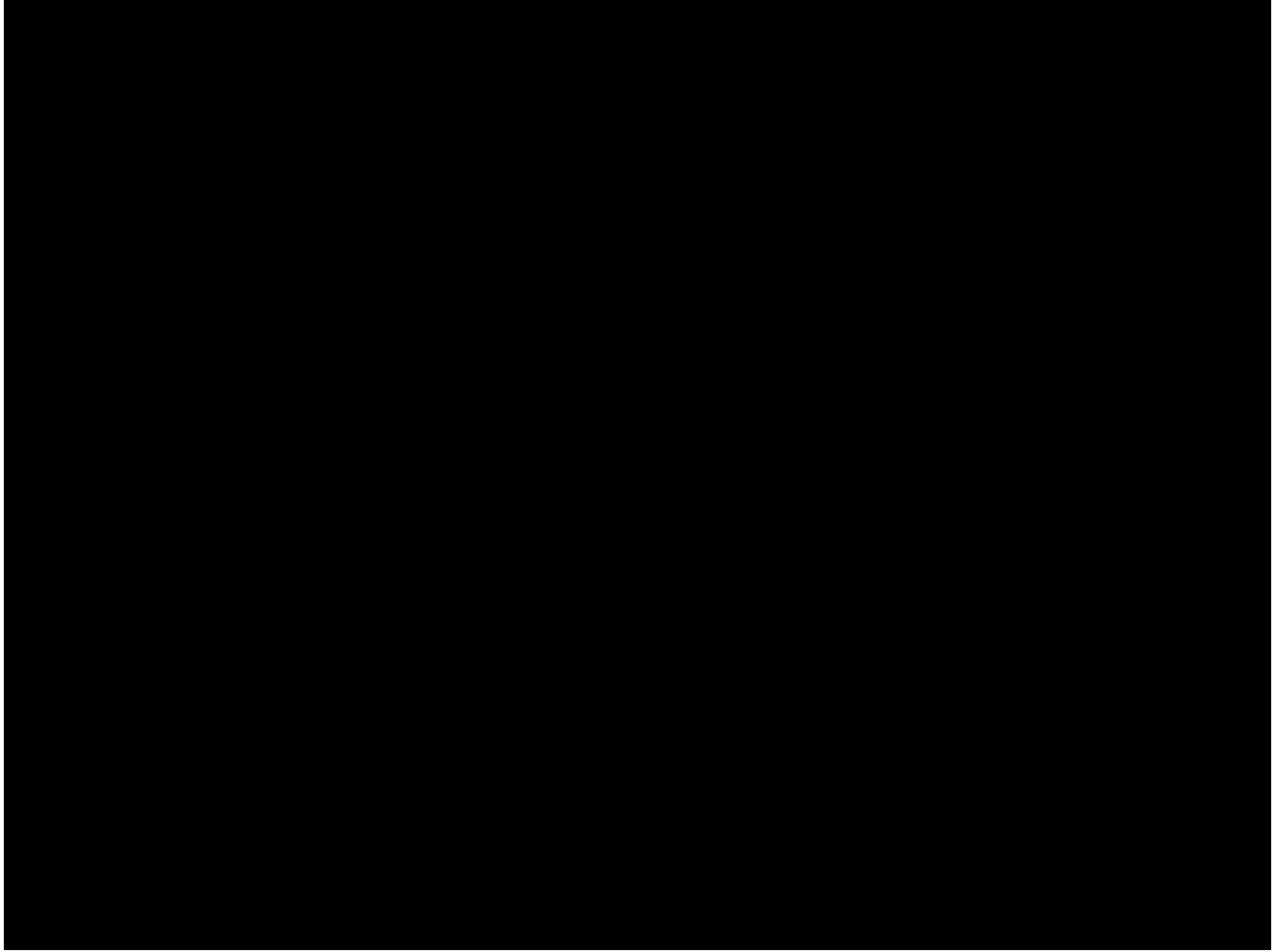


6 point "Glimpse path" (in green) while trying to classify image



6 Foveal Glimpses seen by the network

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



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

# 2 Soft Attention

# Soft Attention

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 a to determine a distribution Pr(g|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. **a** as long as Pr(**g**|**a**) is:

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

# Attention weights

We don't really need a *distribution* at all: any set of *weights* $\boldsymbol{w_i}$* can be used to define an attention readout $\boldsymbol{v}$ from some values $\mathbf{v_i}$:

$$v = \sum_i w_i v_i$$

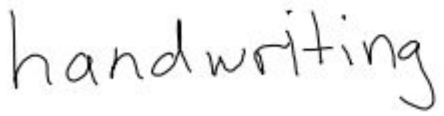
Look familiar? Can think of attention as defining data-dependent *dynamic weights* (c.f. *fast weights*)

Convolution

Attention

*but it's nice if $\sum_i w_i = 1;\ 0 \leq w_i \leq 1 \forall i$

# Handwriting Synthesis with RNNs

handwriting $\longrightarrow$ 


- An RNN takes a text sequence as input, produces a sequence of pen trajectories as output
- **Problem**: the **alignment** between the text and the writing is unknown
- **Solution**: before predicting each point on the trajectory, the network decides **where to look** in the text sequence

# Location Attention

Window vector (input to net)

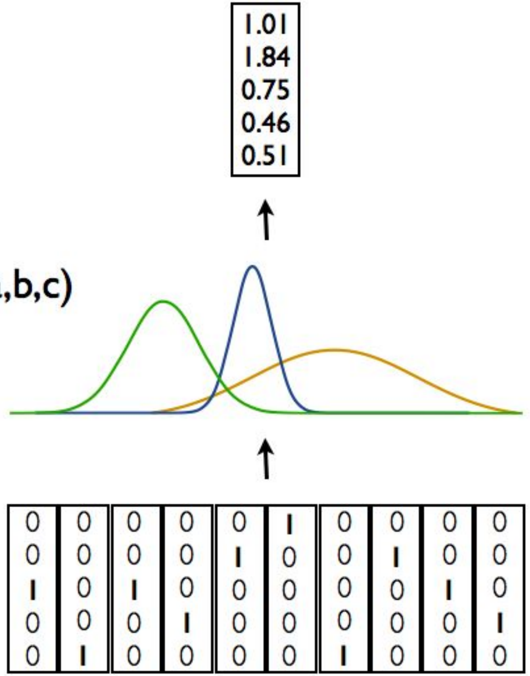
$$v^{t+1} = \sum_{i=1}^{S} w_i^t s_i$$

Window weights (net outputs for a,b,c)

$$w_i^t = \sum_{k=1}^{K} a_k^t \exp\left(-b_k^t [c_k^t - i]^2\right)$$

Input vectors (one-hot)

$(s_1, \ldots, s_S)$




Gaussian 'window' over text sequence index **(soft reading)**

*Generating Sequences with Recurrent Neural Networks*, Graves (2013)

