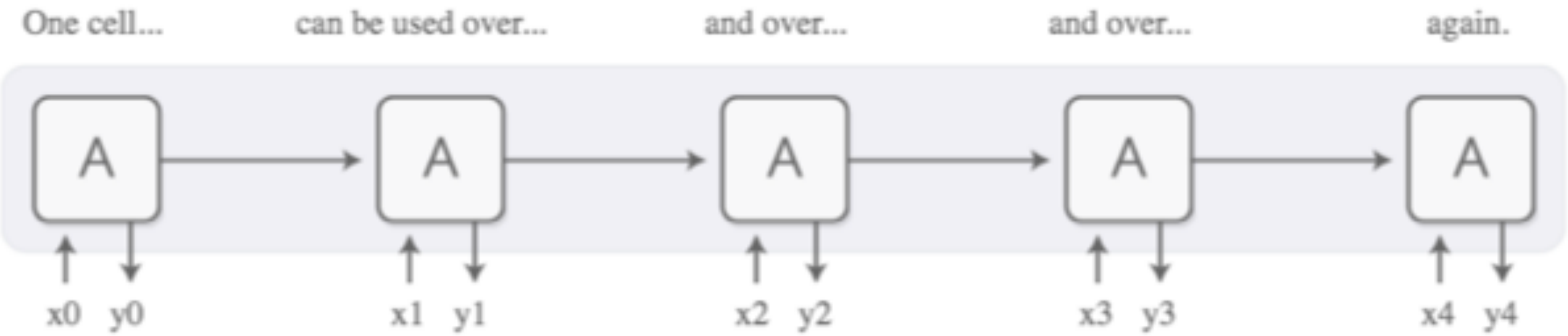
Attention and Augmented Neural Networks

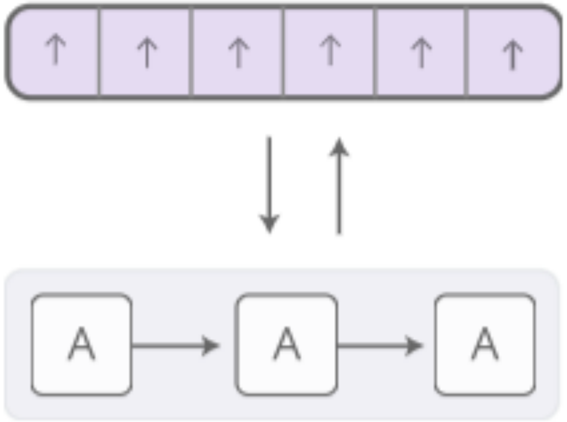
Korbinian Riedhammer

Based on https://distill.pub/2016/augmented-rnns/

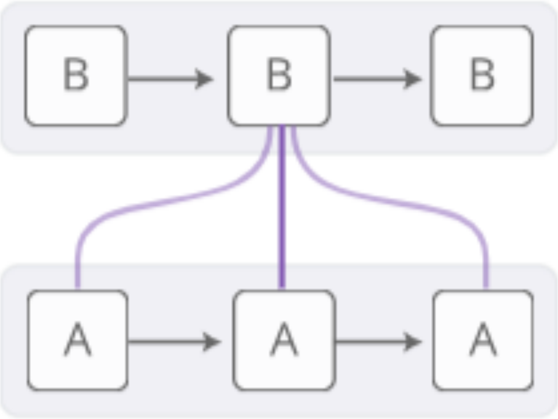
Recurrent Neural Networks



Variants

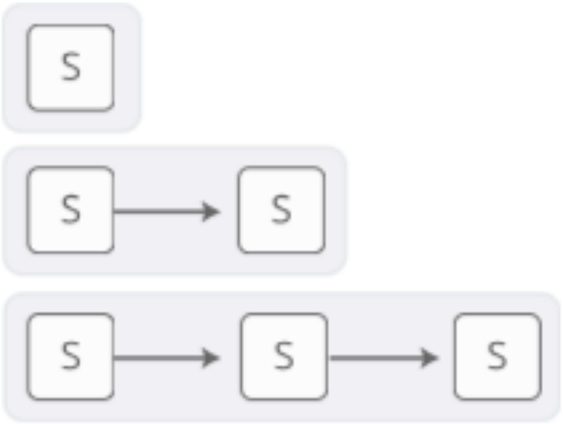


Neural Turing Machines have external memory that they can read and write to.

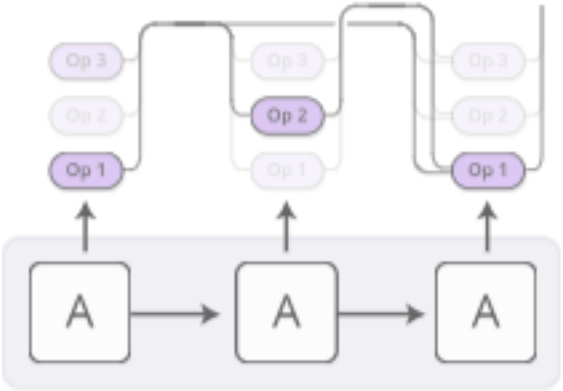


Attentional Interfaces allow RNNs to focus on parts of their input.

Variants

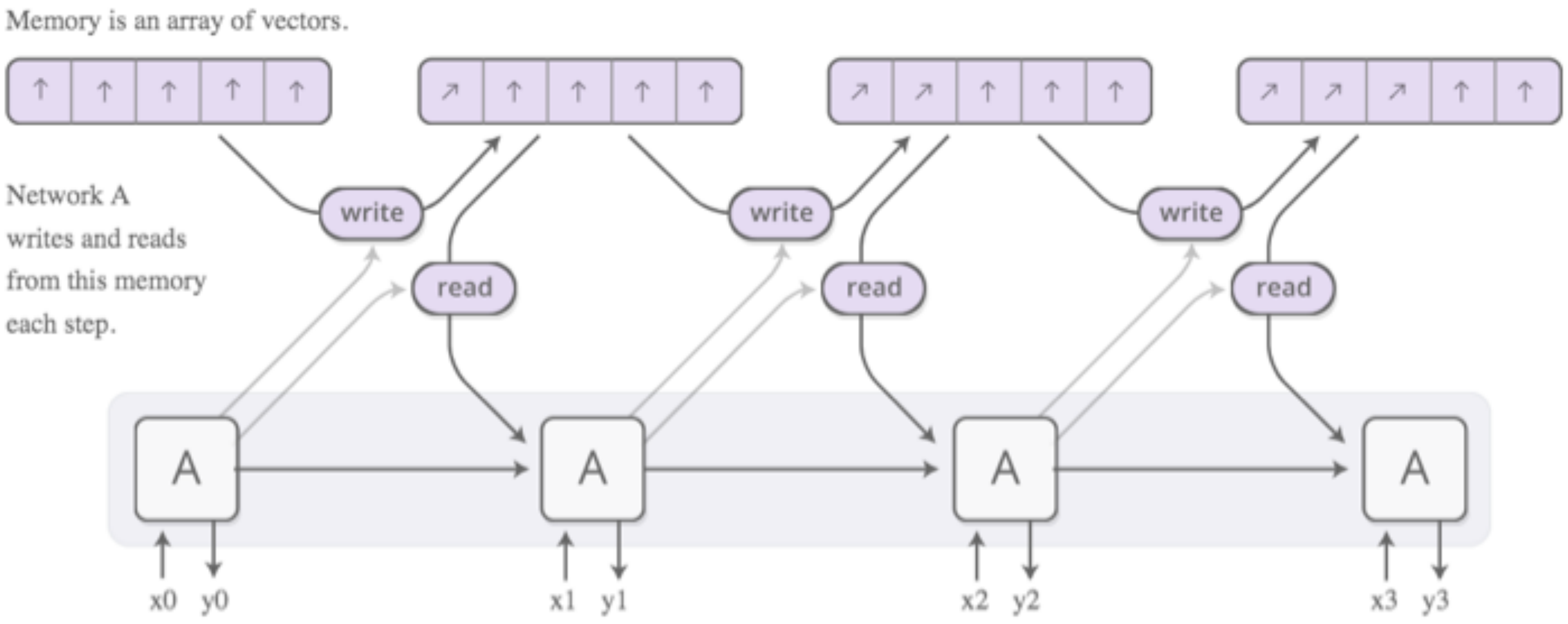


Adaptive Computation Time allows for varying amounts of computation per step.



