Advanced Recurrent Architectures

Davide Bacciu

Dipartimento di Informatica
Università di Pisa

Intelligent Systems for Pattern Recognition (ISPR)

Lecture Outline

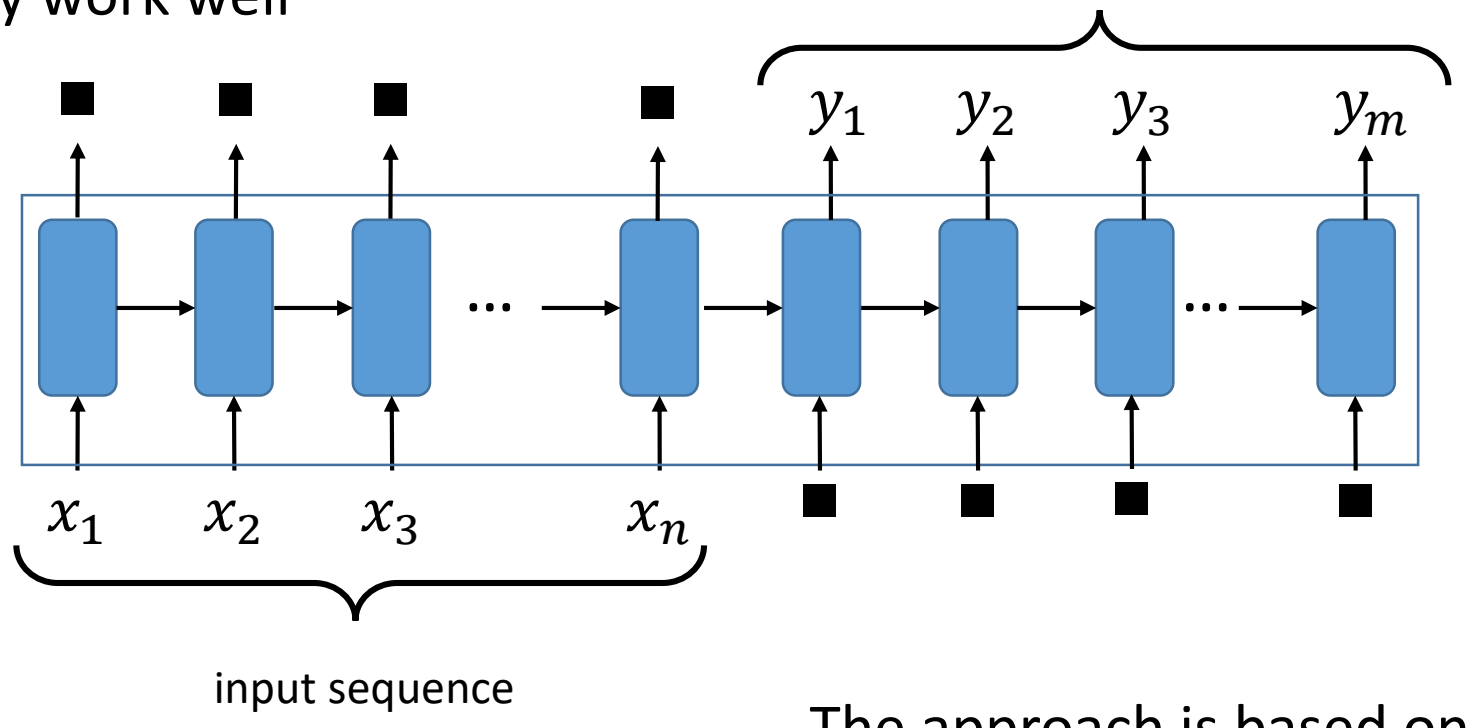
- Dealing with structured/compound data
  - Sequence-to-sequence
  - Attention models
- Dealing with very long-term dependencies
  - Multiscale networks
  - Adding memory components
- Neural reasoning
  - Neural Turing machines

Basic Gated RNN (GRNN) Limitations

- GRNN are excellent to handle size/topology varying data in input
  - How can we handle size/topology varying outputs?
  - Sequence-to-sequence
- Structured data is compound information
  - Efficient processing needs the ability to focus on certain parts of such information
  - Attention mechanism
- GRNN have troubles dealing with very long-range dependencies
  - Introduce multiscale representation explicitly in the architecture
  - Introduce external memory components

Learning to Output Variable Length Sequences

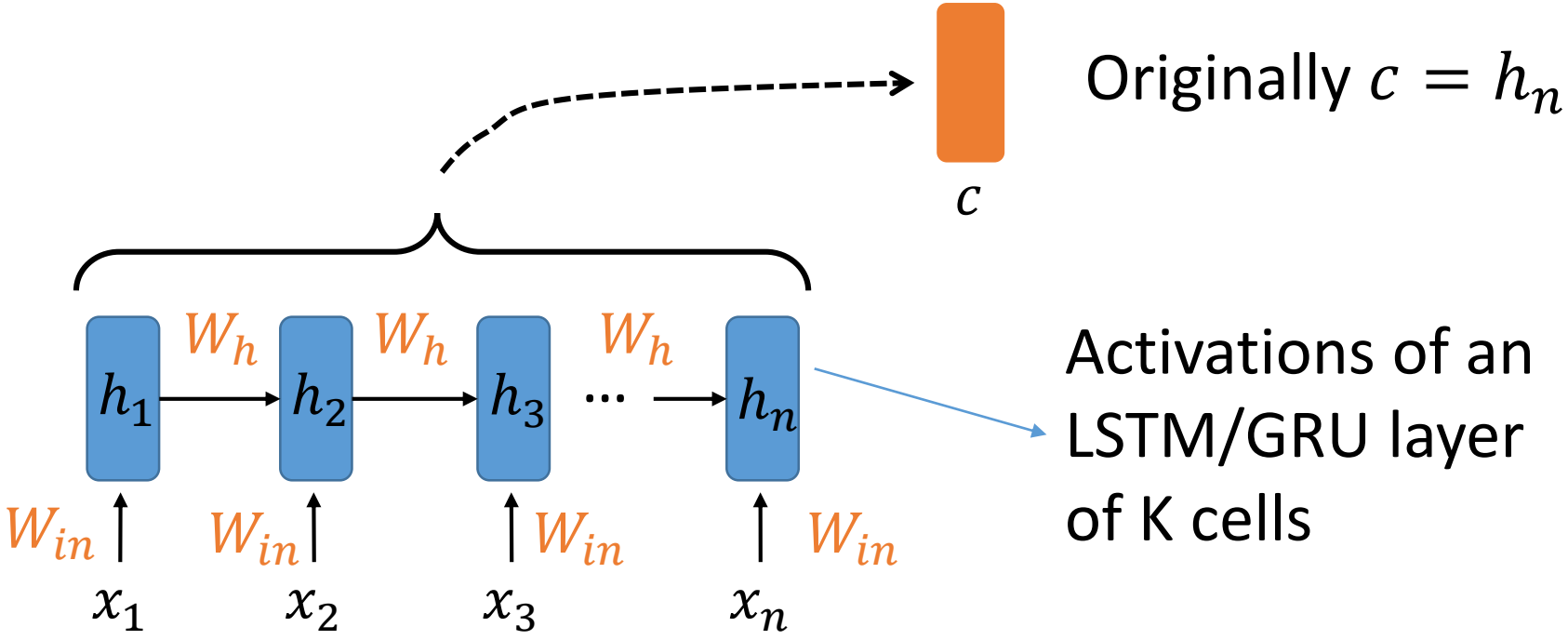
The idea of an unfolded RNN with blank inputs-outputs does not really work well



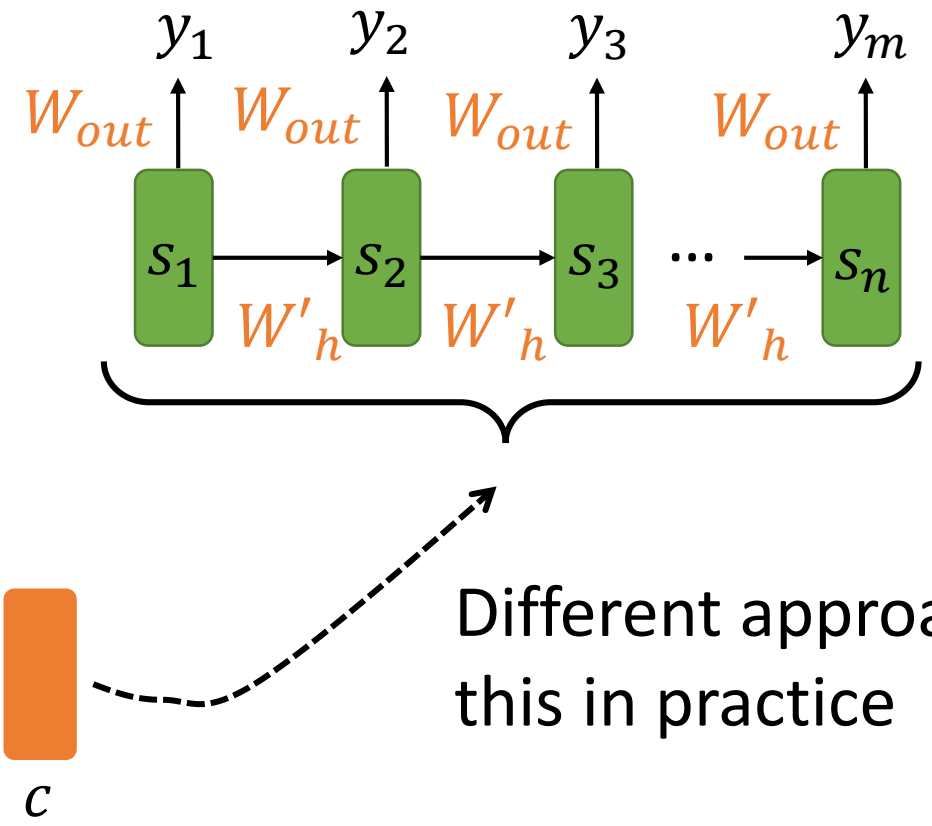
The approach is based on an encoder-decoder scheme

Encoder

Produce a compressed and fixed length representation c of all the input sequence $x_1, \dots, x_n$