# Writing with Attention

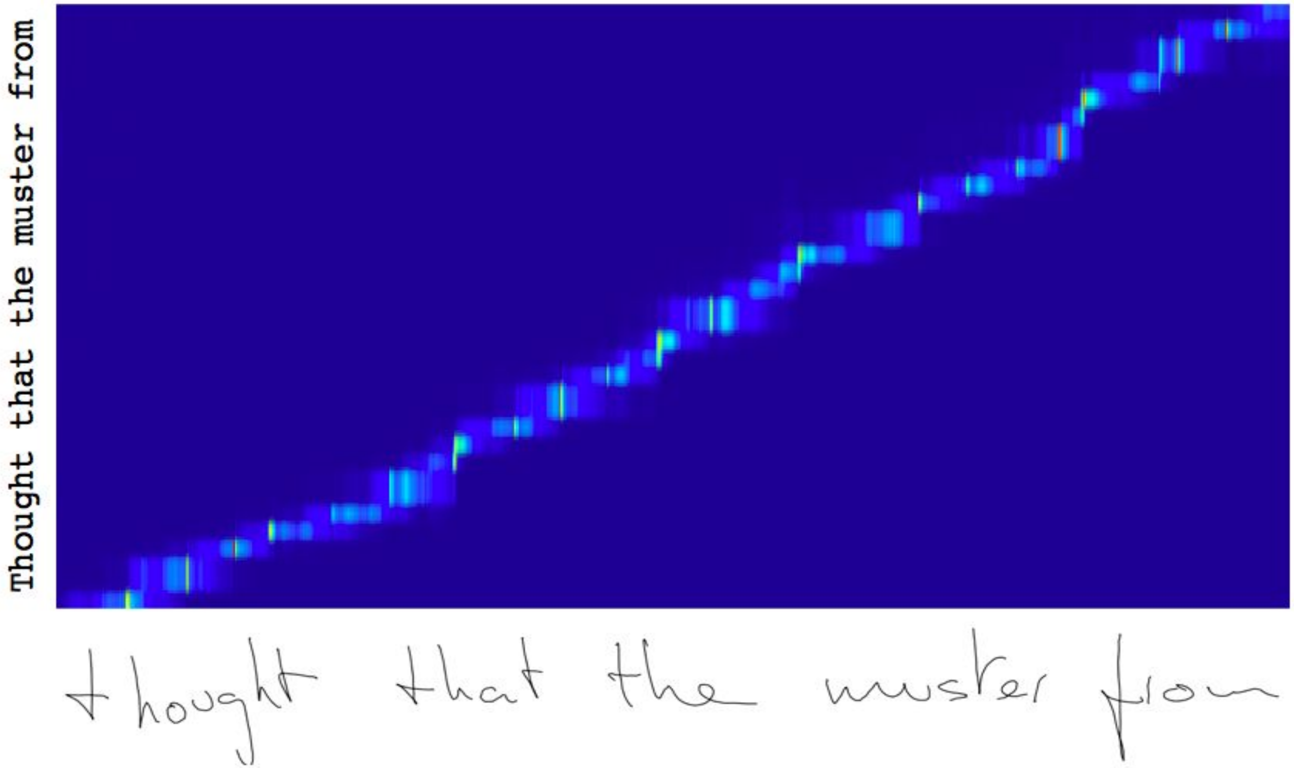
these sequences were generated by

picking samples at every step

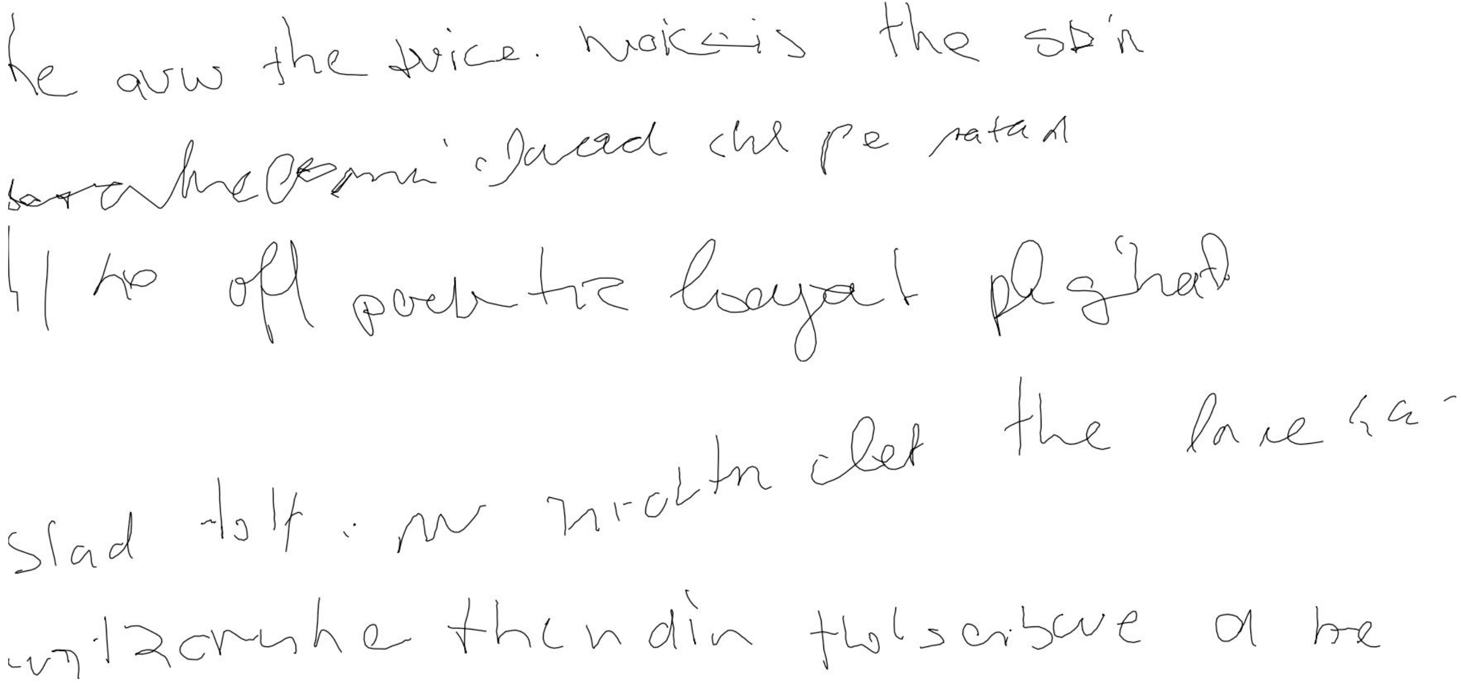
every line is a different style

yes, real people write this badly

# Alignment



# Unconditional Writing



# Associative Attention

Instead of attending by position, we can attend by **content: a key vector *k*** is compared to all $x_i$ in the data using some **similarity function** *S*. The similarities are typically normalised (softmax) and used to define $\mathbf{w}_i$

$$w_i = \frac{\exp S(k, x_i))}{\sum_j \exp S(k, x_j))}$$

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

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

# Keys and Values

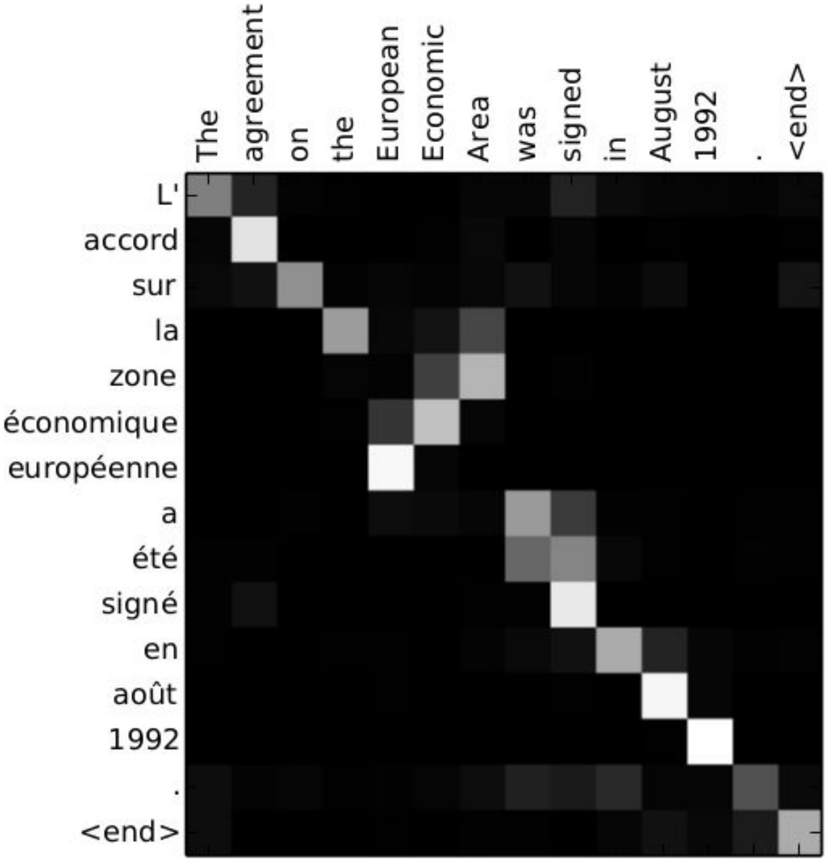
Given $\boldsymbol{w_i}$ we can sum over the data directly to get an attention readout $\boldsymbol{v}$

$$v = \sum_i w_i x_i$$

Or we can split the data into key, value pairs ($\boldsymbol{k_i}$, $\boldsymbol{v_i}$), use the keys to define the attention weights and the values to define the readout:

$$w_i = \frac{\exp S(k, k_i))}{\sum_j \exp S(k, k_j))} \qquad v = \sum_i w_i v_i$$