Neural Programmers can call functions, building programs as they run.

Neural Turing Machines

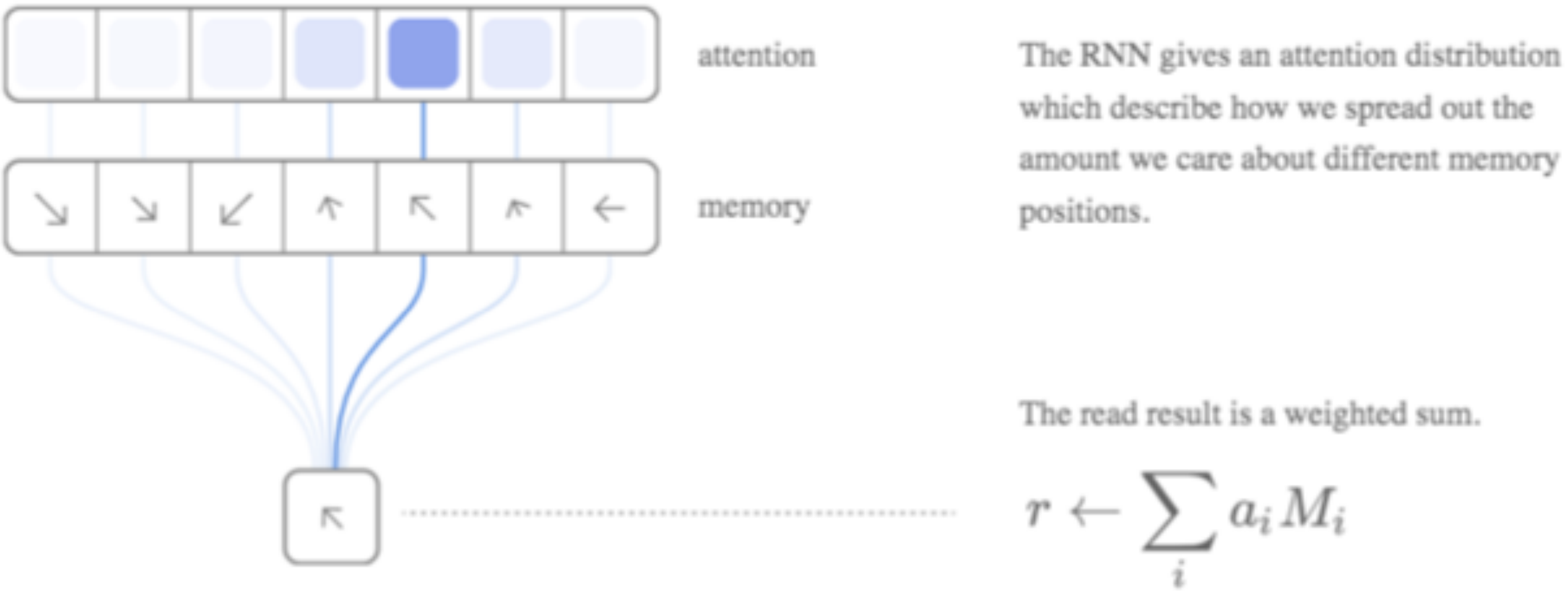

Memory is an array of vectors.
Network A writes and reads from this memory each step.
write
read
write
read
write
read
A
A
A
A
x0
y0
x1
y1
x2
y2
x3
y3

Neural Turing Machines

- Vectors are “natural language” of neural nets
- How to read/write from memory?
- How to differentiate?

In every step, read and write **everywhere!**

“Read” from Memory

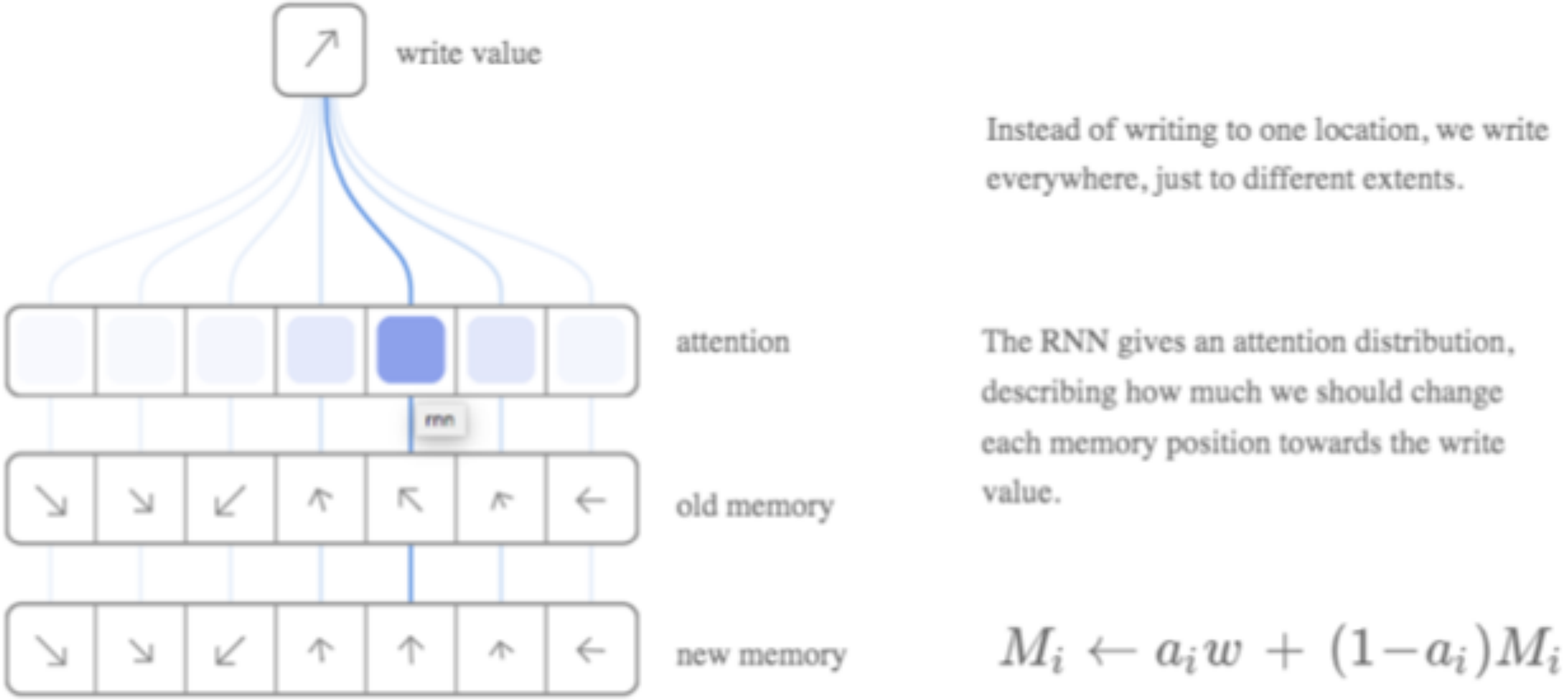


The RNN gives an attention distribution which describe how we spread out the amount we care about different memory positions.