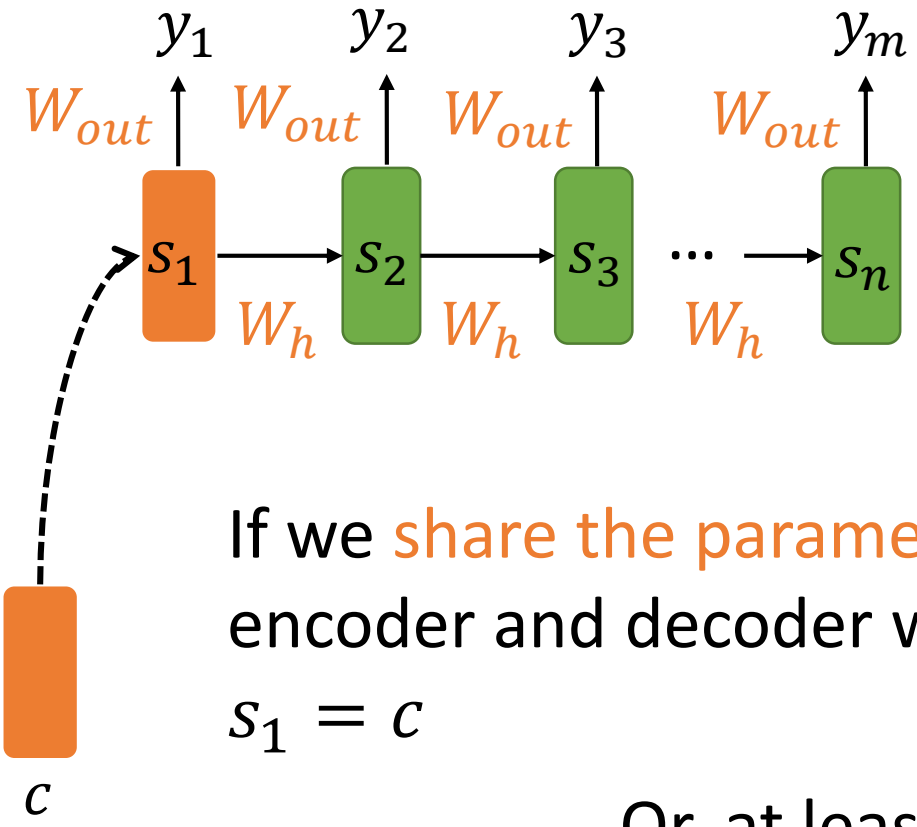
Decoder



A LSTM/GRU layer of K cells seeded by the context vector c

Different approaches to realize this in practice

Decoder

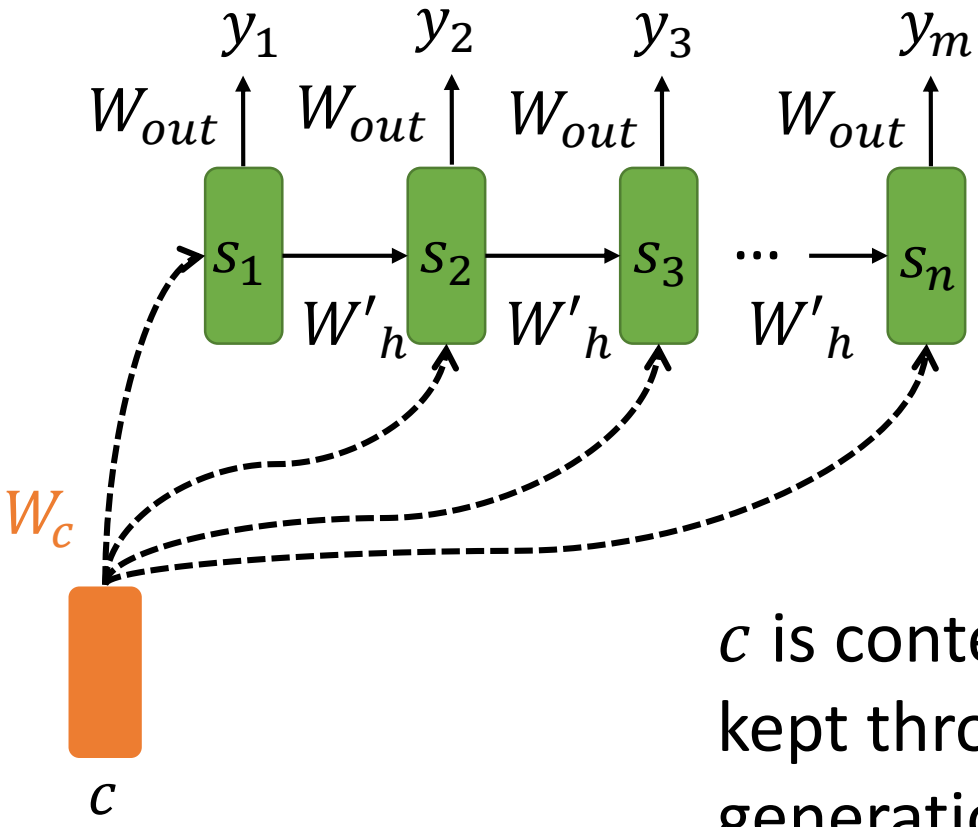


We risk to lose memory of c soon

If we share the parameters between encoder and decoder we can take $s_1 = c$

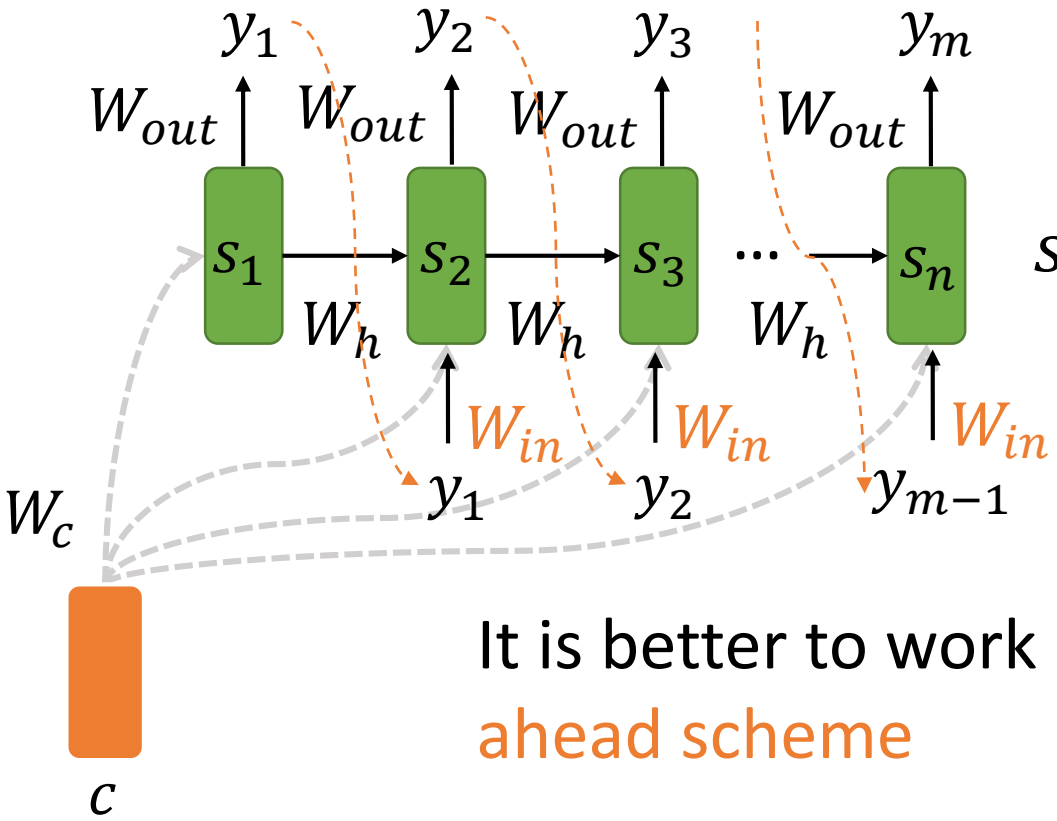
Or, at least, assume c and s_1 have compatible size

Decoder



c is contextual information kept throughout output generation

Decoder



$$s_i = f(c, s_{i-1}, y_{i-1})$$

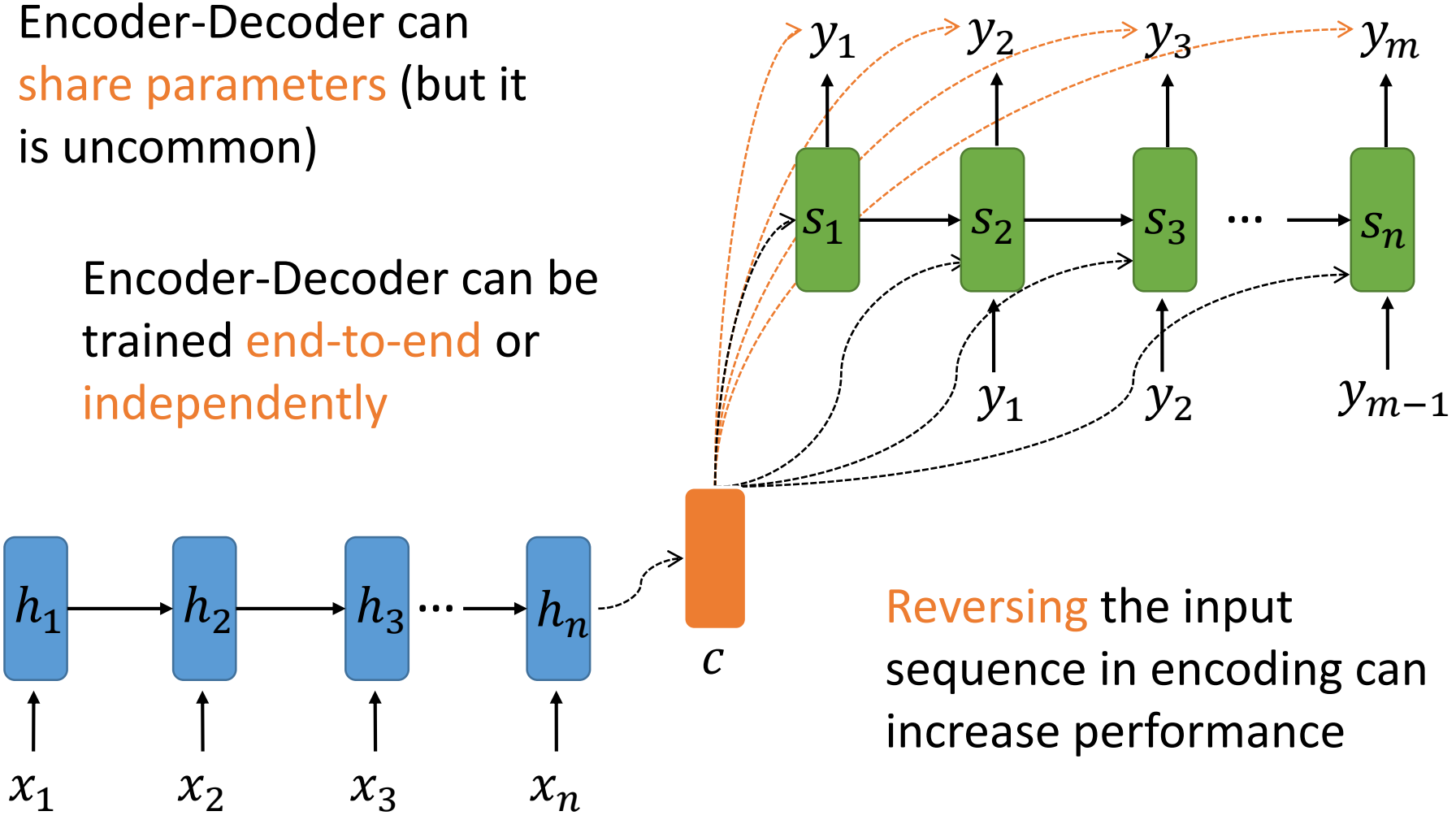
It is better to work on a one-step-ahead scheme

Remember teacher forcing (only) at training time

Sequence-To-Sequence Learning

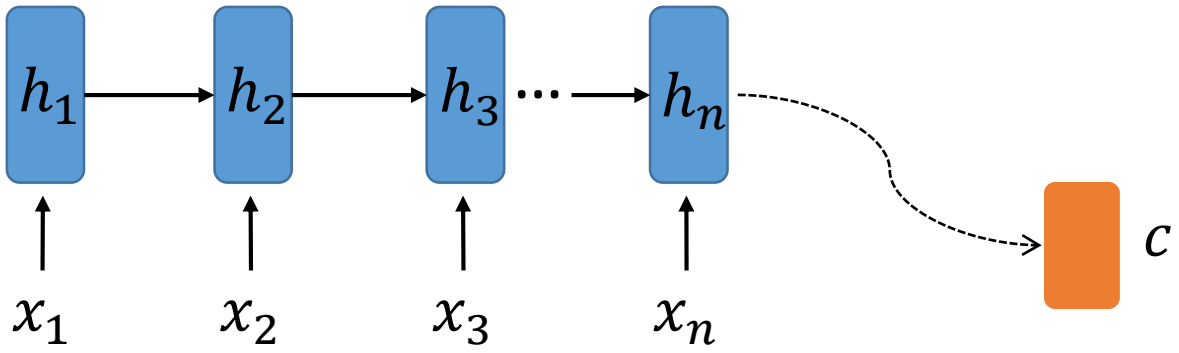
Encoder-Decoder can share parameters (but it is uncommon)