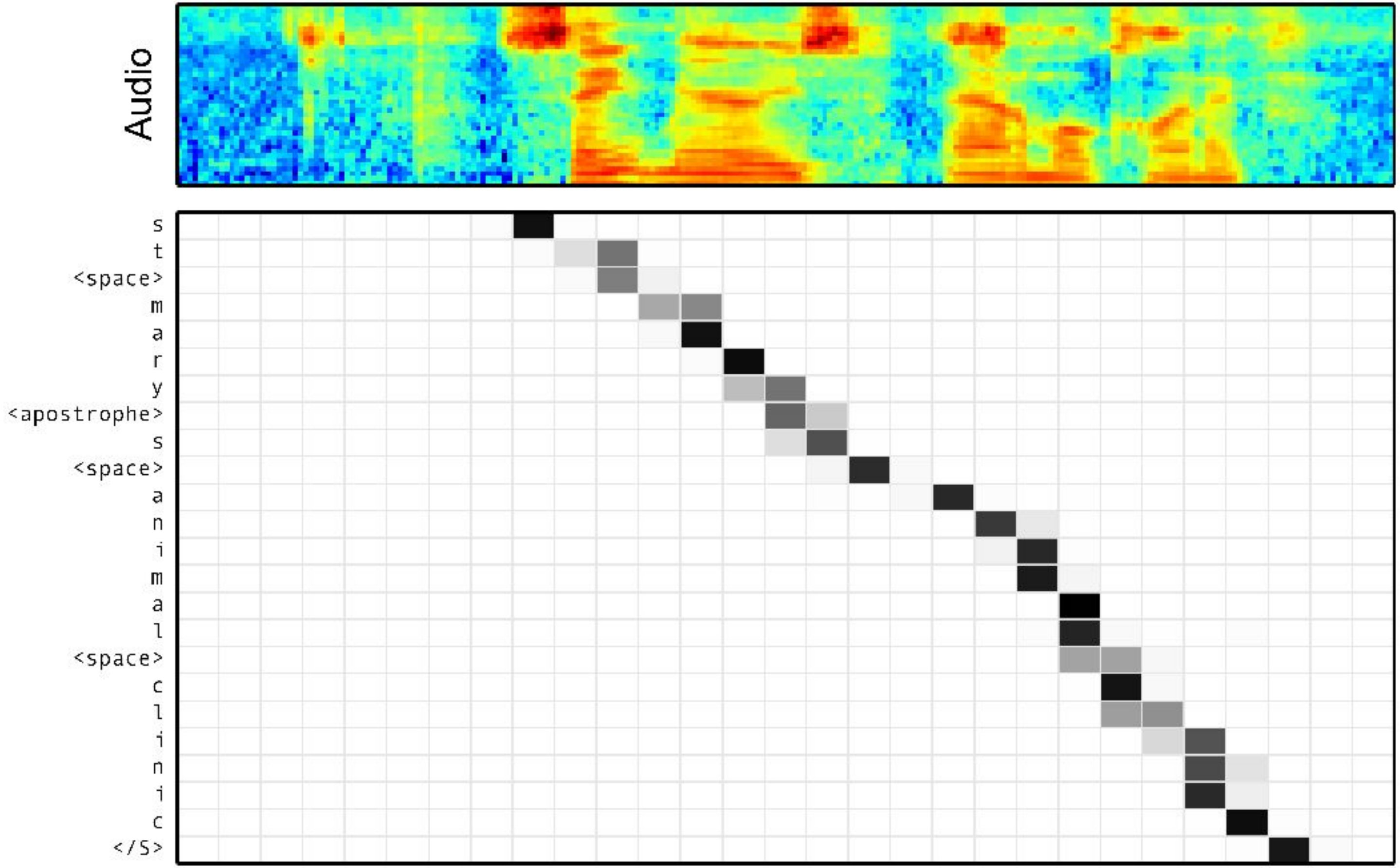
# Reordering in machine translation using associative attention




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

| | |
|---|---|
| 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 in all facets of his life ,and he leaves an inspiring legacy of natural tenacity and focused . . . | 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 , . . . |
| *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)




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

# Differentiable Visual Attention

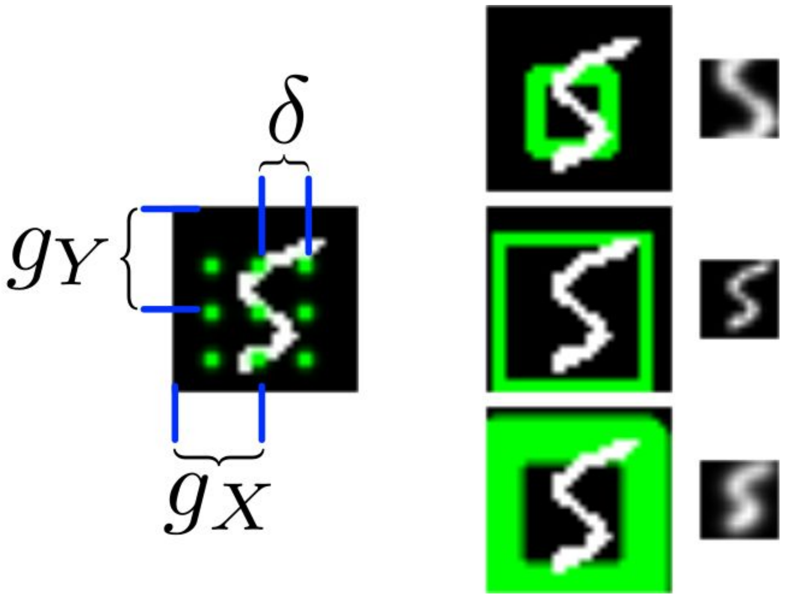
DRAW (Gregor et. al. 2015) uses a grid of Gaussian filters to **read** from input images and **draw** to a **canvas** image:

$\sigma^2$ — filter variance

$g_X, g_Y$ — grid centre

$\delta$ — grid stride

$\gamma$ — intensity



# 3 Introspective Attention

# 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** (Memory = attention through time)

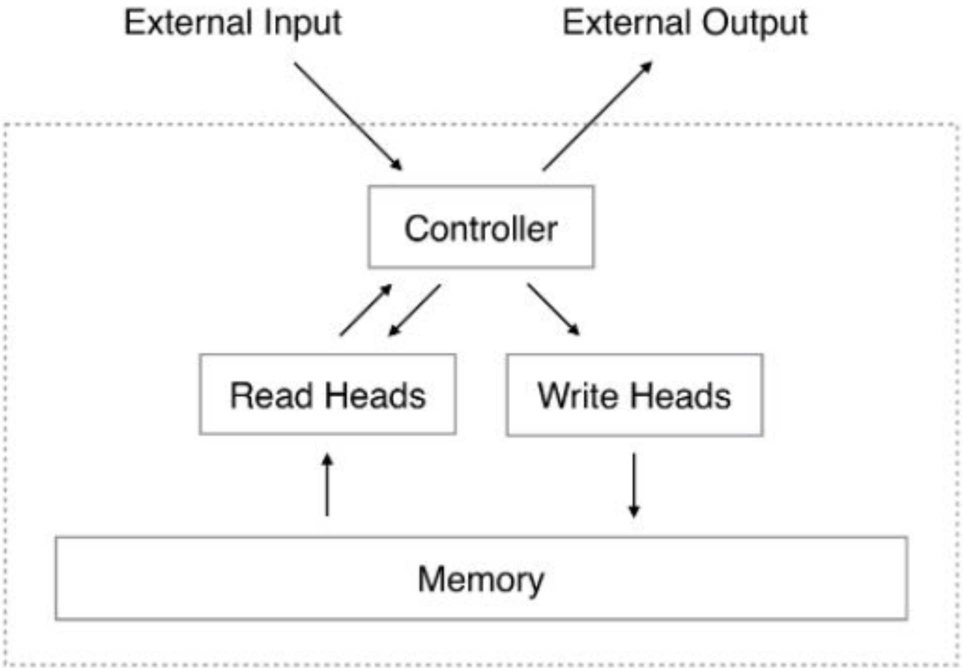
With internal information we can do selective ***writing*** as well as ***reading***, allowing the network to **iteratively modify** its state