The read result is a weighted sum.

$$r \leftarrow \sum_i a_i M_i$$

“Write” to Memory

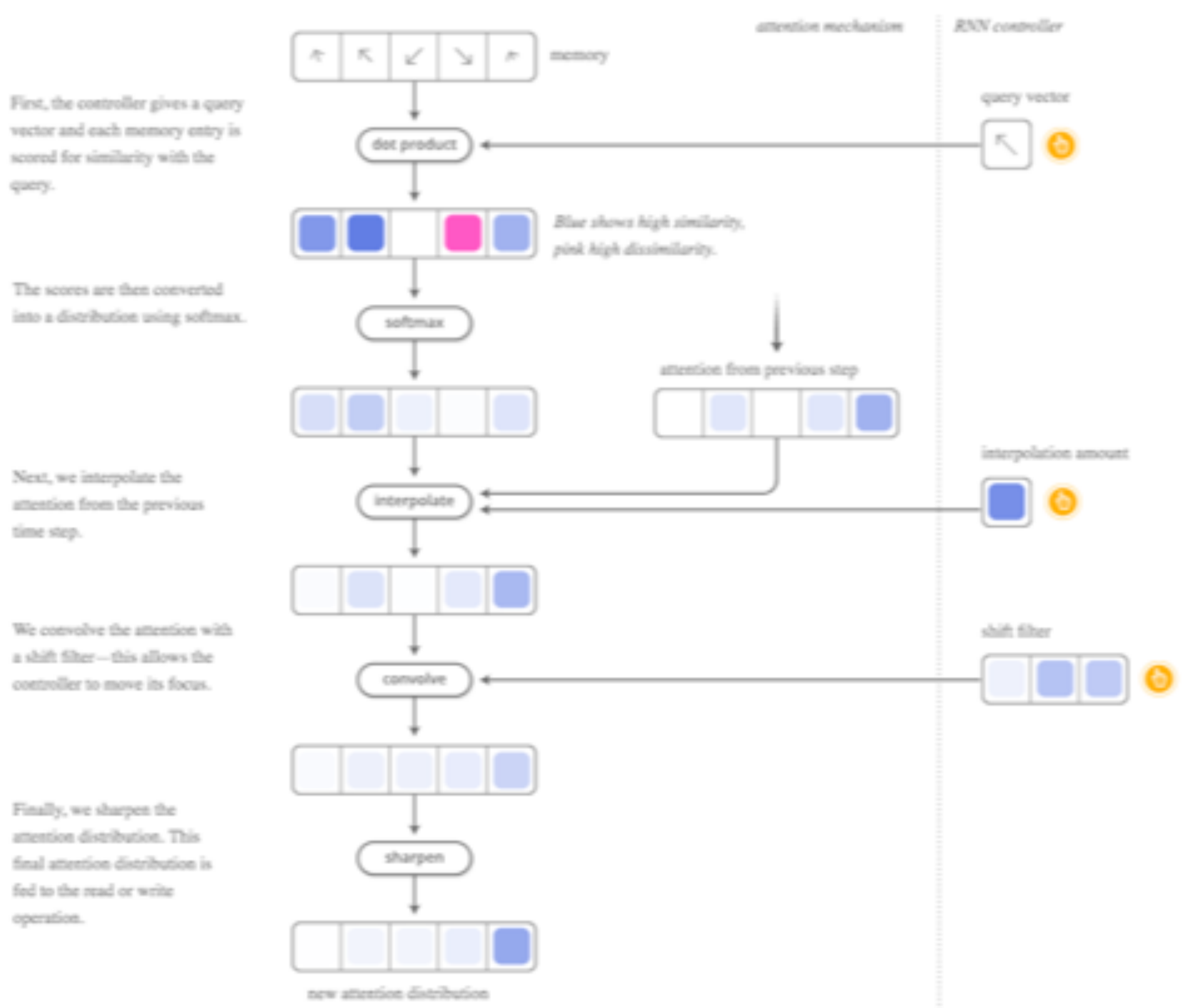


Instead of writing to one location, we write everywhere, just to different extents.

The RNN gives an attention distribution, describing how much we should change each memory position towards the write value.

$$M_i \leftarrow a_i w + (1 - a_i) M_i$$

Content based and Location based Attention



First, the controller gives a query vector and each memory entry is scored for similarity with the query.

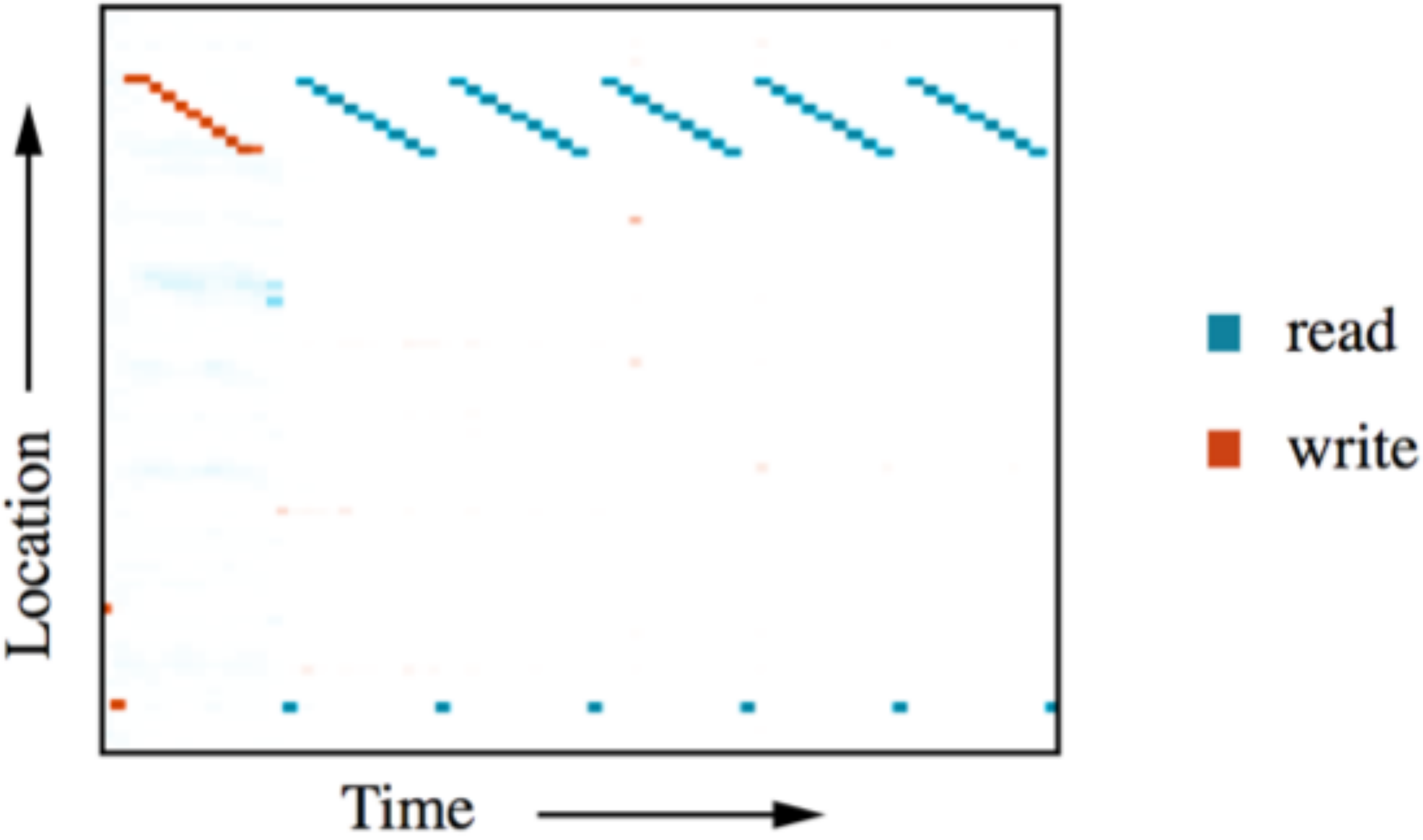
The scores are then converted into a distribution using softmax.