Encoder-Decoder can be trained end-to-end or independently



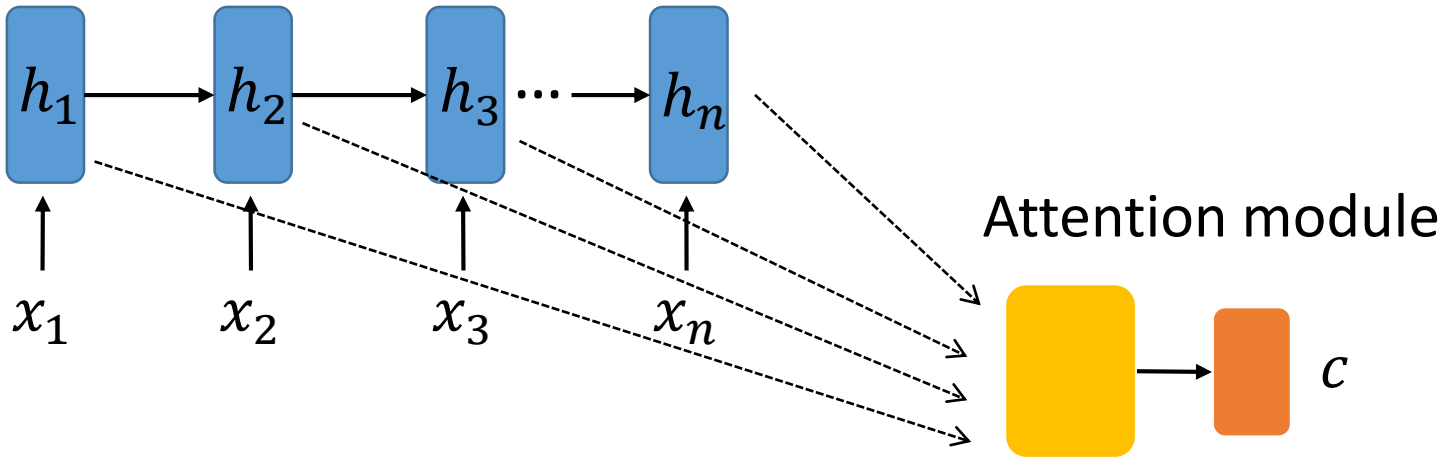
Reversing the input sequence in encoding can increase performance

On the Need of Paying Attention



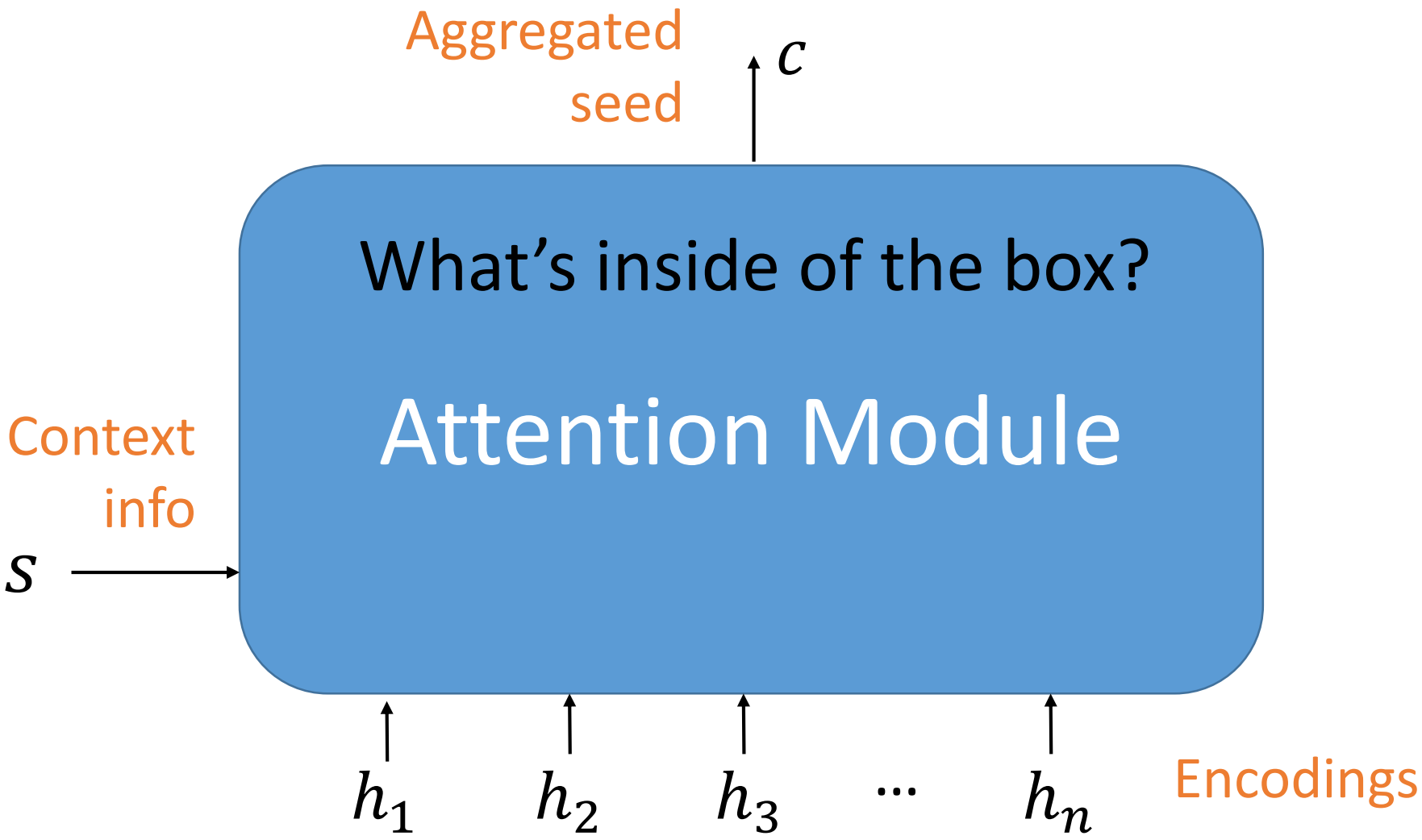
- Encoder-Decoder scheme assumes the hidden activation of the last input element summarizes sufficient information to generate the output
  - Bias toward most recent past
- Other parts of the input sequence might be very informative for the task
  - Possibly elements appearing very far from sequence end

On the Need of Paying Attention



- Attention mechanism select which part of the sequence to focus on to obtain a good c

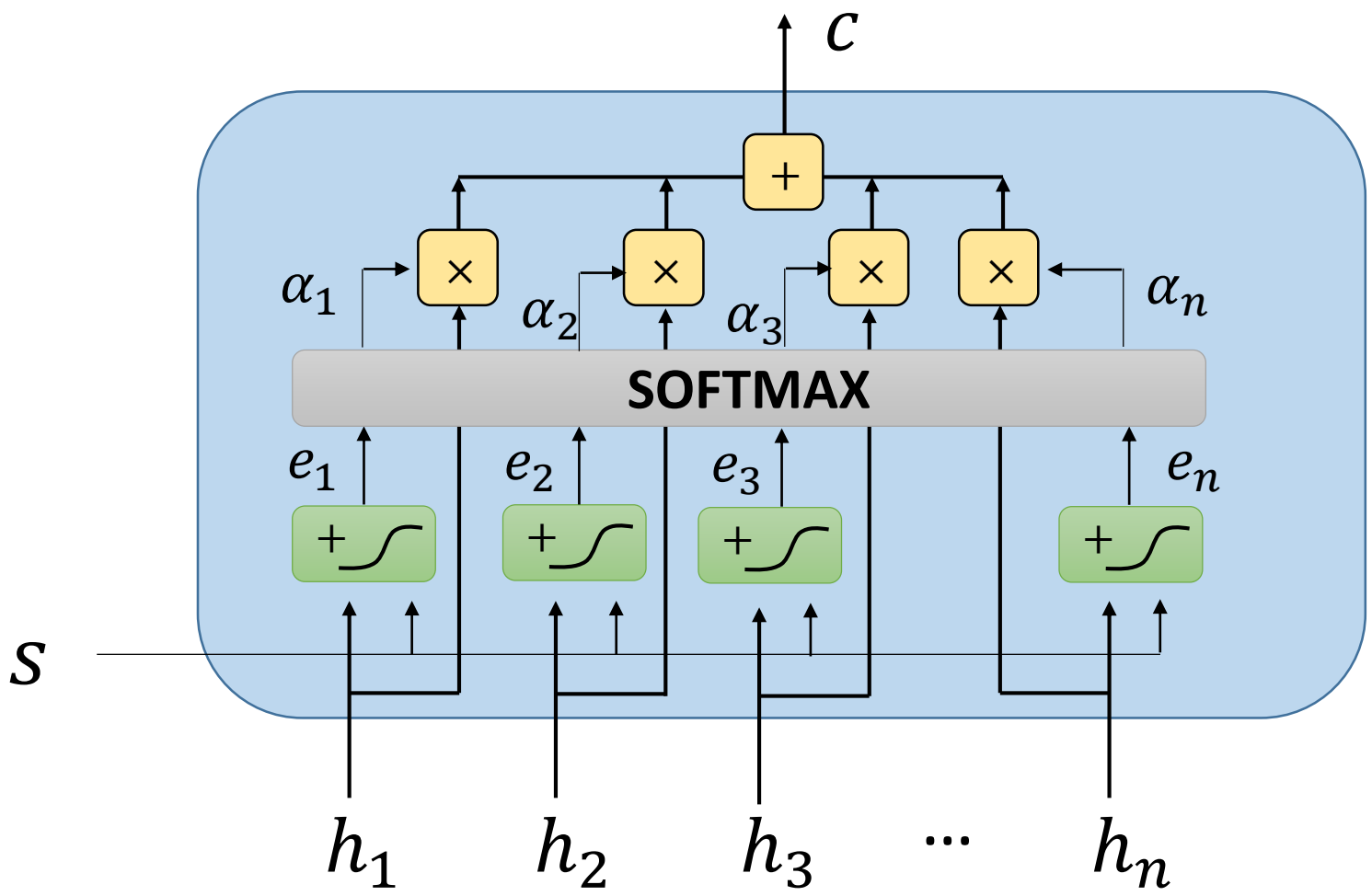
Attention Mechanisms – Blackbox View



What's inside of the box?

The Revenge of the Gates!