# 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 **Memory** is a real-valued **matrix**




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

# Selective Attention

- Want to focus on the parts of memory the network will read and write to: need an ***introspective*** attention model
- We use the controller outputs to parameterise a distribution (weighting) over the rows (locations) in the memory matrix
- The weighting is defined two main attention mechanisms: one based on content and one based on location

# Addressing by content

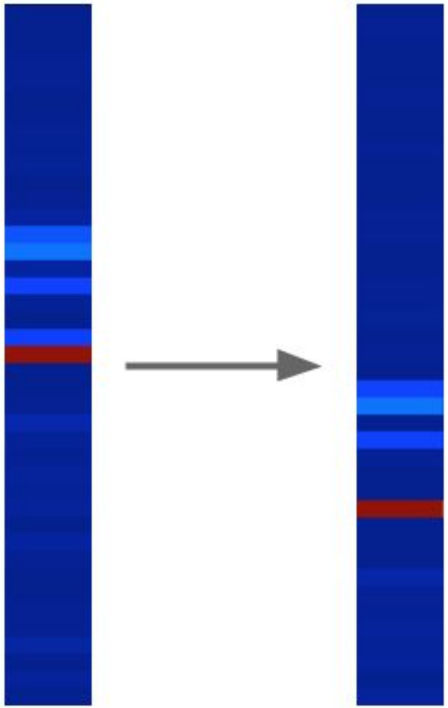
A key vector ***k*** is emitted by the controller and compared to the content of each memory location **M**[i] using a similarity measure $S(\cdot,\cdot)$ (e.g. **cosine distance**) then normalised with a **softmax**. A 'sharpness' $\beta$ is used to narrow the focus. Finds the memories **'closest'**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

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

# Addressing by Location

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

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



# 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

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

# Reading and Writing

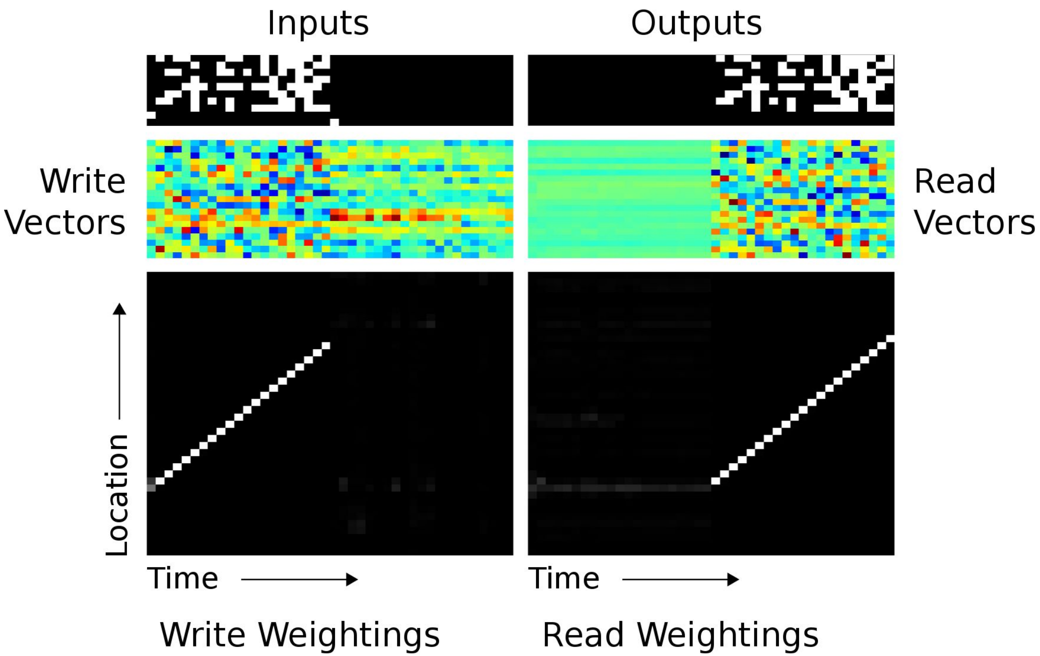
Once the weightings are defined, each read head returns a read vector **r** as input to the controller at the next timestep

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

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

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

# The NTM Copy Algorithm

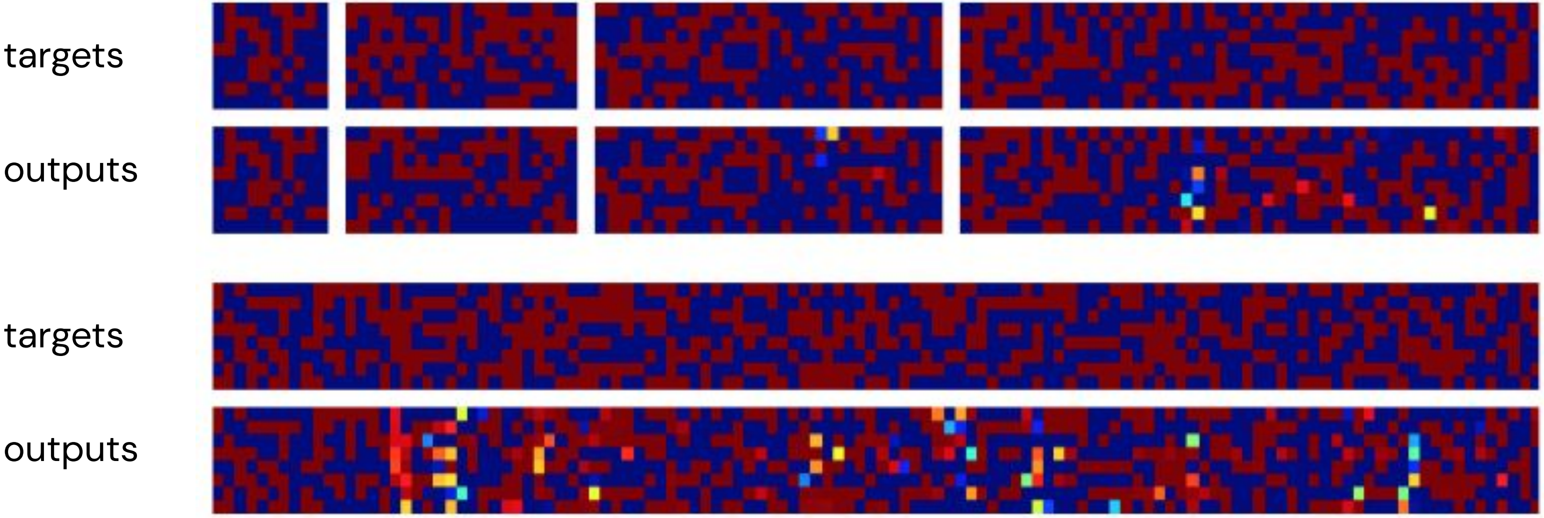



**NTM++**

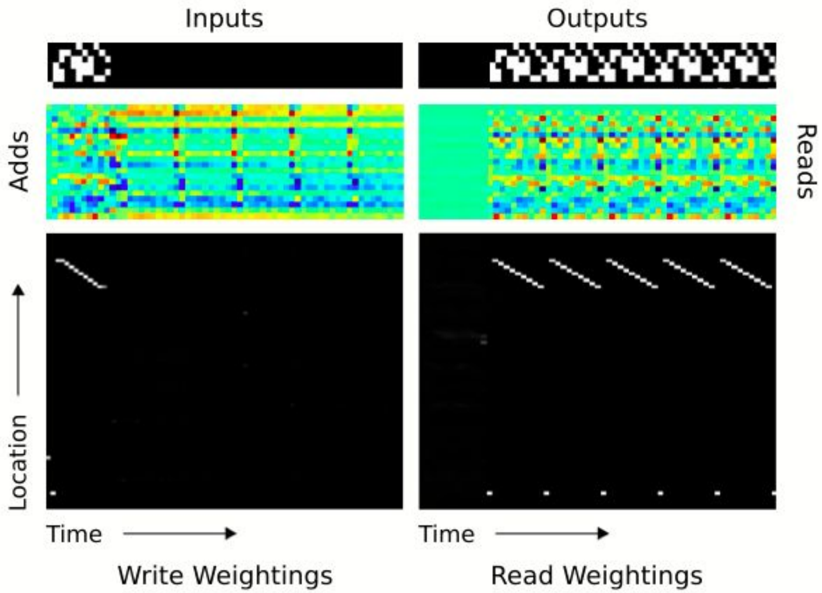
**initialize:** move head to start location
**while** input delimiter not seen **do**
  receive input vector
  write input to head location
  increment head location by 1
**end while**
return head to start location
**while** true **do**
  read output vector from head location
  emit output
  increment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