Next, we interpolate the attention from the previous time step.

We convolve the attention with a shift filter—this allows the controller to move its focus.

Finally, we sharpen the attention distribution. This final attention distribution is fed to the read or write operation.

Visualization

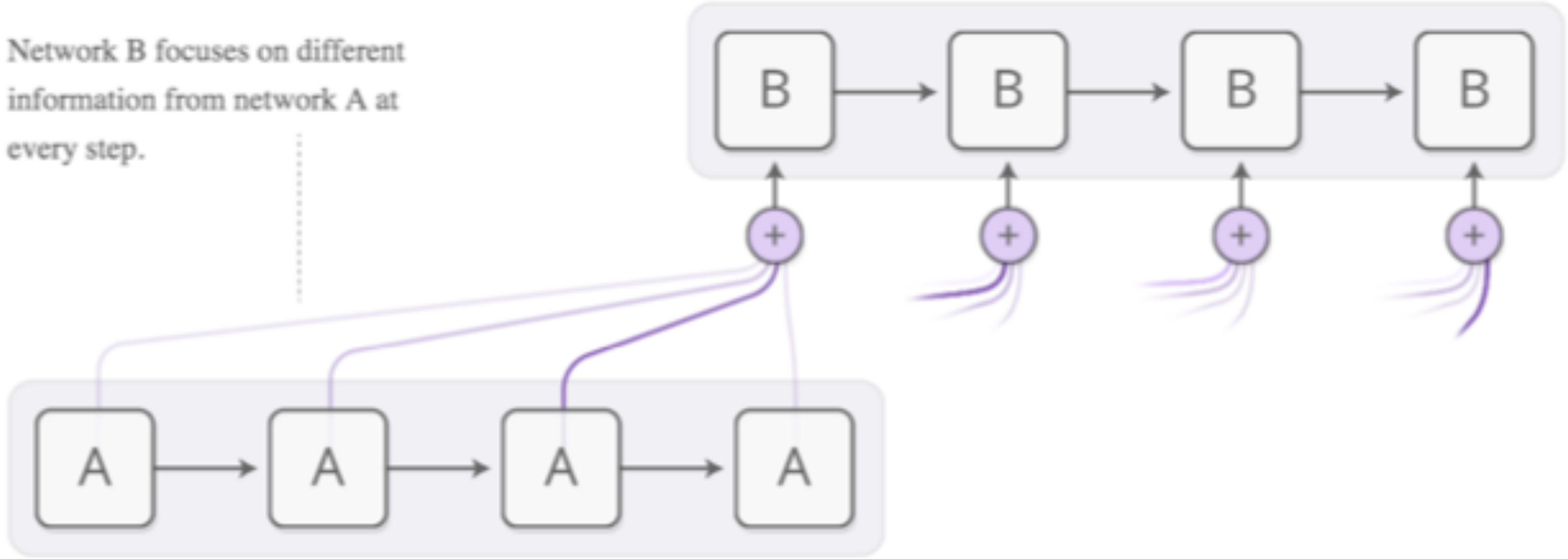


Attentional Interfaces

- Observation: In many complex human tasks (eg. translation, transcription, description, …), you pay attention to different aspects (in time and space)
- Model this “attention” in neural nets?
- How can we incorporate hidden state of previous time steps? …and be differentiable?

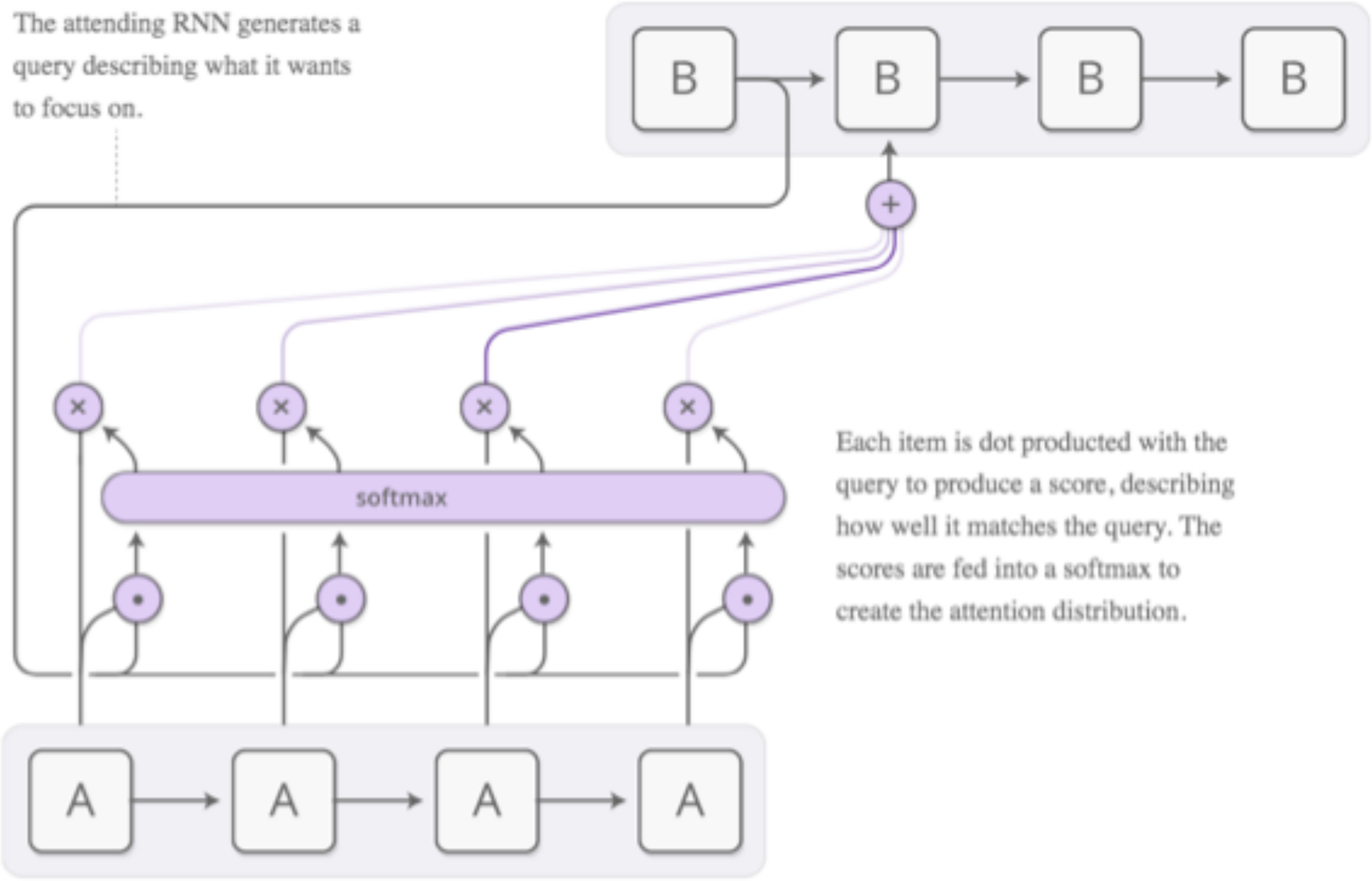
Similar to NTM, focus
everywhere
but with different amount

Attentional Interfaces



Network B focuses on different information from network A at every step.