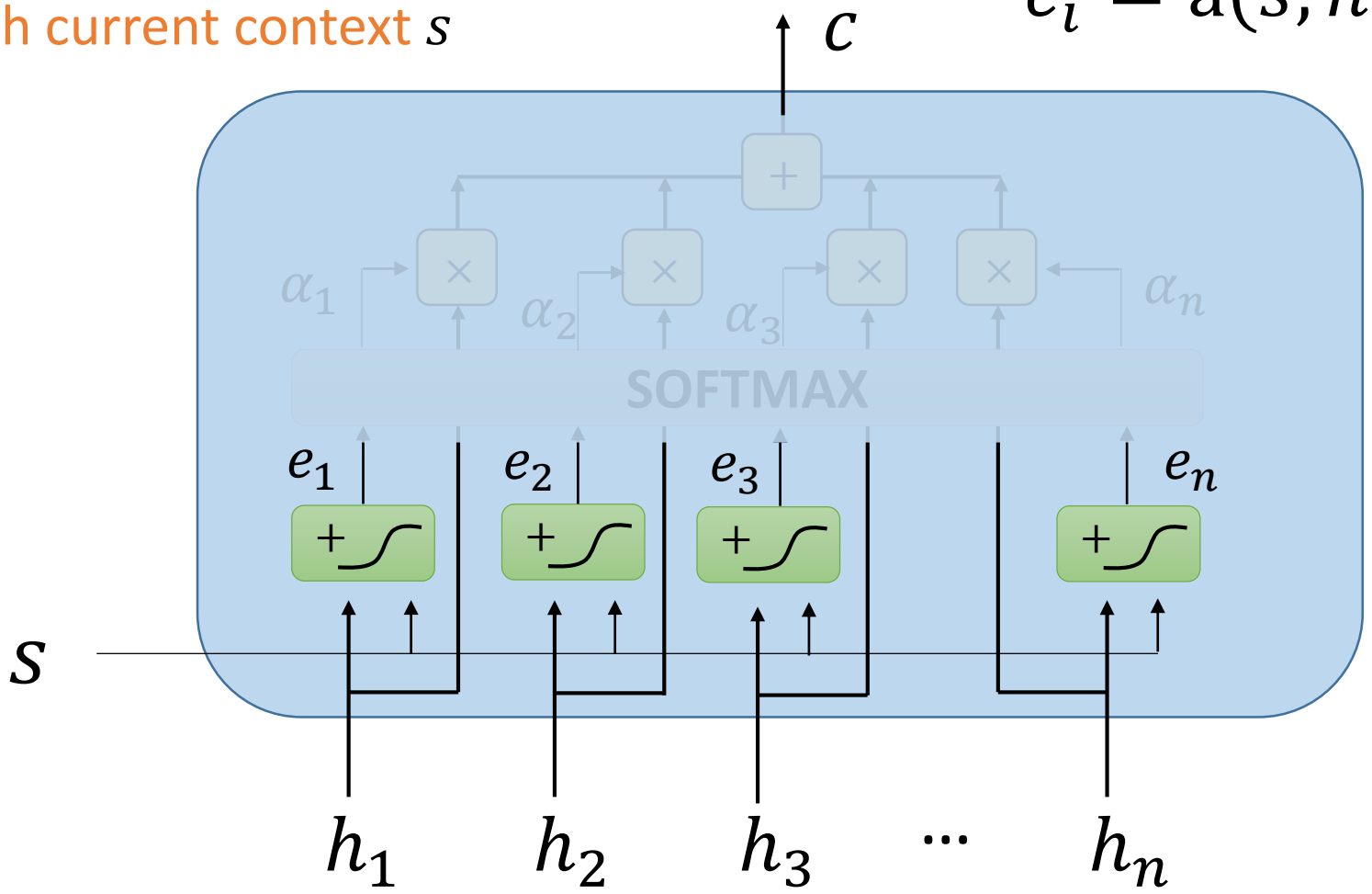
Opening the Box



Opening the Box – Relevance

Tanh layer fusing each encoding with current context s

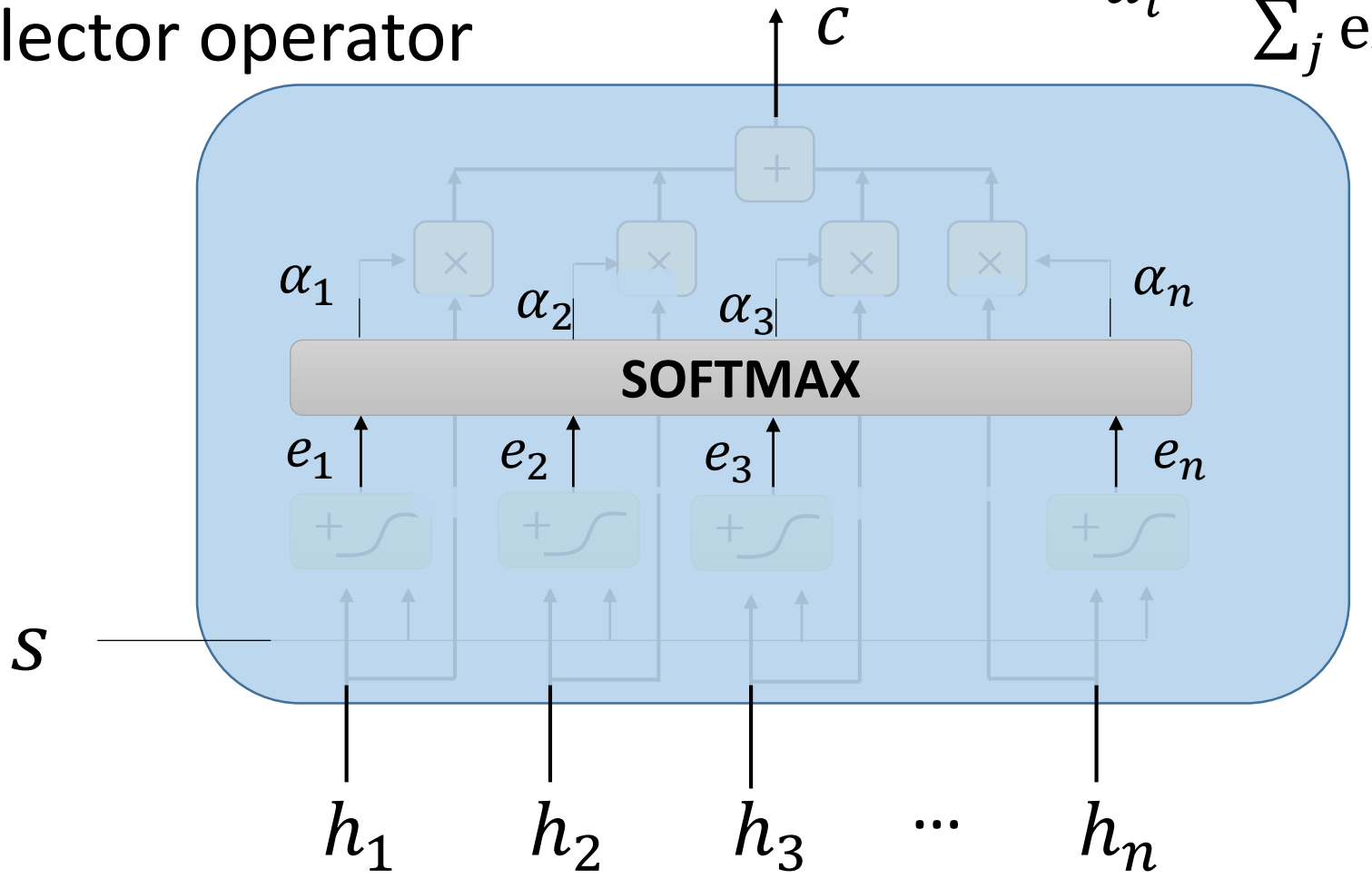
$$e_i = \mathrm{a}(s, h_i)$$



Opening the Box – Softmax

A differentiable max selector operator

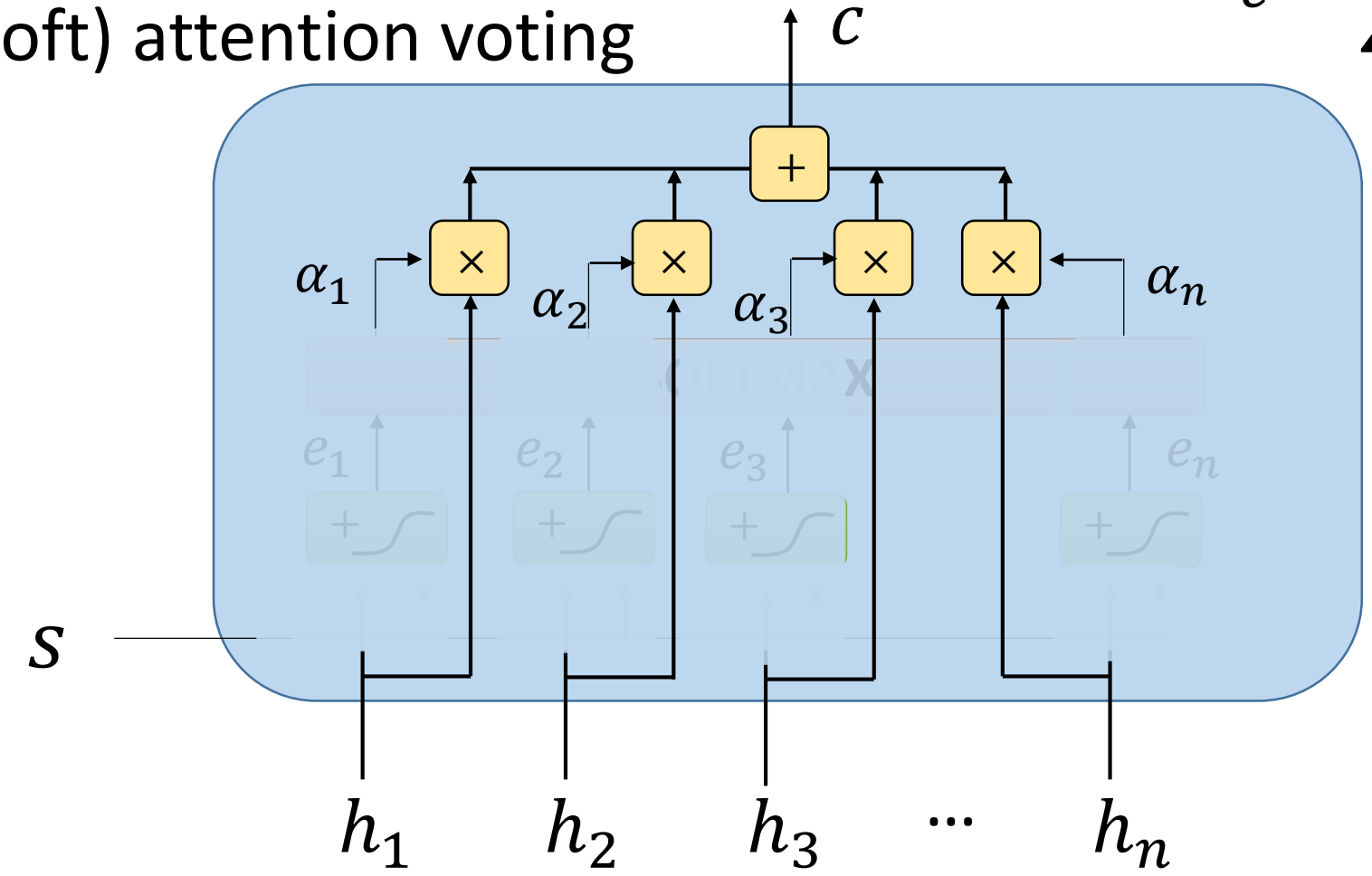
$$\alpha_i = \frac{\exp(e_i)}{\sum_j \exp(e_j)}$$