**pseudocode**

# Copy Generalisation: length 10 to 120

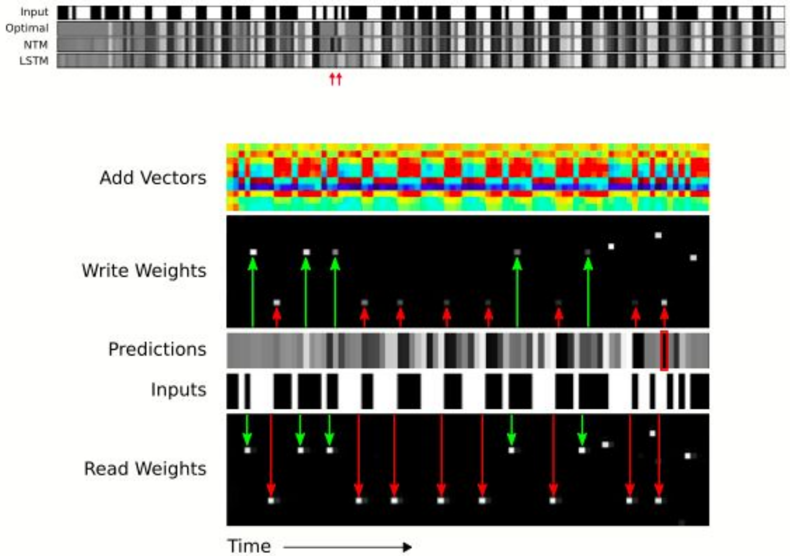


# Copy *N* Times



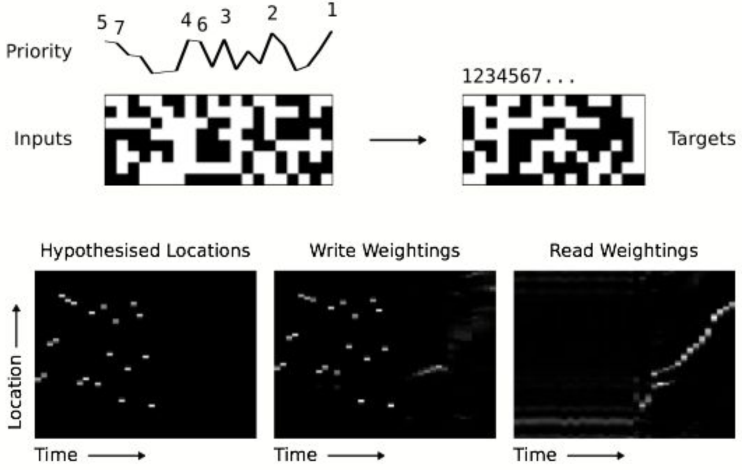

NTM learns its first **for-loop**, using **content** to jump, **iteration** to step, and a **variable** to count to *N*.

# N-Gram Inference



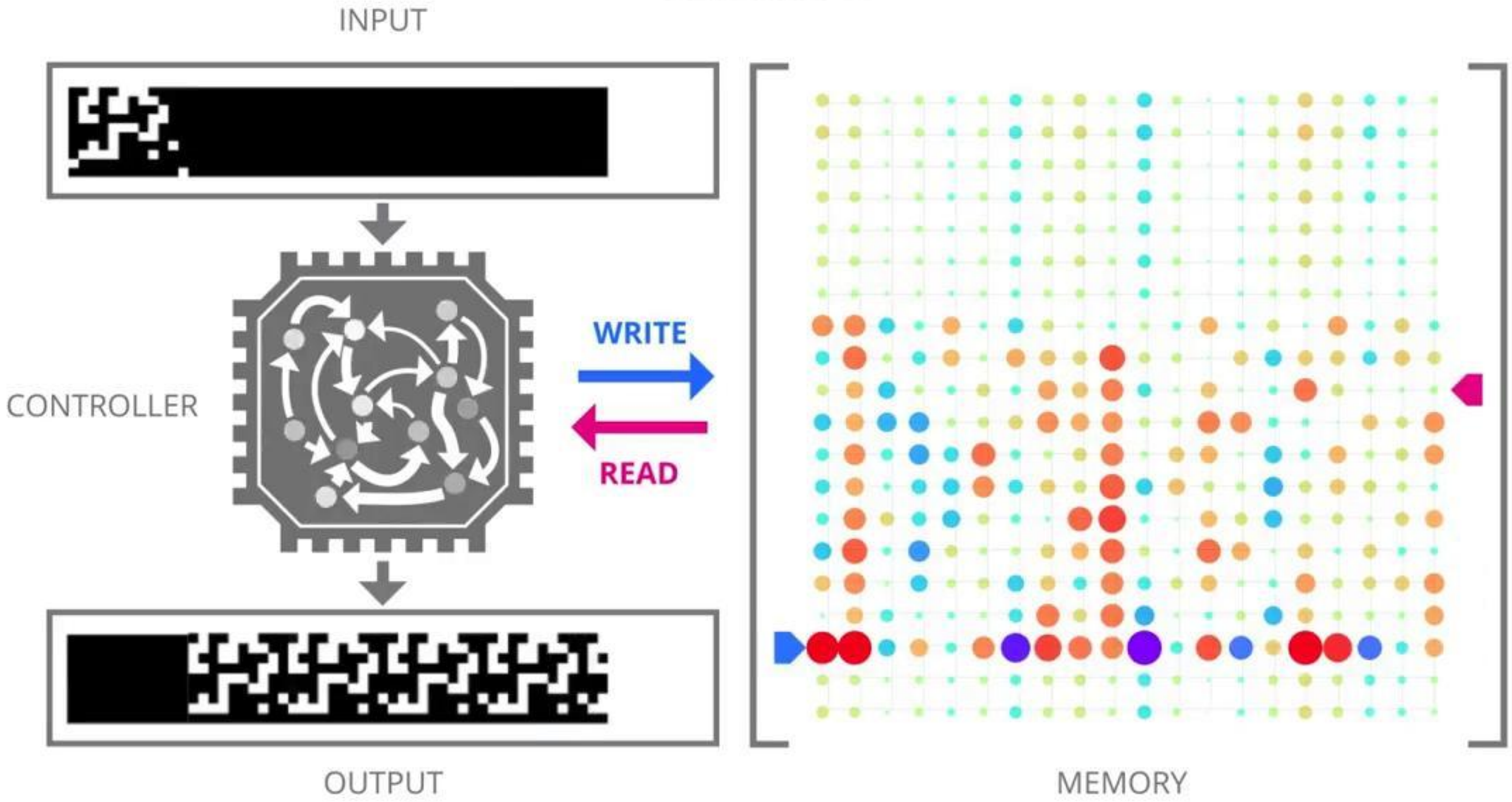

Specific memory locations store **variables** that **count** the **occurrences** of particular N-Grams

# Priority Sort




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

# TESTING



# Differentiable Neural Computers

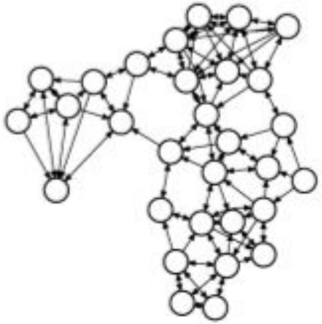
**DNC** is a successor architecture to Neural Turing Machines with new attention mechanisms for memory access

*Hybrid Computing Using a Neural network with Dynamic External Memory*, Graves et. al. (2016)

# Graph Experiments

## Training Data

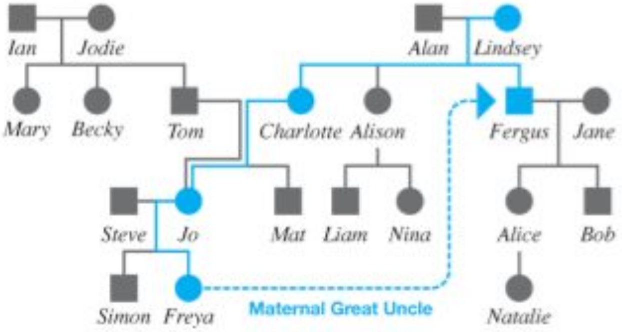
a. Random Graph



## Test Examples

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, PicadillyCircus, Bakerloo)
(OxfordCircus, NottingHillGate, Central)
(OxfordCircus, Euston, Victoria)