Content-Based Attention



The attending RNN generates a query describing what it wants to focus on.

Each item is dot producted with the query to produce a score, describing how well it matches the query. The scores are fed into a softmax to create the attention distribution.

Example: direct dependency

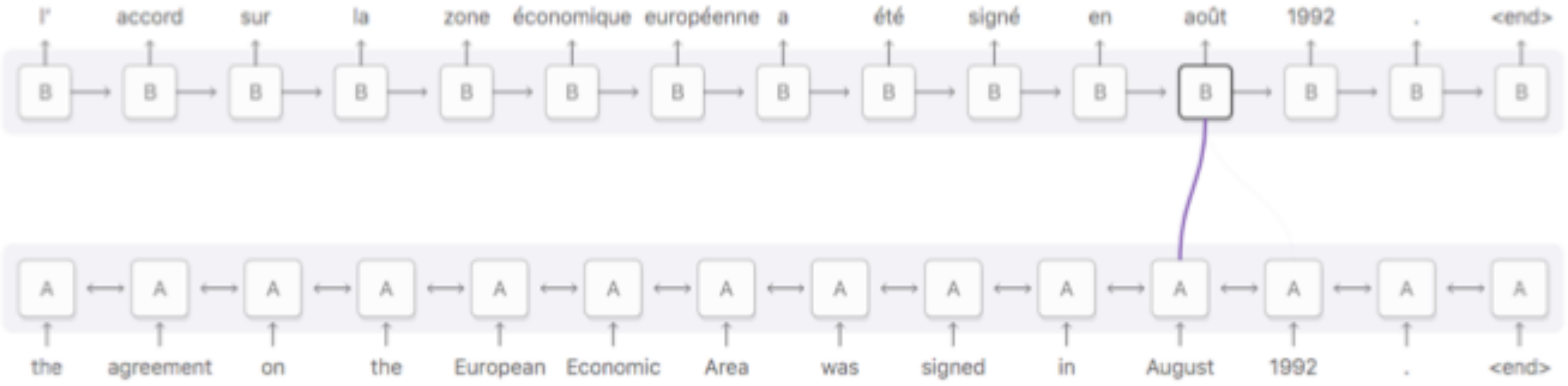


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

Example: multi-dependency

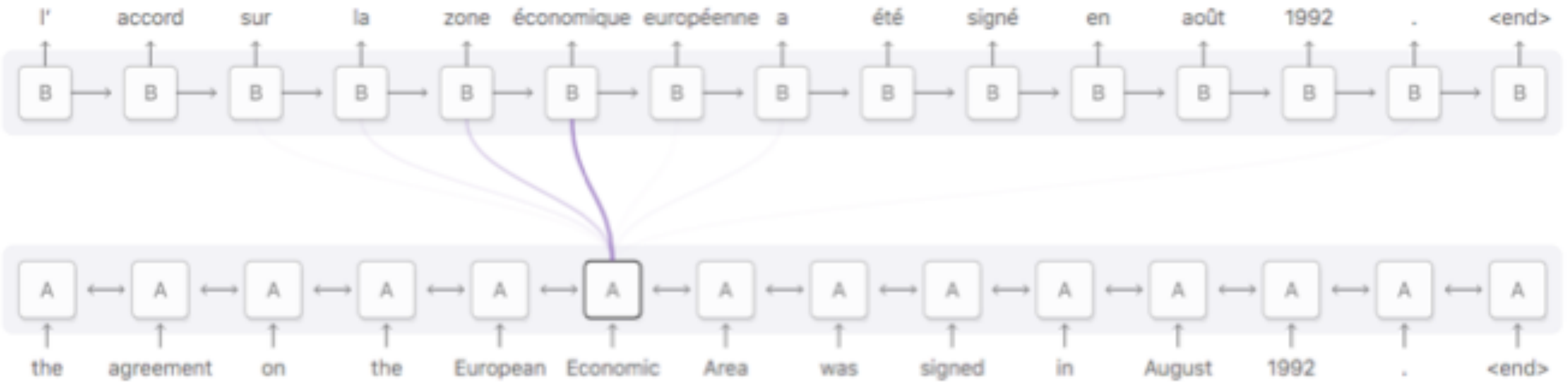


Diagram derived from Fig. 3 of Bahdanau, *et al.* 2014

Attention in Speech

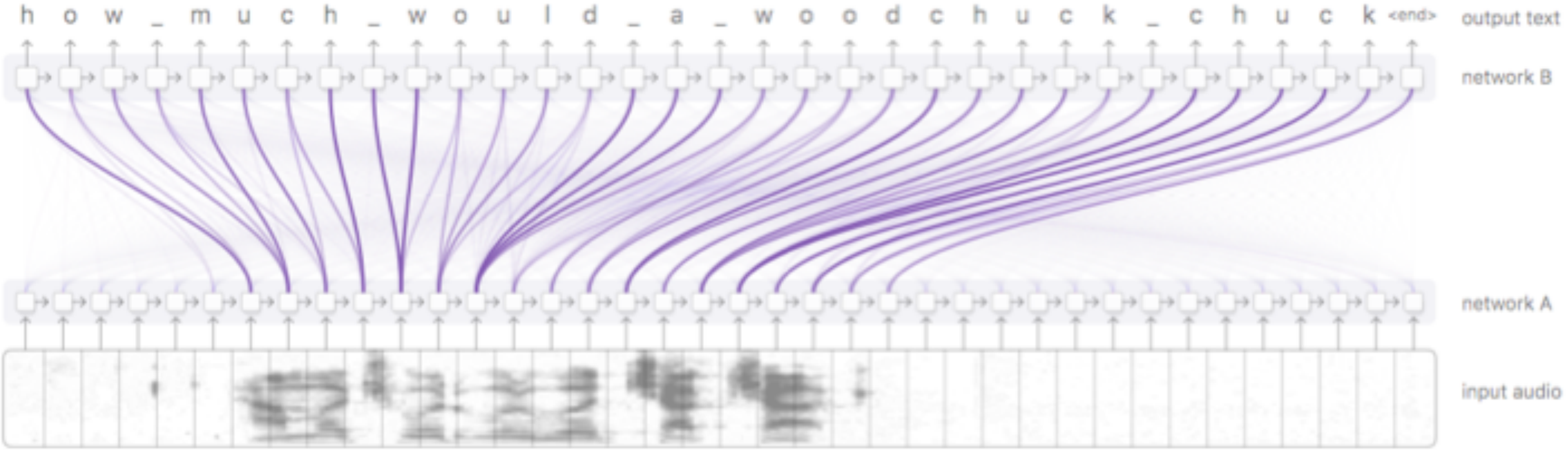
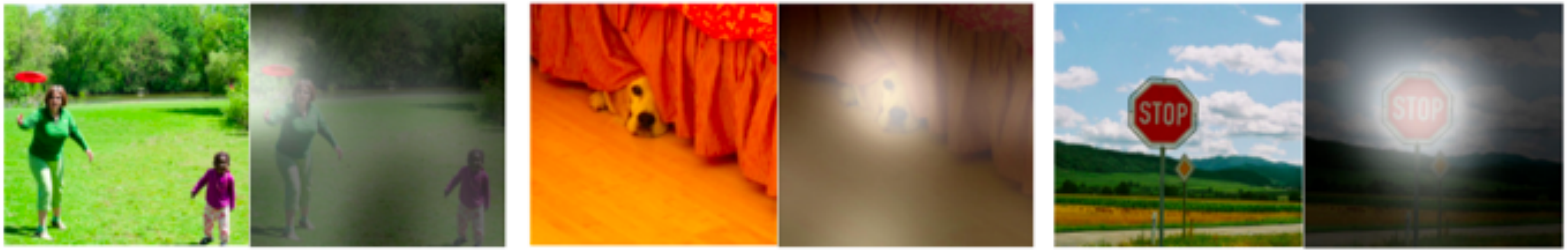

h o w _ m u c h _ w o u l d _ a _ w o o d c h u c k _ c h u c k <end>
output text
network B
network A
input audio