Opening the Box – Voting

Aggregated seed by (soft) attention voting

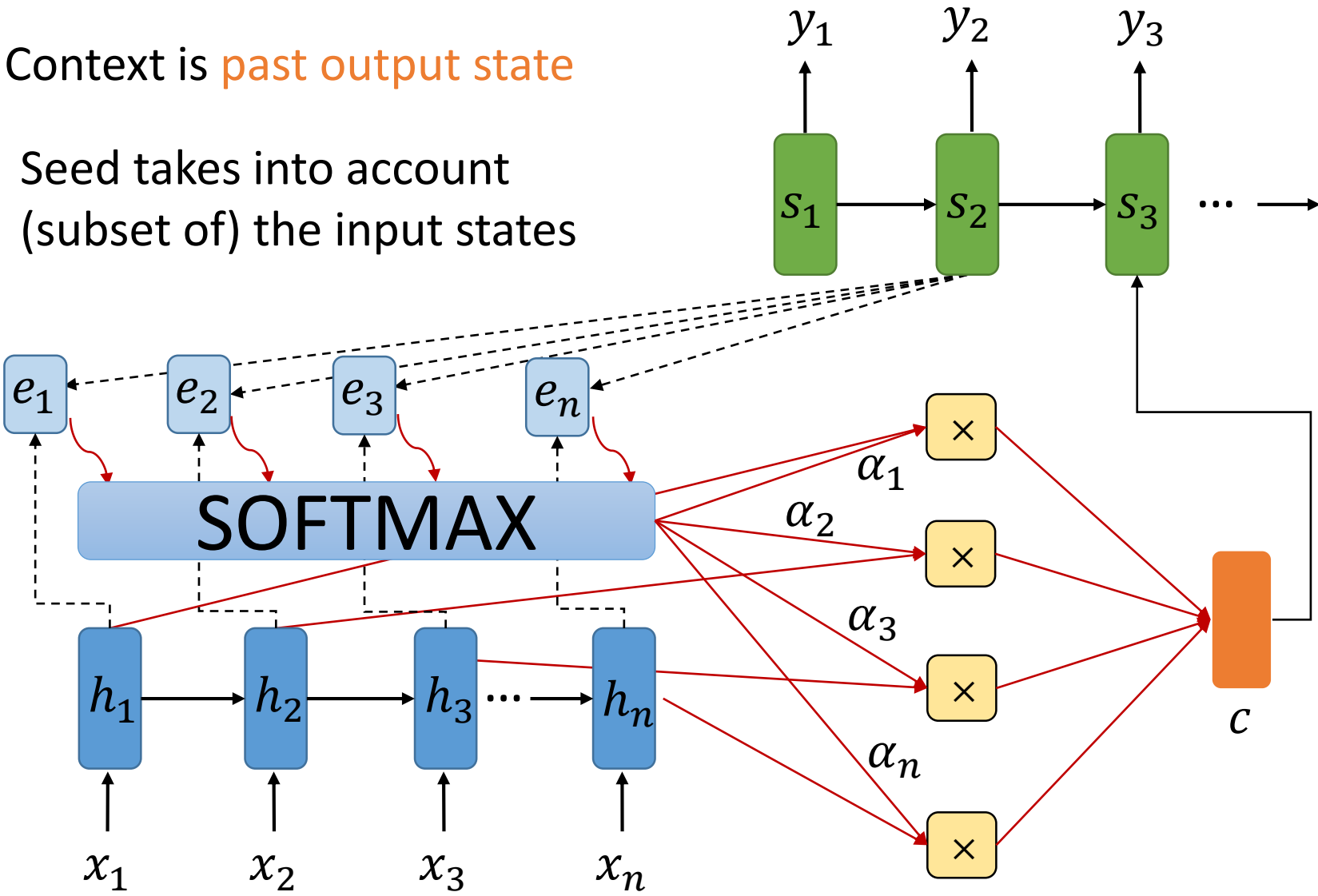
$$c = \sum_i \alpha_i h_i$$



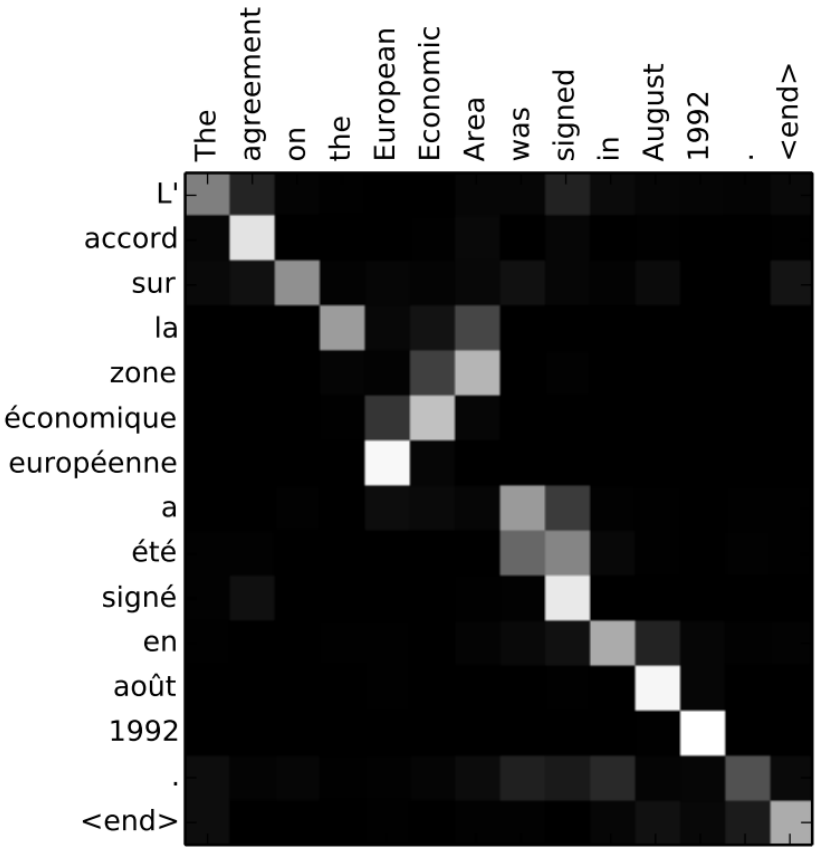
Attention in Seq2Seq

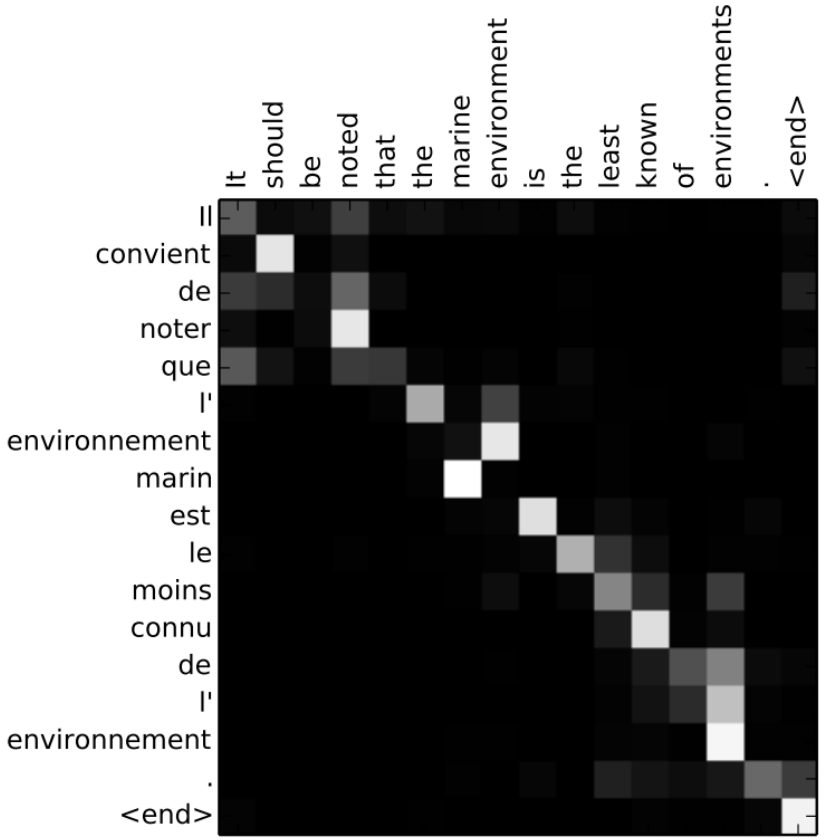
Context is past output state

Seed takes into account (subset of) the input states



Learning to Translate with Attention

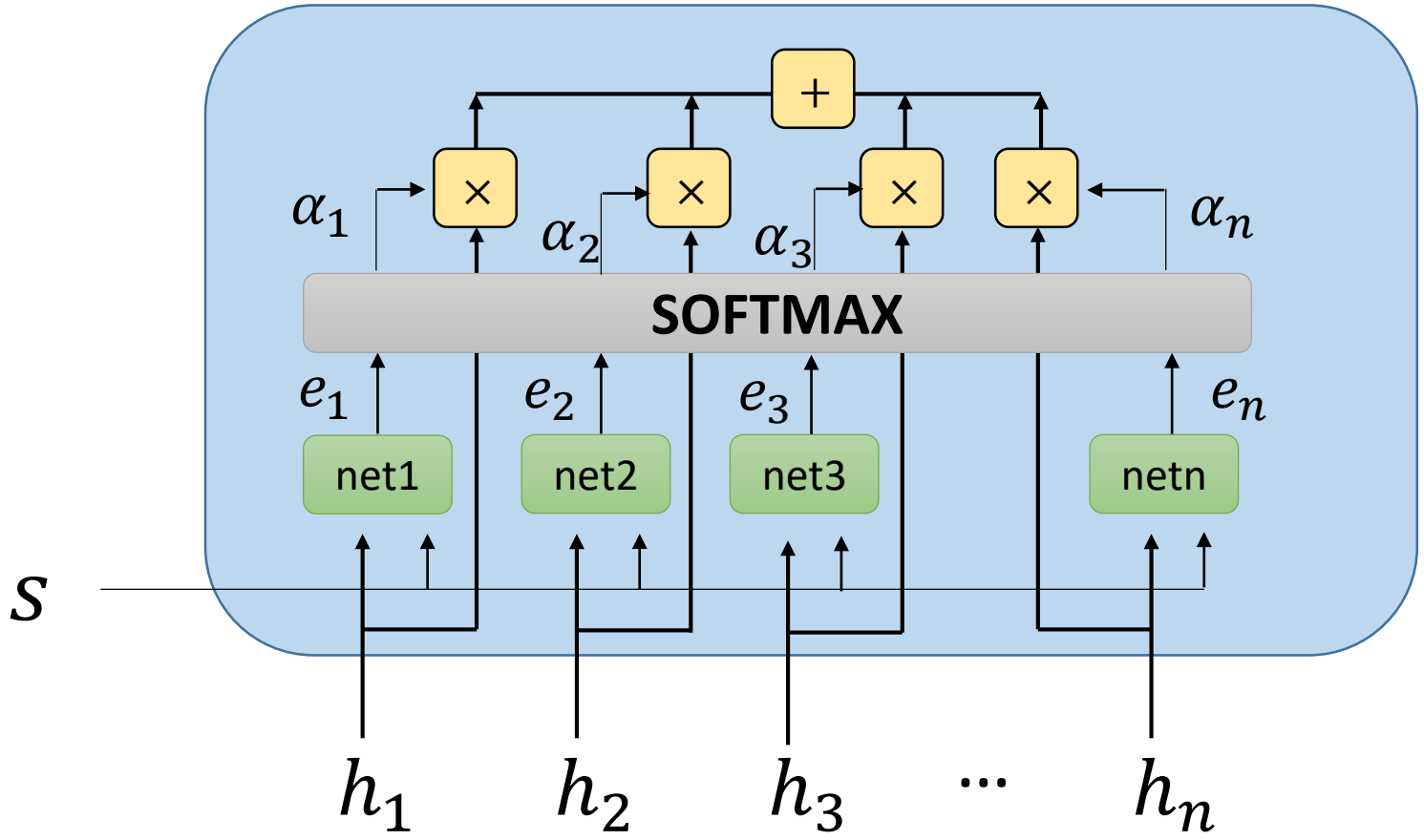




Bahdanau et al, Show, Neural machine translation by jointly learning to align and translate, ICLR 2015

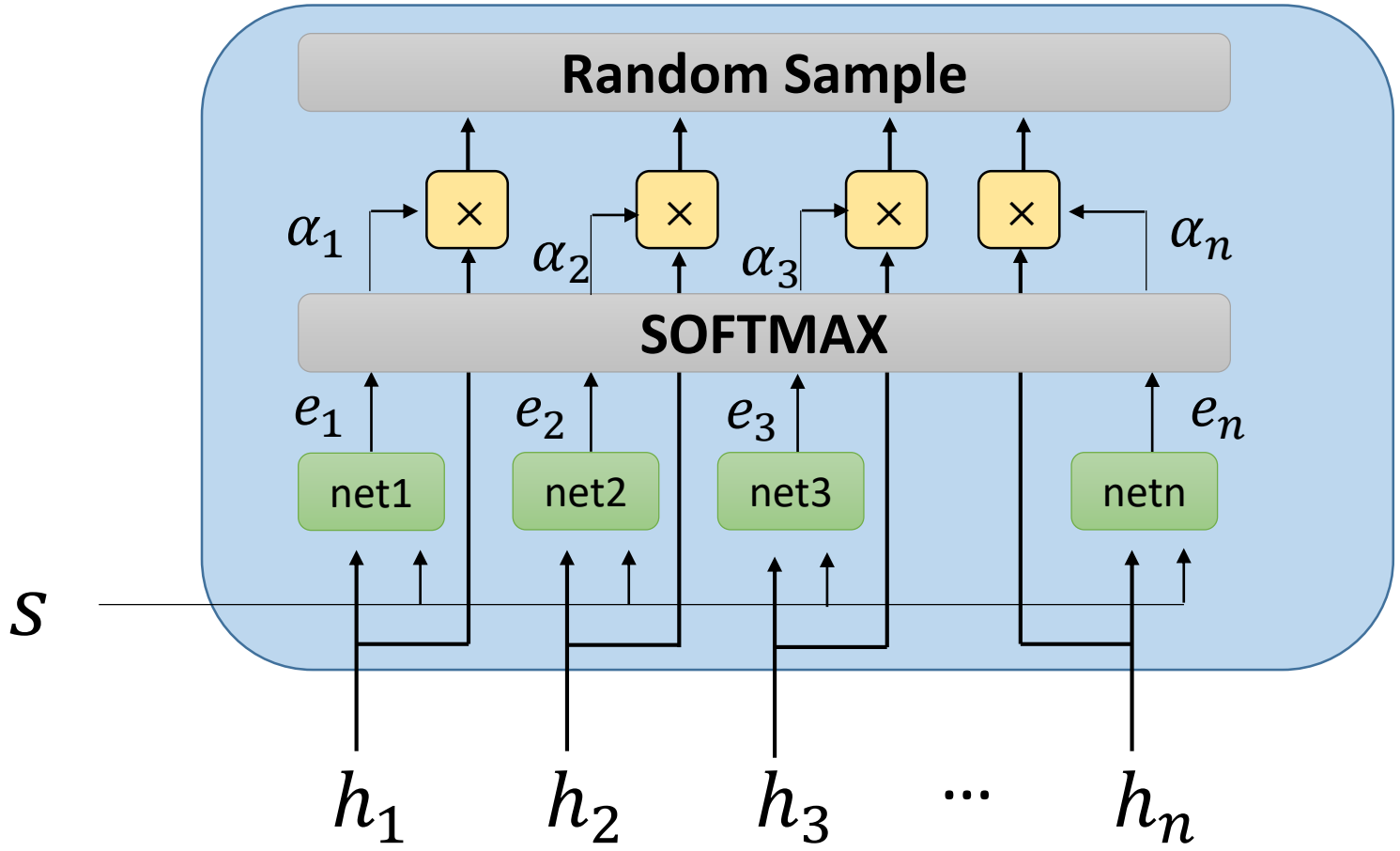
Advanced Attention – Generalize Relevance