*- 84 edges in total*

**Traversal Question:**

(OxfordCircus, _, Central), (_, _, Circle)
(_, _, Circle), (_, _, Circle),
(_, _, Bakerloo), (_, _, Victoria),
(_, _, Victoria), (_, _, Circle),
(_, _, Bakerloo), (_, _, Jubilee)

**Answer:**

(OxfordCircus, NottingHillGate, Central)
(NottingHillGate, Paddington, Circle)
...
(Embankment, Waterloo, Bakerloo)
(Waterloo, GreenPark, Jubilee)

**Shortest Path Question:**

(Moorgate, PicadillyCircus, _)

**Answer:**

(Moorgate, Bank, Northern)
(Bank, Holborn, Central)
(Holborn, LeicesterSq, Picadilly)
(LeicesterSq, PicadillyCircus, Picadilly)

**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)
(Jane, Alice, Daughter1)
(Mat, Charlotte, Mother)

*- 54 edges in total*

**Inference Question:**

(Freya, _, MaternalGreatUncle)

**Answer:**

(Freya, Fergus, MaternalGreatUncle)

# bAbI Tasks

Set of 20 question-answering tasks on synthetic 'stories'

*"One Supporting Fact"*

**Story**

Mary went to the hallway.

John went to the kitchen.

**Q:** Where is Mary?

**A**: hallway

*20 different tasks*

●●●

"Counting"

**Story**

Abe got the football. Abe dropped the football. Abe got the milk .

**Q:** How many objects is Abe holding?

**A:** two.

*Towards AI Complete Question Answering: A Set of Prerequisite Toy Tasks.* Weston et. al. (2015)

# bAbI Results

| | bAbI Best Results | | | | | | |
|---|---|---|---|---|---|---|---|
| Task | LSTM (Joint) | NTM (Joint) | DNC1 (Joint) | DNC2 (Joint) | MemN2N (Joint) [21] | MemN2N (Single) [21] | DMN (Single) [20] |
| 1: 1 supporting fact | 24.5 | 31.5 | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| 2: 2 supporting facts | 53.2 | 54.5 | 1.3 | 0.4 | 1.0 | **0.3** | 1.8 |
| 3: 3 supporting facts | 48.3 | 43.9 | 2.4 | **1.8** | 6.8 | 2.1 | 4.8 |
| 4: 2 argument rels. | 0.4 | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| 5: 3 argument rels. | 3.5 | 0.8 | **0.5** | 0.8 | 6.1 | 0.8 | 0.7 |
| 6: yes/no questions | 11.5 | 17.1 | **0.0** | **0.0** | 0.1 | 0.1 | **0.0** |
| 7: counting | 15.0 | 17.8 | **0.2** | 0.6 | 6.6 | 2.0 | 3.1 |
| 8: lists/sets | 16.5 | 13.8 | **0.1** | 0.3 | 2.7 | 0.9 | 3.5 |
| 9: simple negation | 10.5 | 16.4 | **0.0** | 0.2 | **0.0** | 0.3 | **0.0** |
| 10: indefinite knowl. | 22.9 | 16.6 | 0.2 | 0.2 | 0.5 | **0.0** | **0.0** |
| 11: basic coreference | 6.1 | 15.2 | **0.0** | **0.0** | **0.0** | 0.1 | 0.1 |
| 12: conjunction | 3.8 | 8.9 | 0.1 | **0.0** | 0.1 | **0.0** | **0.0** |
| 13: compound coref. | 0.5 | 7.4 | **0.0** | 0.1 | **0.0** | **0.0** | 0.2 |
| 14: time reasoning | 55.3 | 24.2 | 0.3 | 0.4 | **0.0** | 0.1 | **0.0** |
| 15: basic deduction | 44.7 | 47.0 | **0.0** | **0.0** | 0.2 | **0.0** | **0.0** |
| 16: basic induction | 52.6 | 53.6 | 52.4 | 55.1 | **0.2** | 51.8 | 0.6 |
| 17: positional reas. | 39.2 | 25.5 | 24.1 | **12.0** | 41.8 | 18.6 | 40.4 |
| 18: size reasoning | 4.8 | 2.2 | 4.0 | **0.8** | 8.0 | 5.3 | 4.7 |
| 19: path finding | 89.5 | 4.3 | **0.1** | 3.9 | 75.7 | 2.3 | 65.5 |
| 20: agent motiv. | 1.3 | 1.5 | **0.0** | **0.0** | **0.0** | **0.0** | **0.0** |
| Mean Err. (%) | 25.2 | 20.1 | 4.3 | **3.8** | 7.5 | 4.2 | 6.4 |
| Failed (err. > 5%) | 15 | 16 | **2** | **2** | 6 | 3 | **2** |

# 4 Further Topics

# Self-Attention

**Transformer networks** take attention to its logical extreme: get rid of everything else (recurrent state, convolutions, external memory) and **just use attention** to repeatedly transform a complete sequence

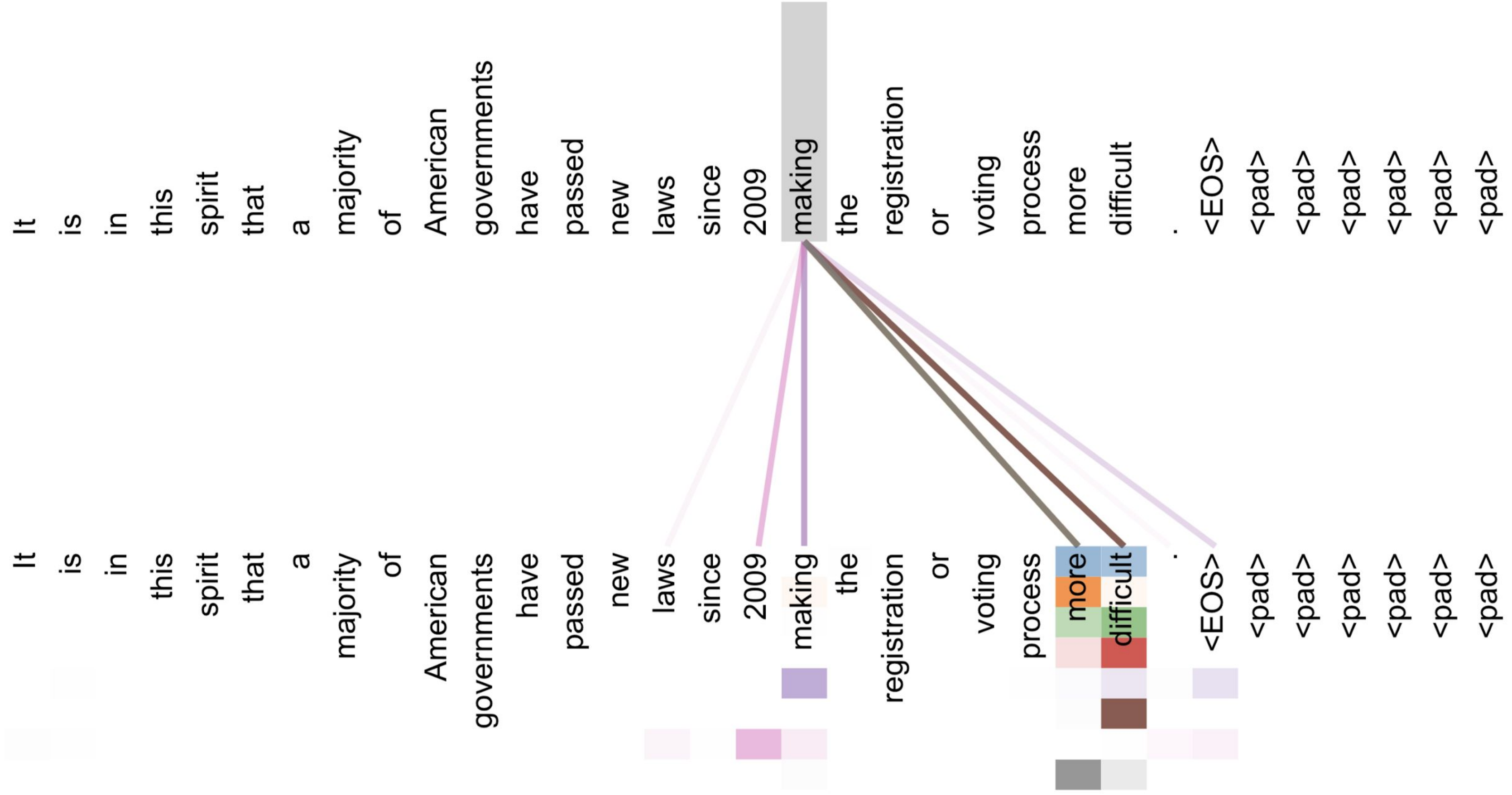
Instead of a **controller** emitting a query, every vector in the input sequence is compared with every other: **anarchist attention?**