Figure derived from Chan, *et al.* 2015

Attention in Images



A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

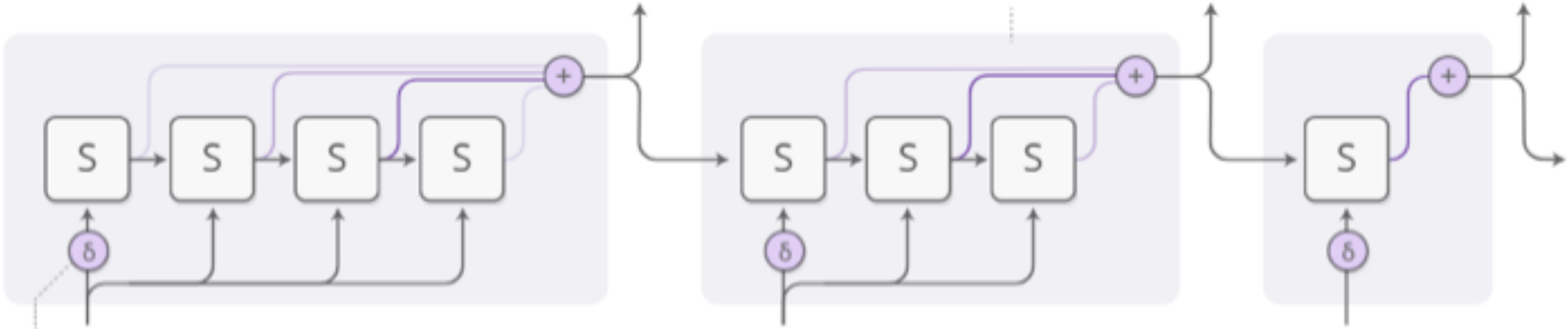
Figure from [3]

General Principle

- Everything in neural net needs to be differentiable (→ learning with backprop!)
- Model discrete selections (single outputs) as continuous selections (select all with different weight)
- Neural networks become “computational graphs”

Adaptive Computation Time

- Allow RNN to execute variable amounts of computation for each timestep?
- How many timesteps? …attention!



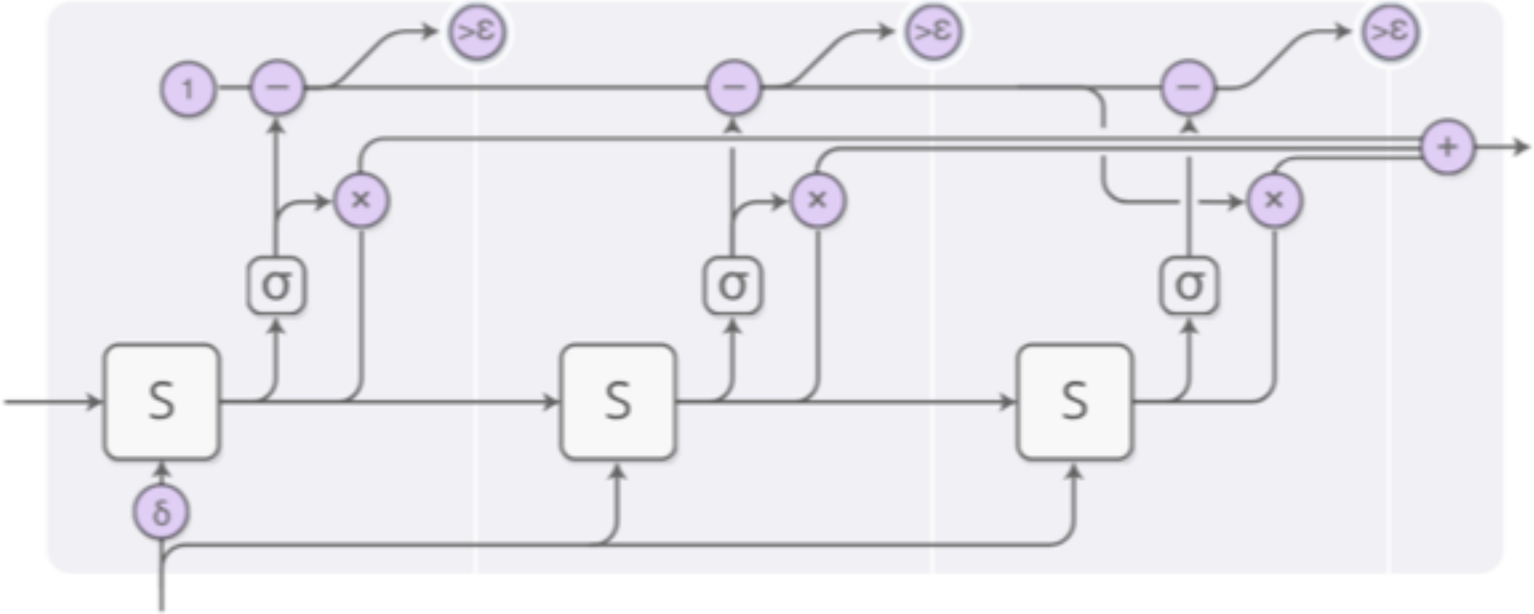
For every time step the RNN can do multiple computation steps.

The output is a weighted combination of the computation step outputs.

The process is repeated for each time step.

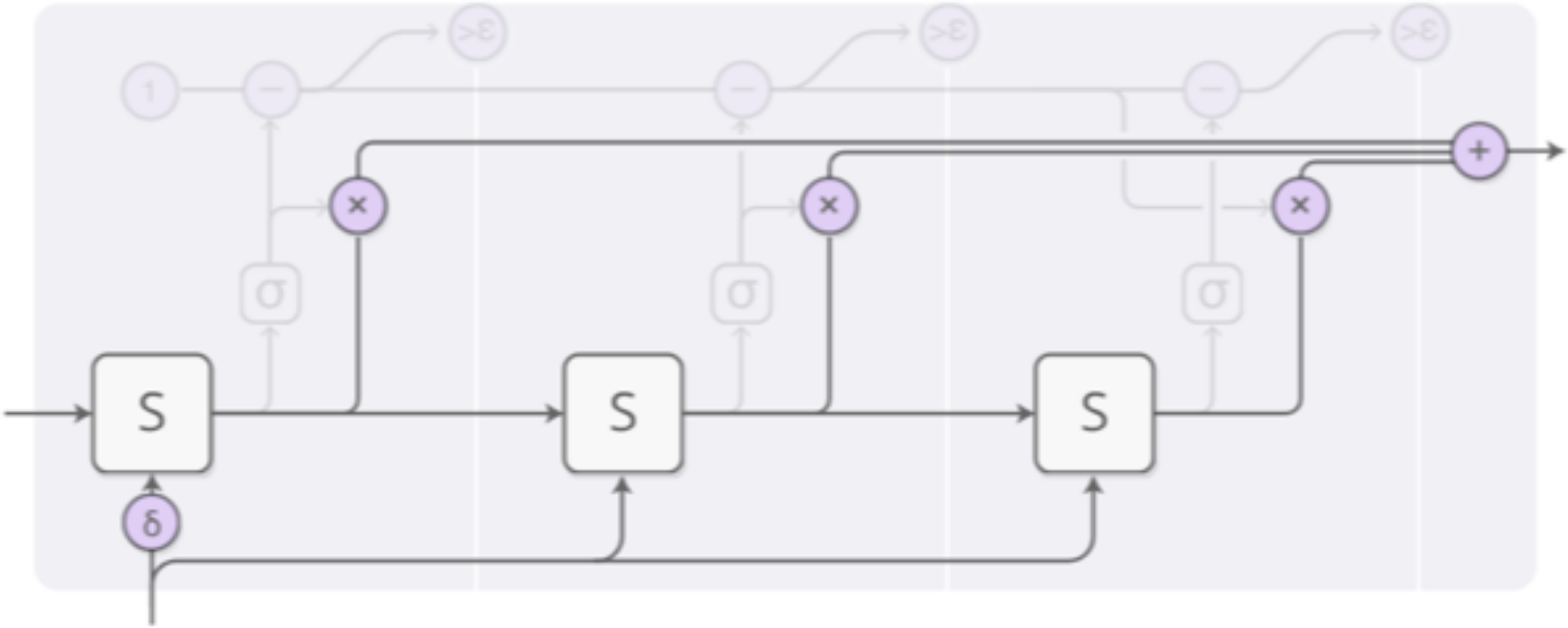
A special bit is set to denote the first computation step.