This component determines how much each h is correlated/associated with current context s



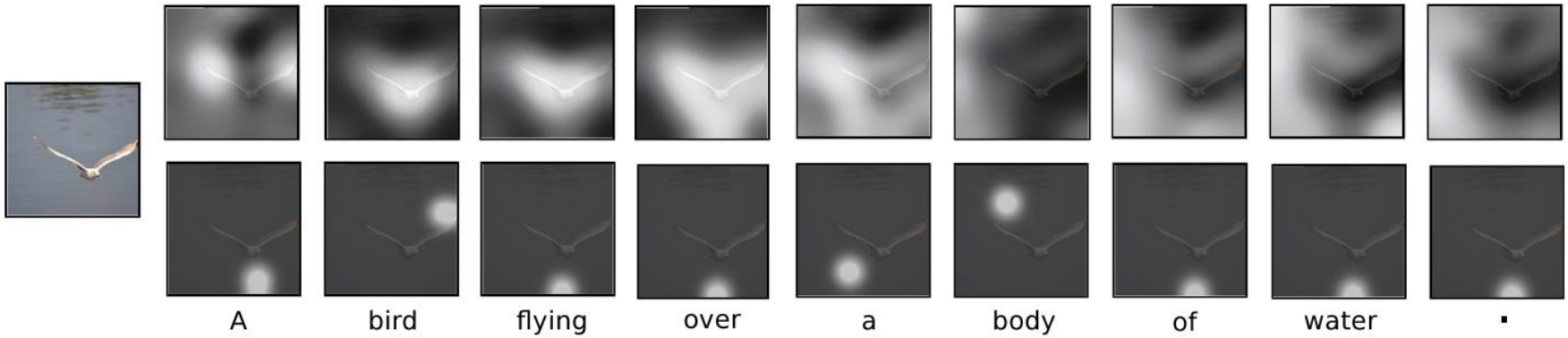
Advanced Attention – Hard Attention

Sample a single encoding using probability α_i



Attention-Based Captioning – Focus Shifting

Soft Attention



Hard Attention

Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

Attention-Based Captioning - Generation

Learns to correlate textual and visual concepts

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.

Helps understanding why the model fails

A large white bird standing in a forest.

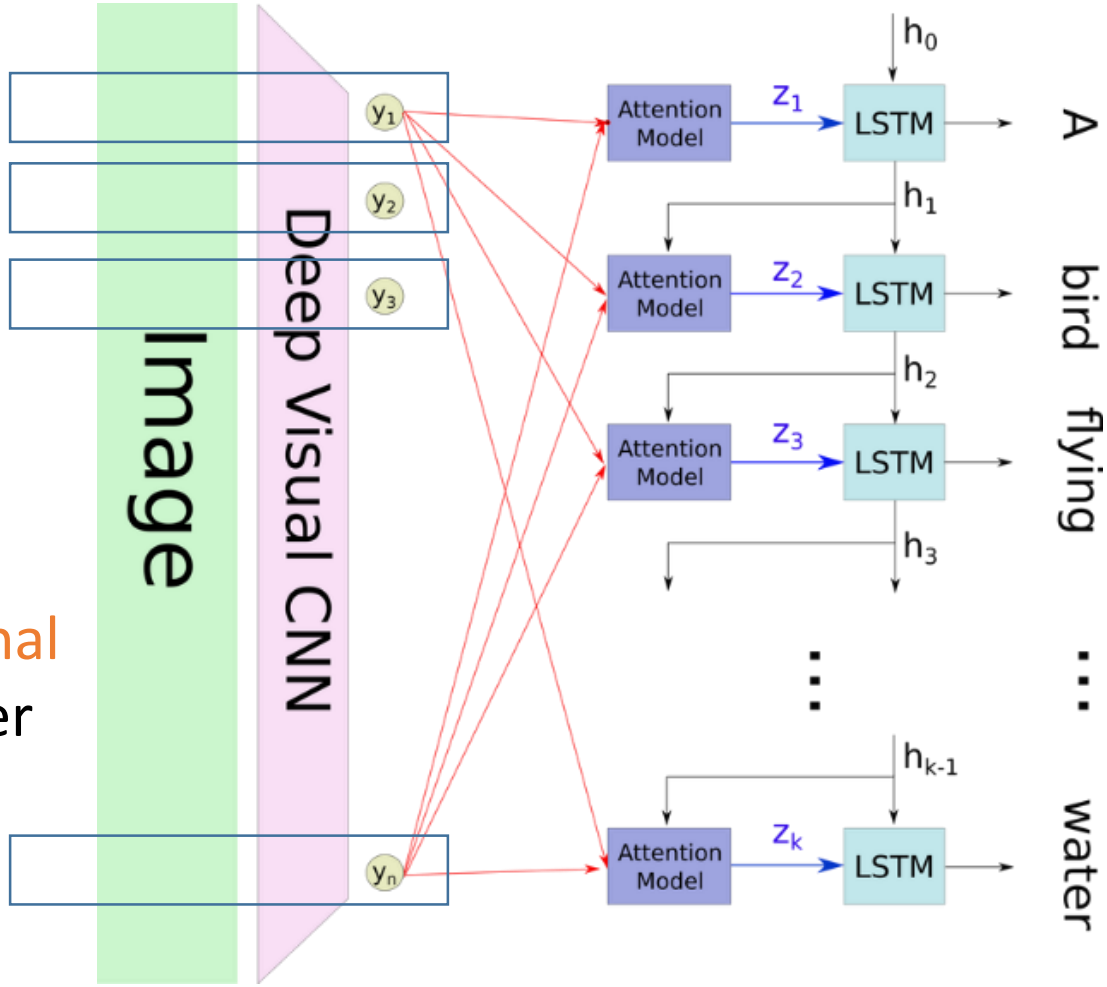
A woman holding a clock in her hand.

Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

Attention-Based Captioning – The Model

Encodings associated to *n* image regions

From convolutional layers rather than from fully connected



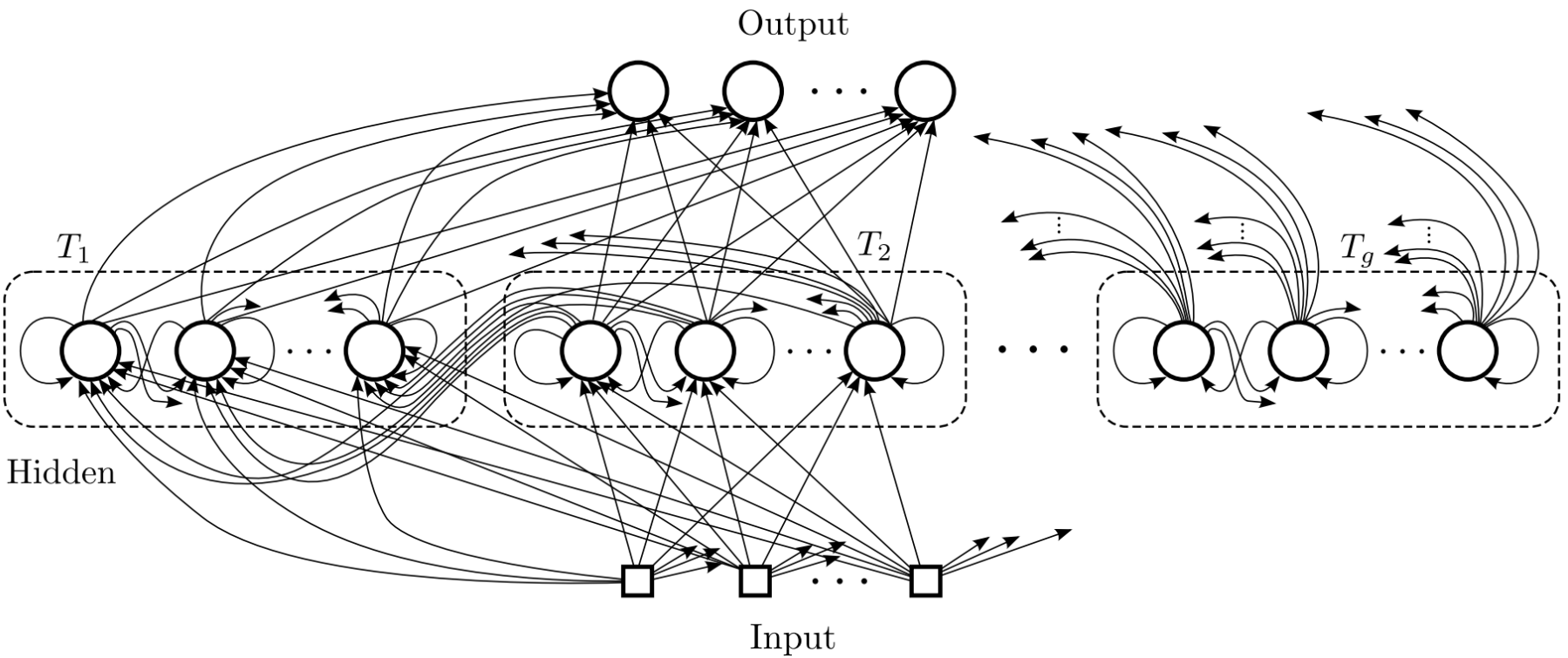
Xu et al, Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, ICML 2015

RNN and Memory – Issue 1

- Gated RNN claim to solve the problem of learning long-range dependencies
- In practice it is still difficult to learn on longer range
- Architectures trying to optimize dynamic memory usage
  - Clockwork RNN
  - Skip RNN
  - Multiscale RNN
  - Zoneout