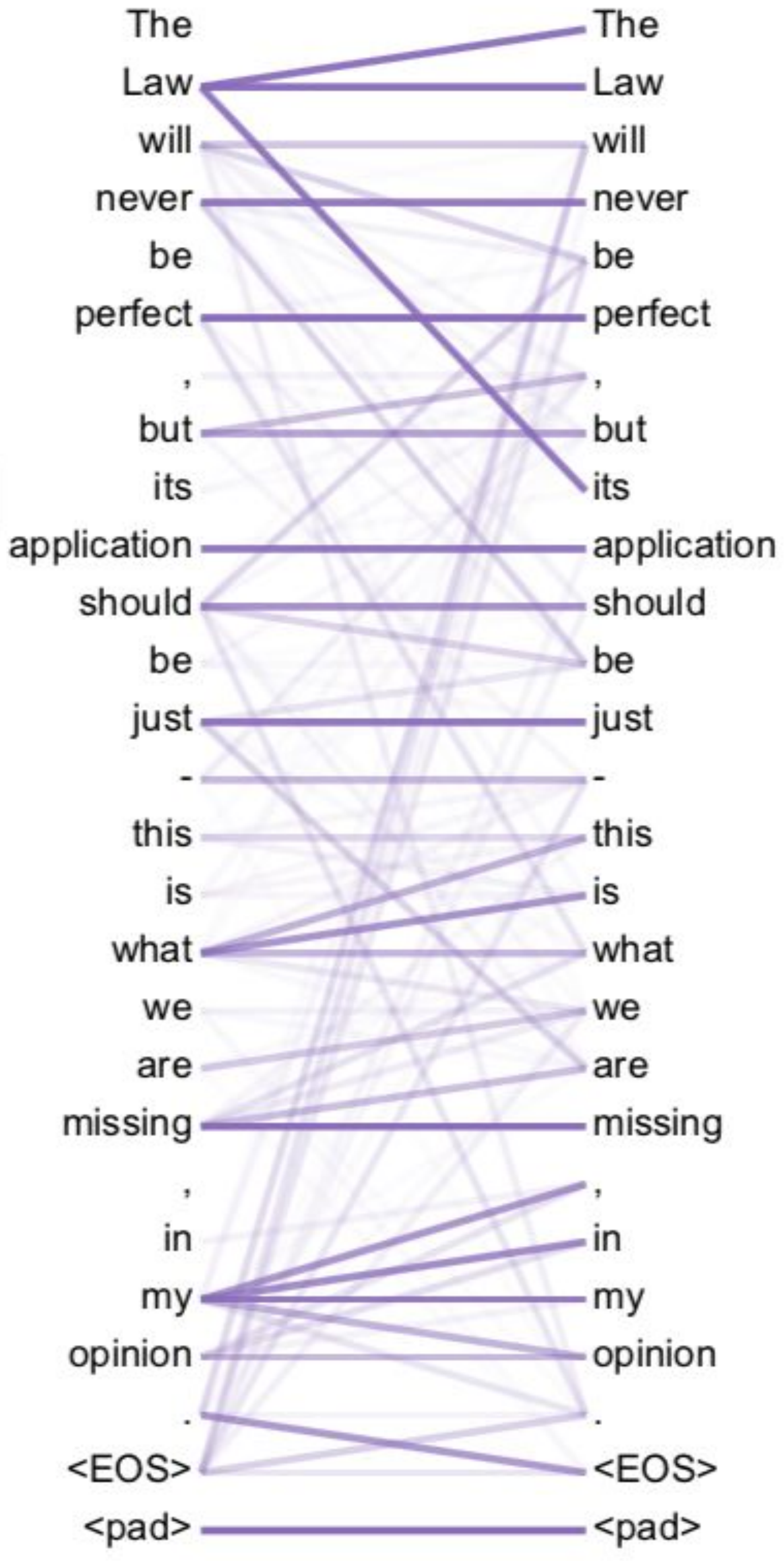
$$\mathrm{Attention}(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

Like NTM / DNC, multiple **heads** are used for **multimodal** attention

*Attention is All You Need*, Vaswani et. al. (2017)

*The Annotated Transformer*: http://nlp.seas.harvard.edu/2018/04/03/attention.html


It
is
in
this
spirit
that
a
majority
of
American
governments
have
passed
new
laws
since
2009
making
the
registration
or
voting
process
more
difficult
.
<EOS>
<pad>
<pad>
<pad>
<pad>
<pad>
<pad>


The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>
The Law will never be perfect , but its application should be just - this is what we are missing , in my opinion . <EOS> <pad>


The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>
The
Law
will
never
be
perfect
,
but
its
application
should
be
just
-
this
is
what
we
are
missing
,
in
my
opinion
.
<EOS>
<pad>

Table 2: The Transformer achieves better BLEU scores than previous state-of-the-art models on the English-to-German and English-to-French newstest2014 tests at a fraction of the training cost.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | $\mathbf{3.3 \cdot 10^{18}}$ | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

# Generating text with Transformers

Human Prompt

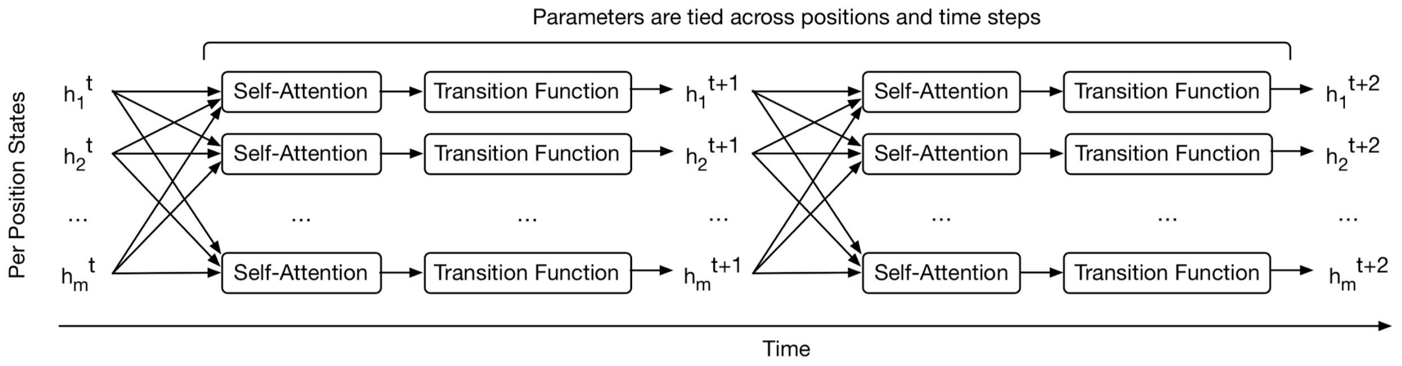
**In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.**

Generated Text (10 tries)

The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science.
Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved.
Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.
Pérez and the others then ventured further into the valley. "By the time we reached the top of one peak, the water looked blue, with some crystals on top," said Pérez.
Pérez and his friends were astonished to see the unicorn herd. These creatures could be seen from the air without having to move too much to see them – they were so close they could touch their horns.