Adaptive Compute Time



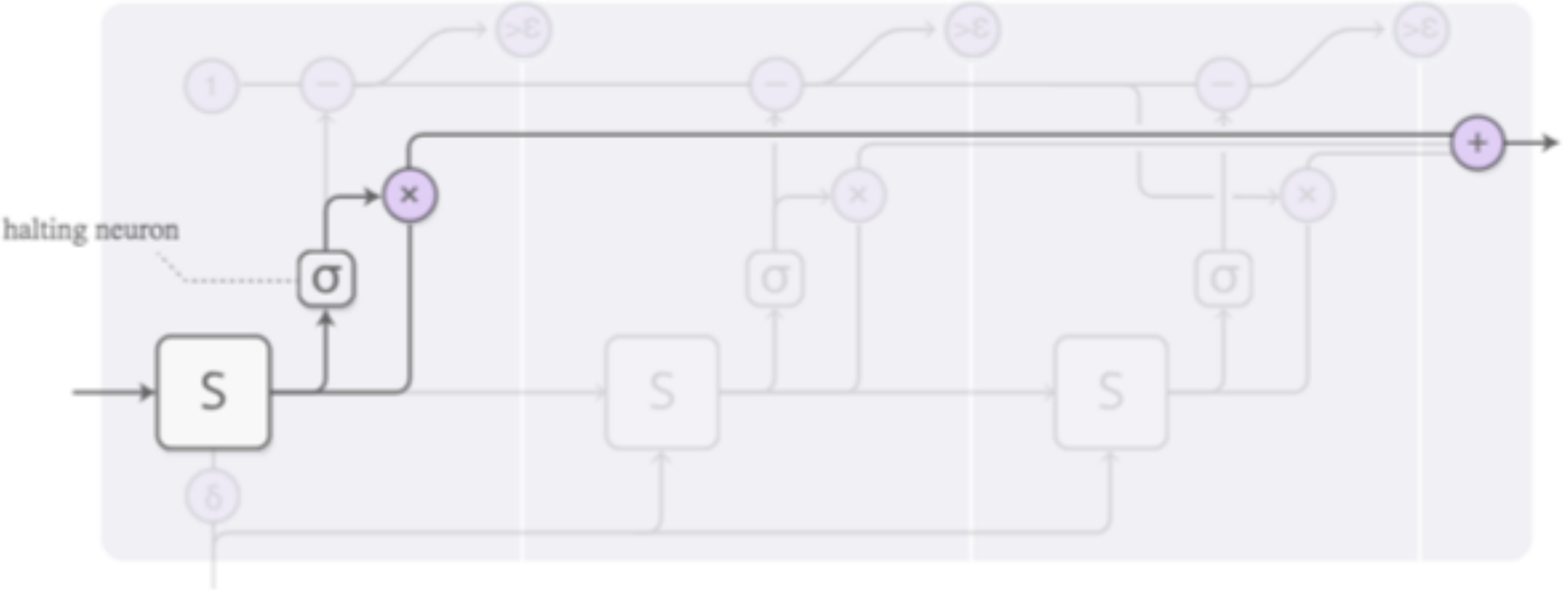
ACT in detail

Adaptive Compute Time



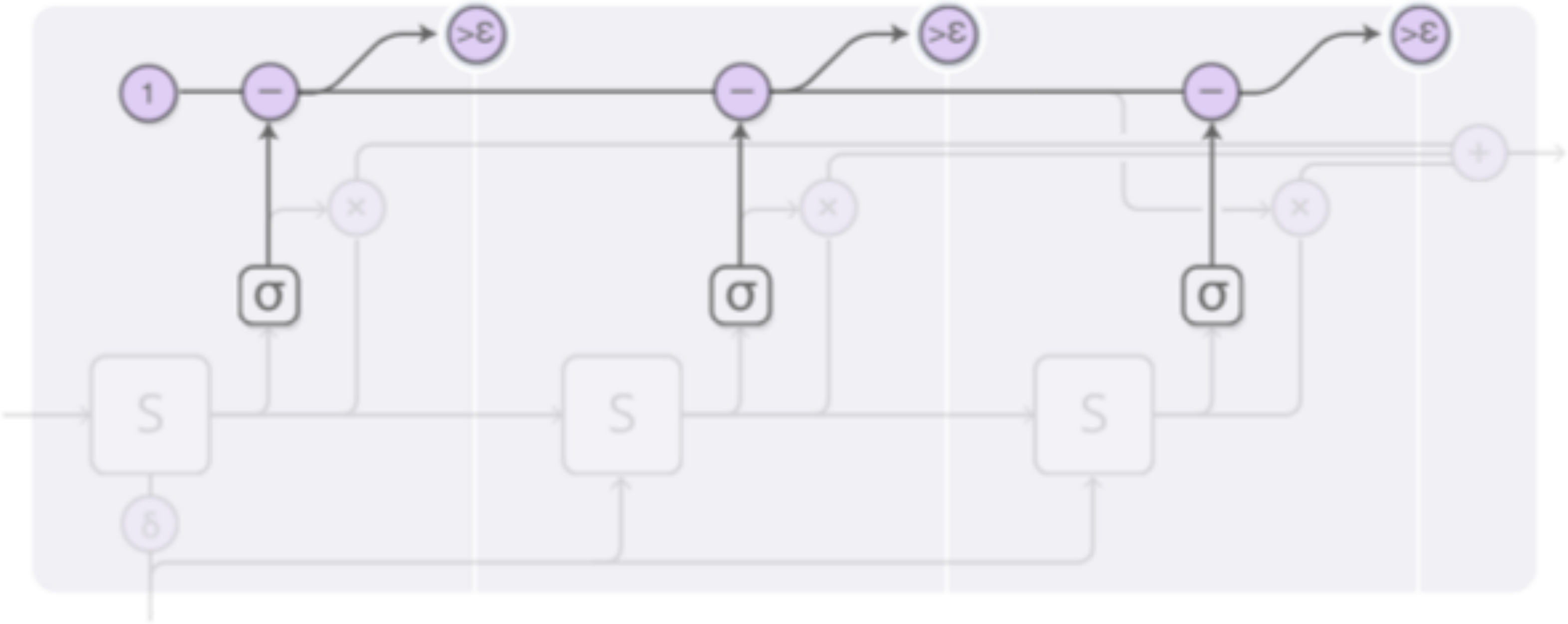
Output: weighted combination of states

Adaptive Compute Time



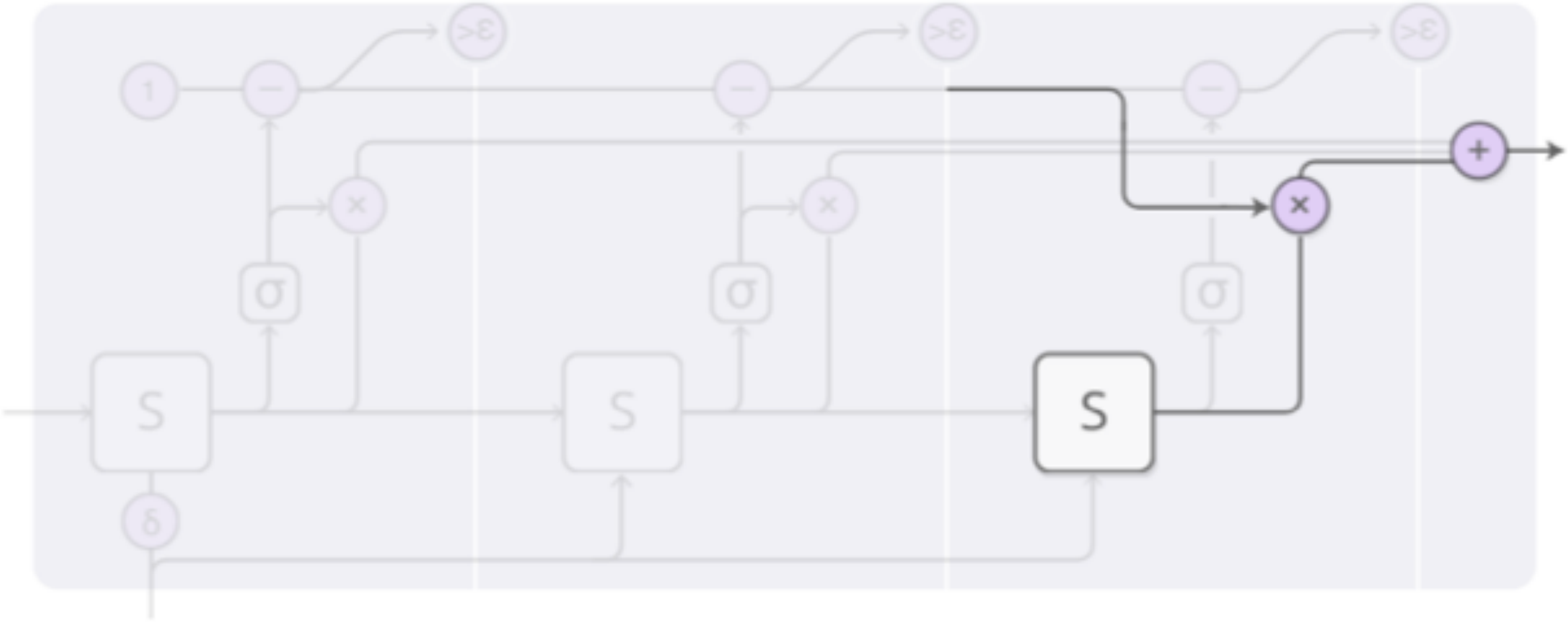
Individual weights determined by “halting neuron”
(sigmoid activation, read “likelihood to stop here”)

Adaptive Compute Time



Make sure that weights sum up to 1!
Stop when no weight is left

Adaptive Compute Time