Clockwork RNN

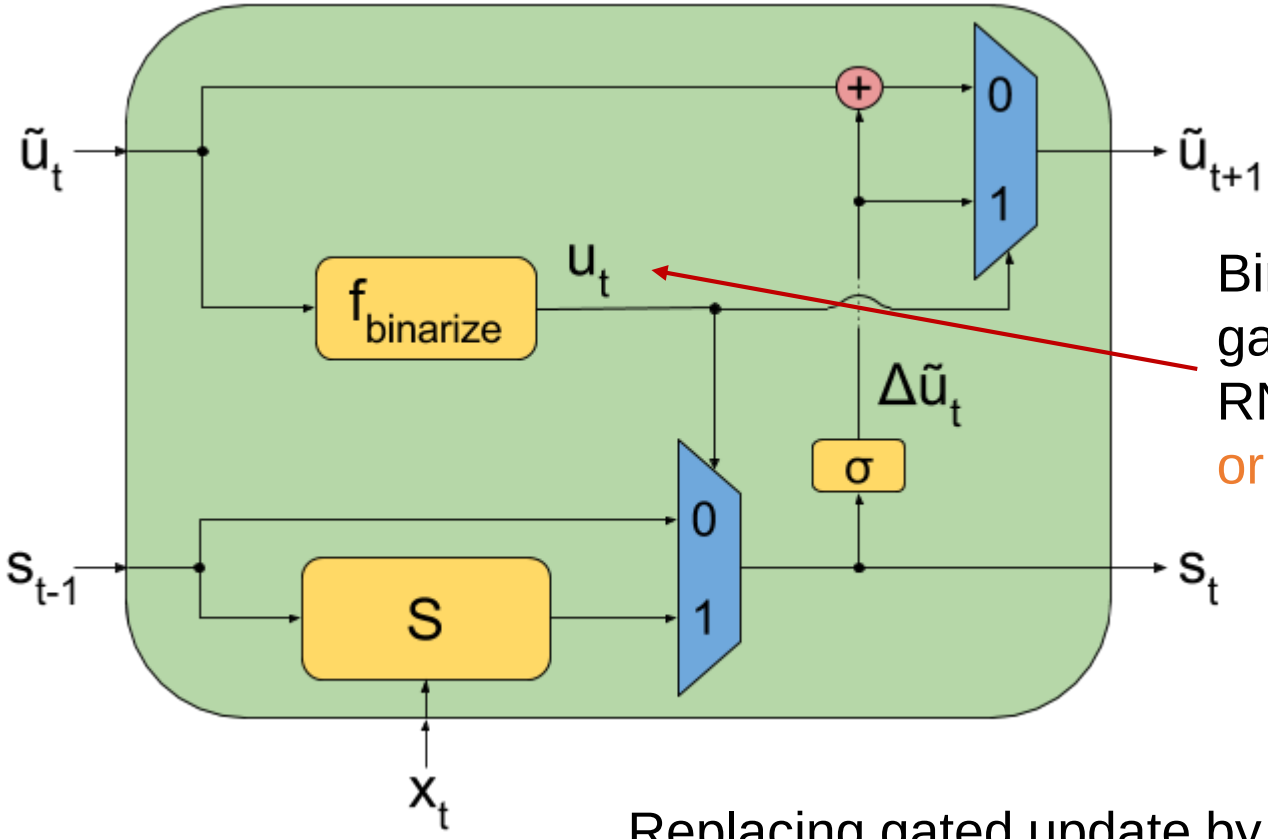
Modular recurrent layer where each module is updated at different clock



Modules interconnected only when destination clock time is larger

Koutnik et al, A Clockwork RNN, ICML 2014

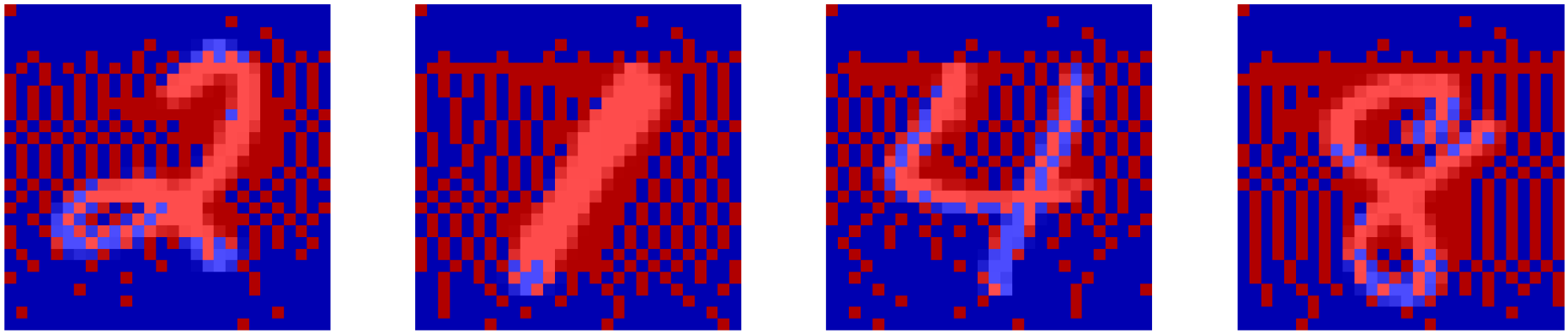
Skip RNN



Binary state update gate determining if RNN state is updated or copied

Replacing gated update by copying increases network memory (LSTM has an exponential fading effect due to the multiplicative gate)

Skip RNN and Attention



Attended pixels

Ignored pixels

Campos et al, Skip RNN: Skipping State Updates in Recurrent Neural Networks, ICLR 2018

RNN and Memory – Issue 2

A motivating example:

Task 3: Three Supporting Facts

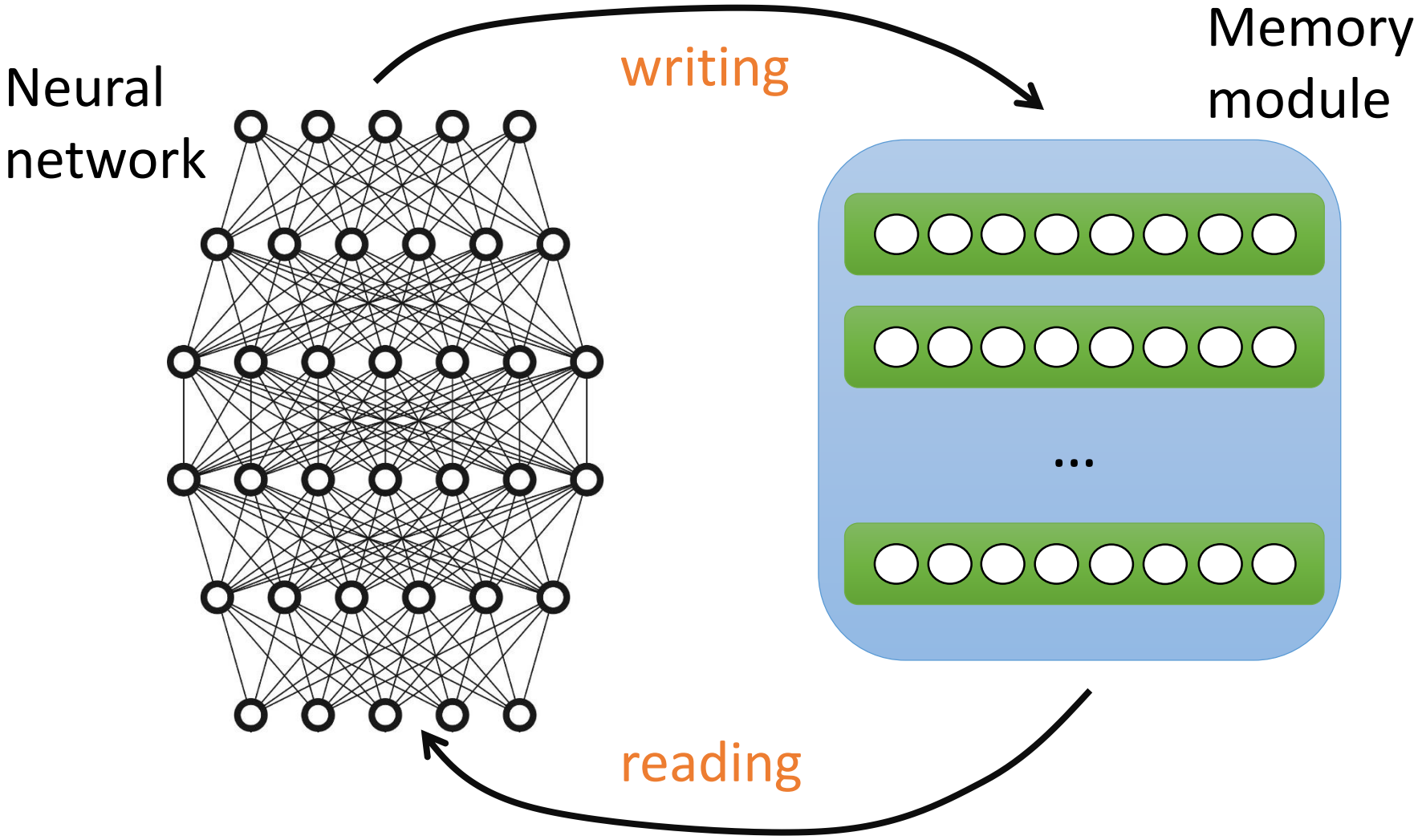
John picked up the apple.
John went to the office.
John went to the kitchen.
John dropped the apple.
Where was the apple before the kitchen? A:office

Task 15: Basic Deduction

Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of? A:wolves

- In order to solve the task need to memorize
  - Facts
  - Question
  - Answers
- A bit too much for the dynamical RNN memory
- Try to address it through an external memory

Memory Networks - General Idea



Memory Network Components

(I) **Input feature map**: Encodes the input in a feature vector

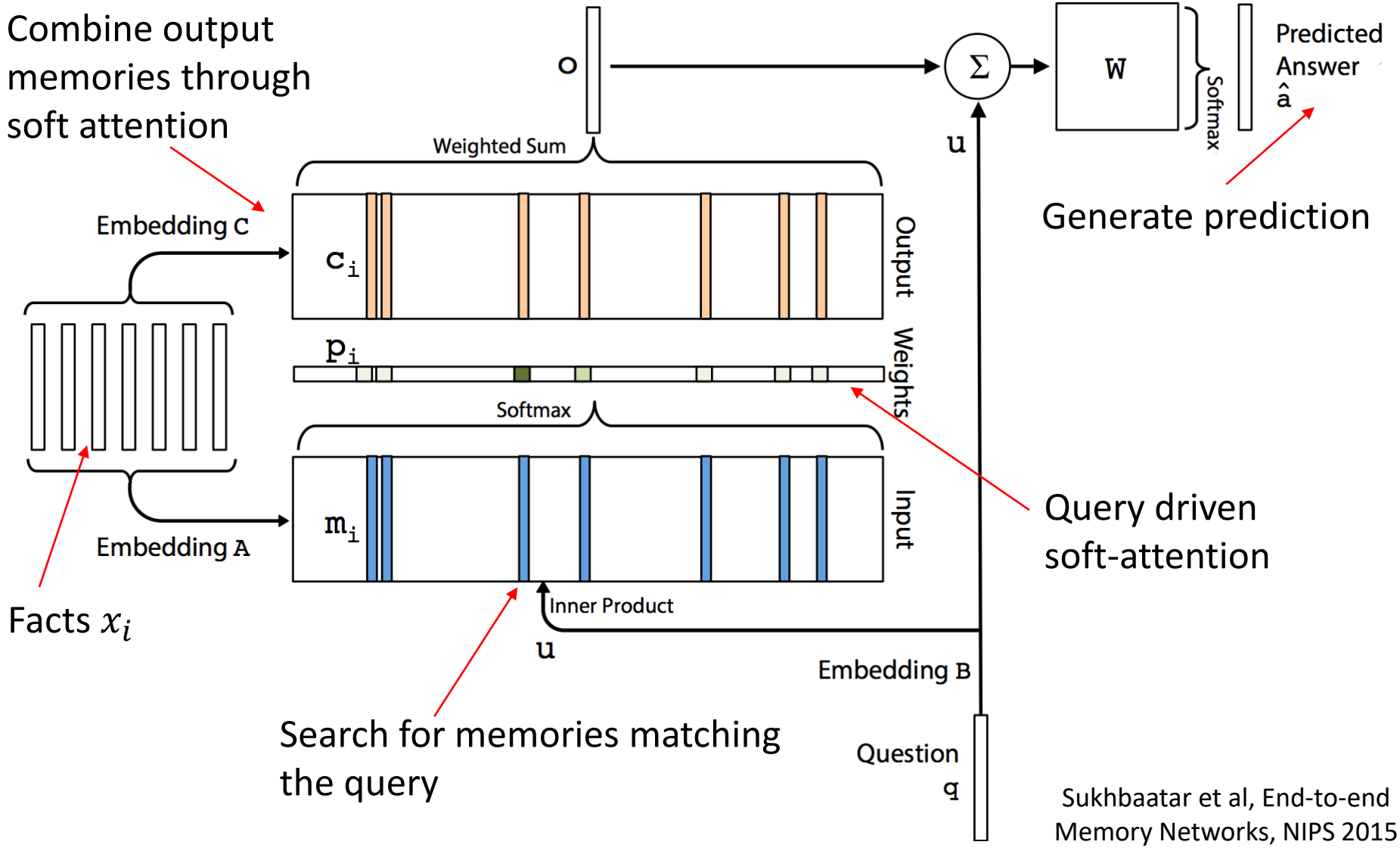
(G) **Generalization**: decide what input (or function of it) to write to memory