*Language Models are Unsupervised Multi Task Learners*, Radford et. al. (2019)

# Universal Transformers

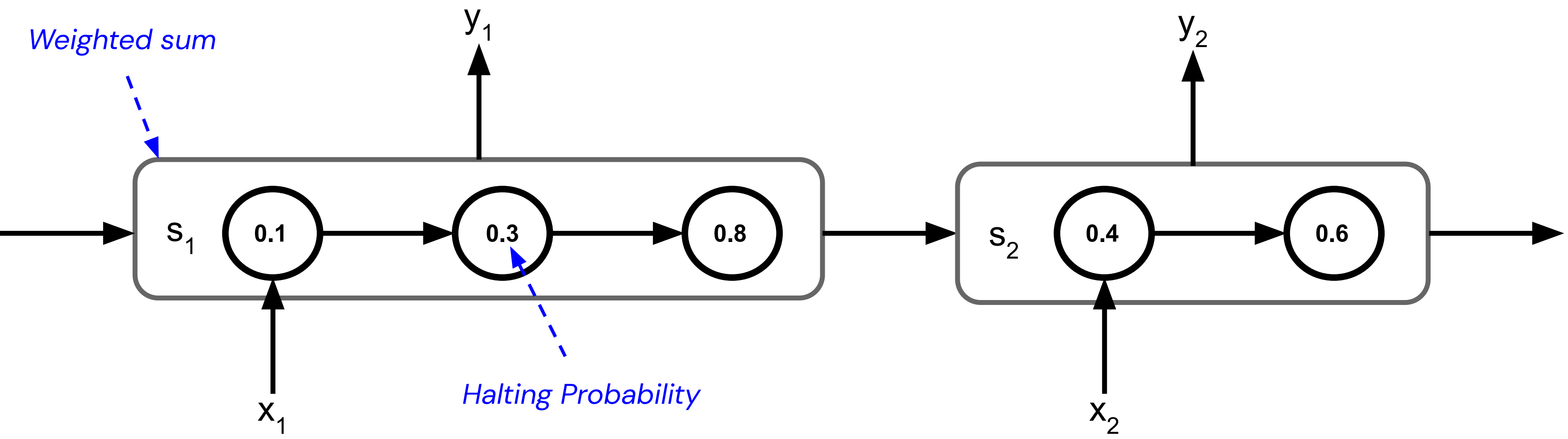



**Tying the weights** at each transform makes the system like an **RNN** in depth instead of time: variable runtime, recursive transforms

Recurrent state + parallel attention = best of both worlds? Strong results on MT, bAbI, LAMBADA...

*Universal Transformers*, Dehghani et. al. (2019)

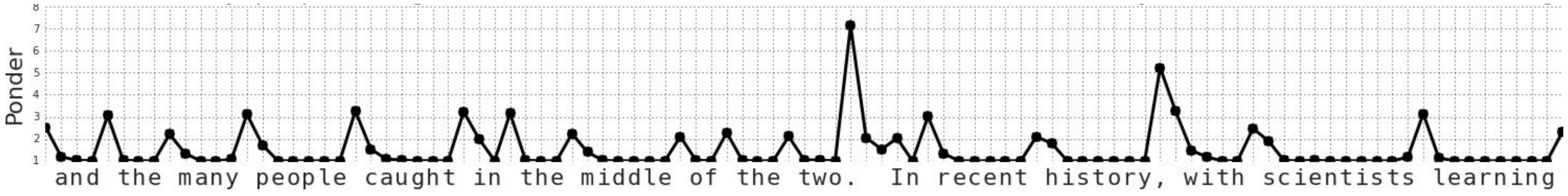
# Adaptive computation Time (ACT)

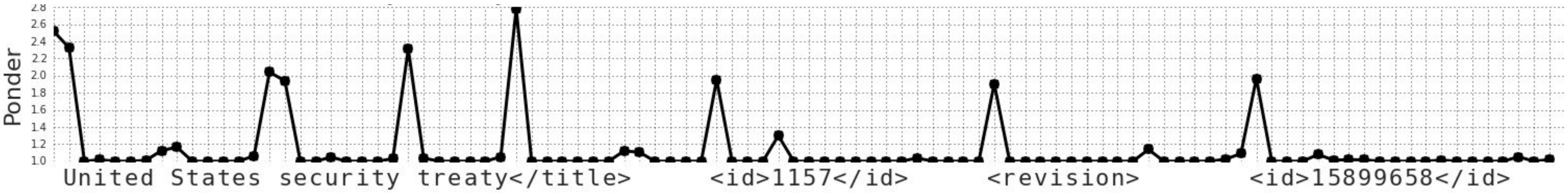



A **time penalty** acts to reduce the total number of **‘ponder’** steps

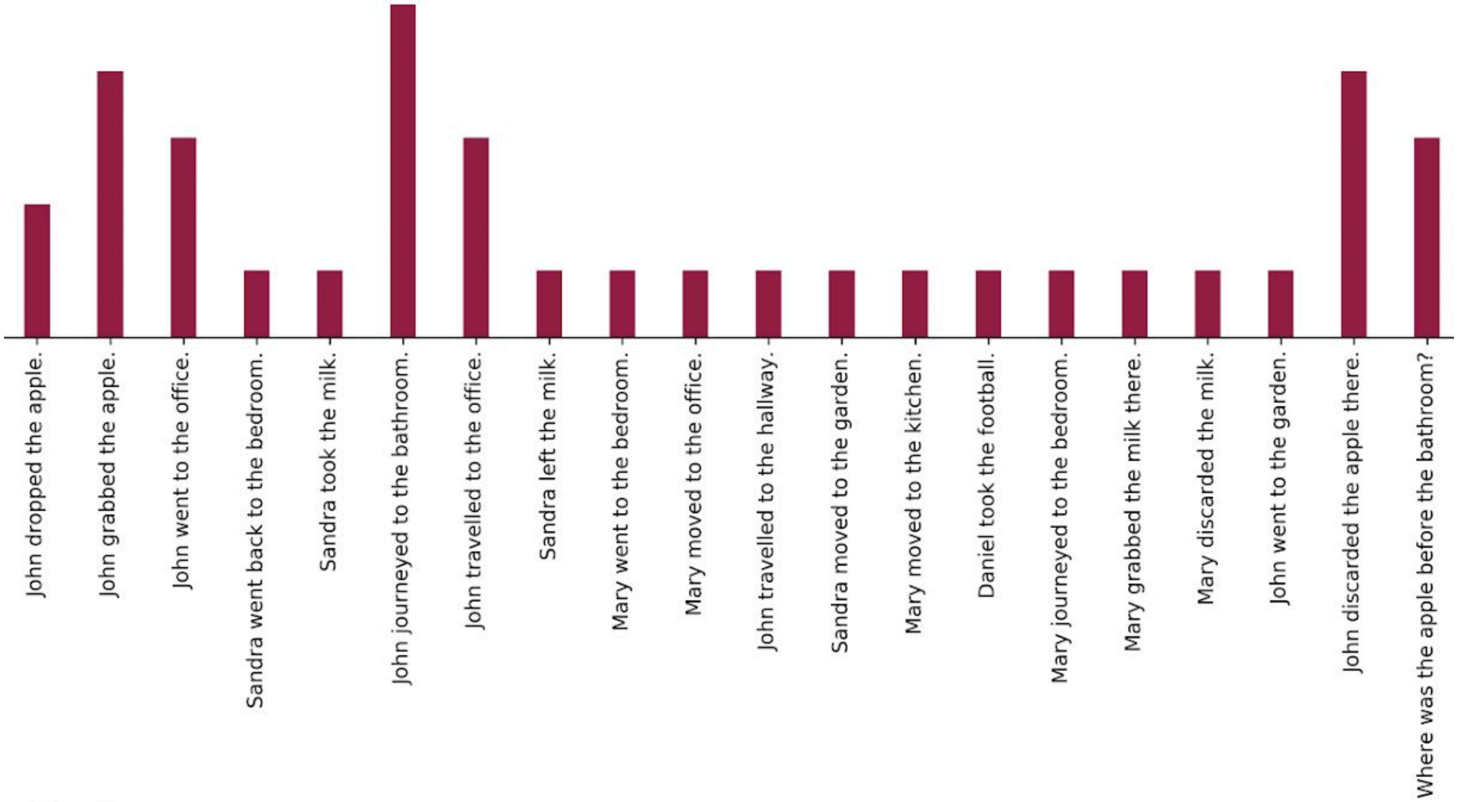
*Adaptive Computation Time With Recurrent Neural Networks, Graves (2016)*

# 'Pondering' with ACT: attention by concentration?



# ACT with Universal Transformers



# 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 **backdrop**
- We can use attention to attend to **memory** as well as directly to **data**
- Many types of attention mechanism (**content**, **spatial**, **visual**, **temporal…**) can be defined
- Can get great results in sequence learning *just using attention* (**transformers**)

# Thank you

# Questions