Add residual weight to output by forcing last state

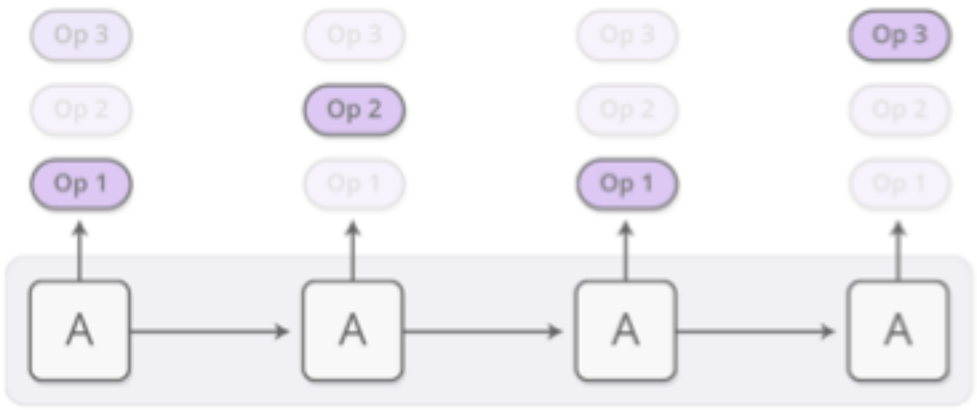
Still not creepy enough?

There is even more!

Neural Programmer

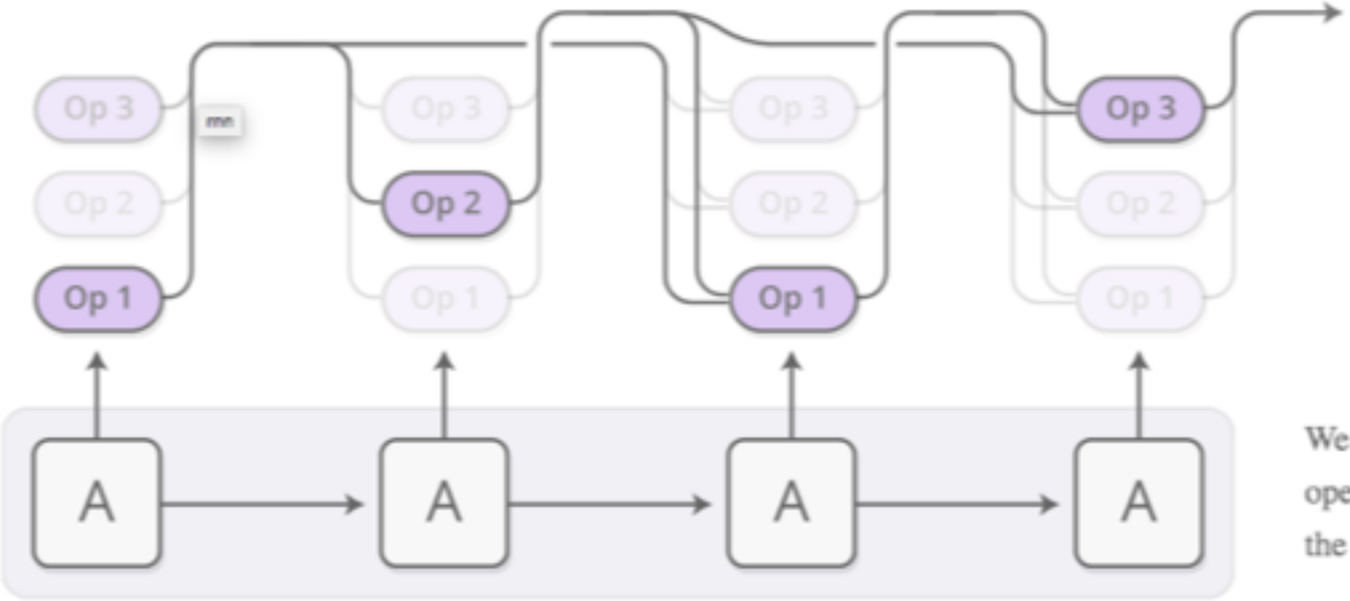
- How about modeling actions/operations?
- Like arithmetic, loops, etc.?

Model as distribution of operations



At each step the controller RNN outputs a probability distribution.

Neural Programmer



We run all of the operations and average the outputs together.

…and use attention to make it differentiable!