(O) **Output feature map**: reads the relevant memory slots

(R) **Response**: returns the prediction given the retrieved memories

Weston et al, Memory Networks, ICLR 2015

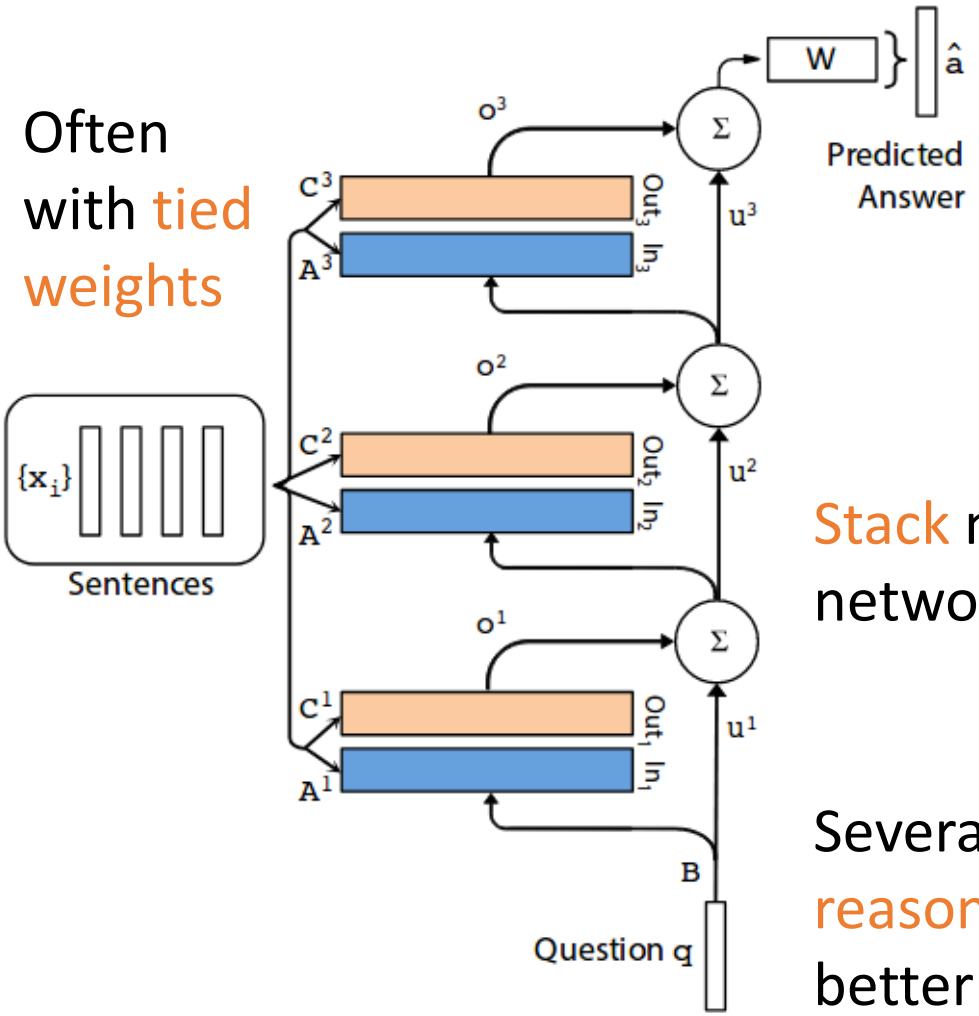
End-to-End Memory Networks



Sukhbaatar et al, End-to-end Memory Networks, NIPS 2015

Memory Network Extensions

Often with tied weights

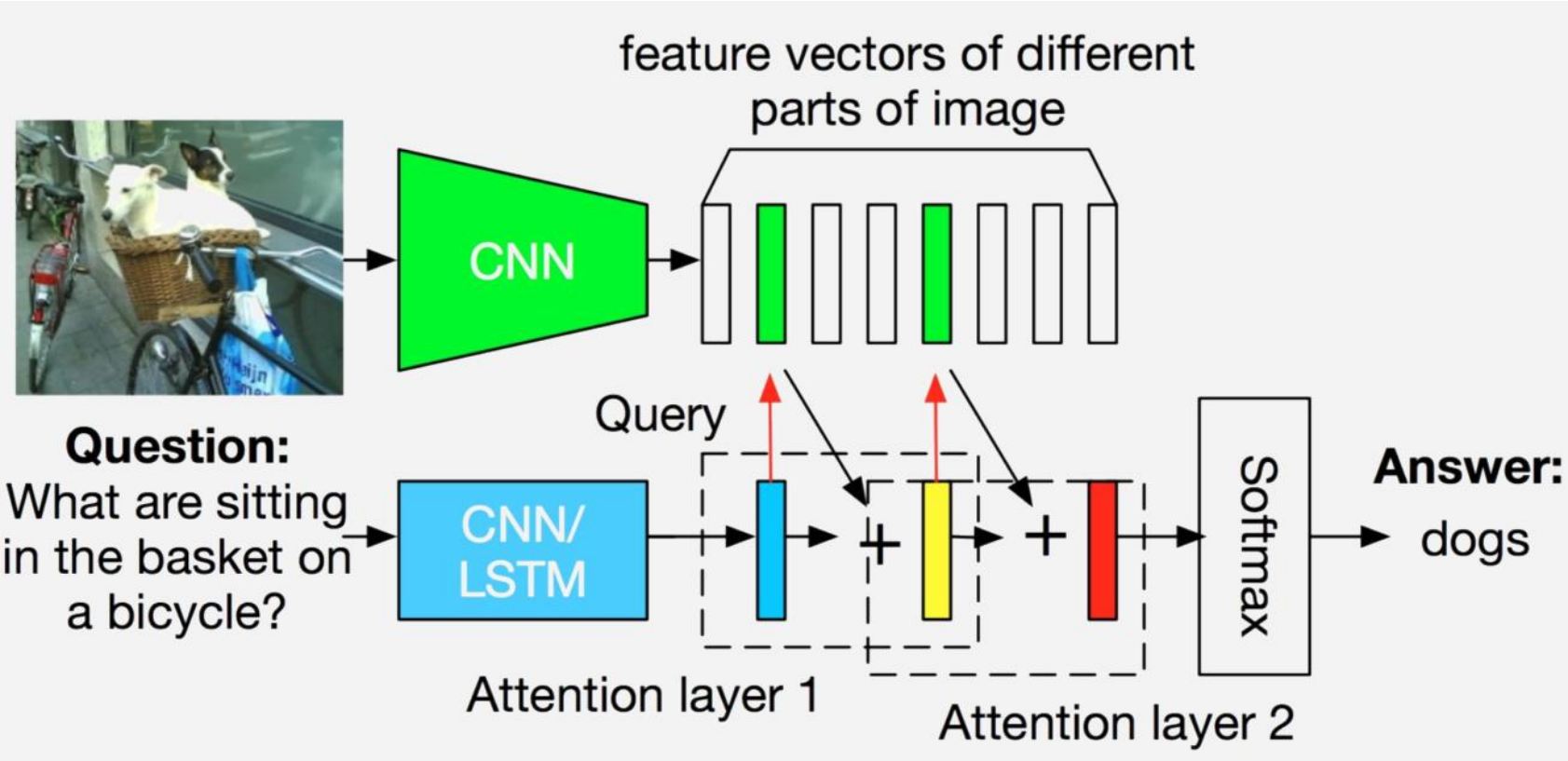


Use more complex output components, e.g. RNN to generate response sequences

Stack multiple memory network layers

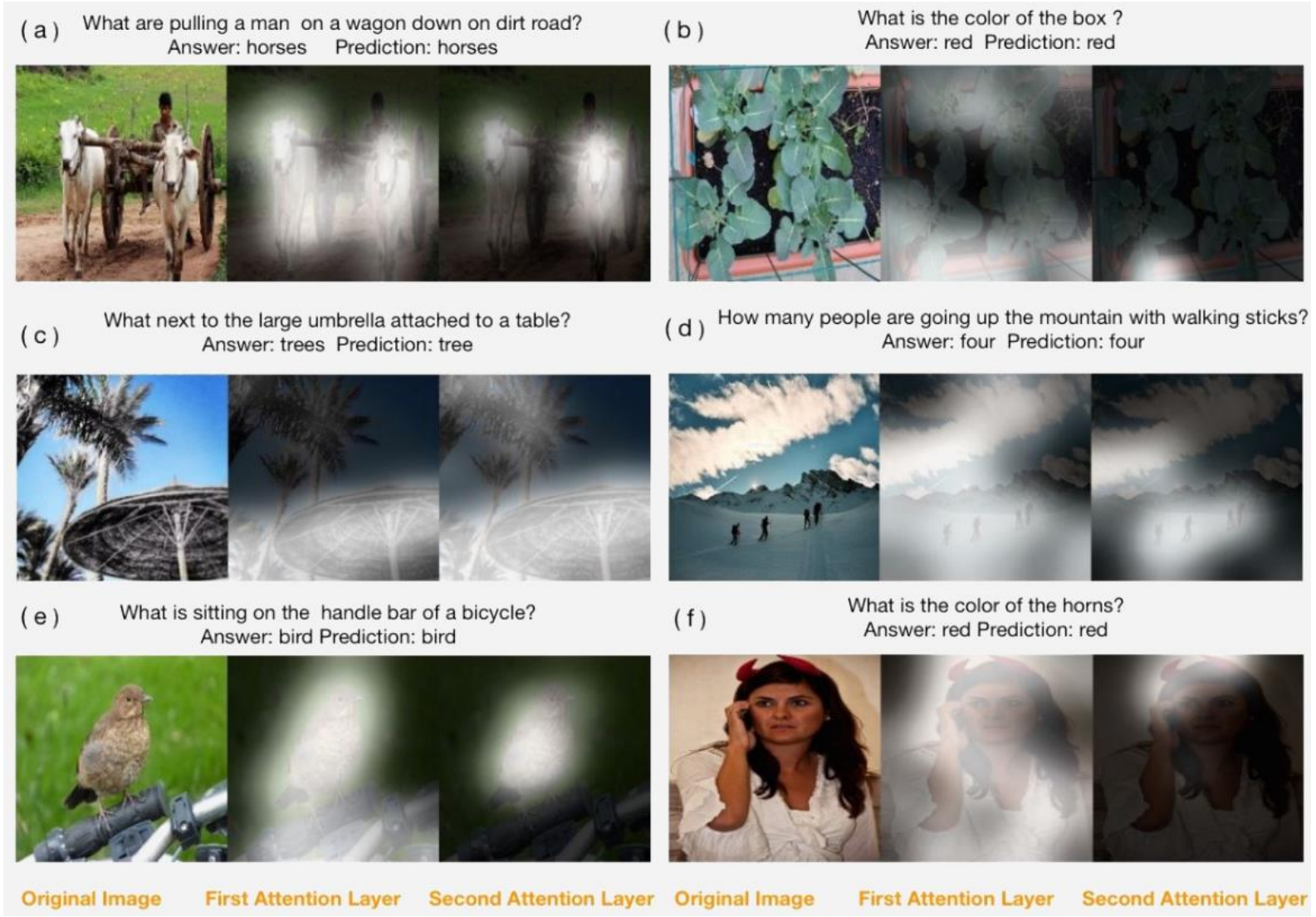
Several iterations of reasoning to produce a better answer

Memory Nets for Visual QA with Attention

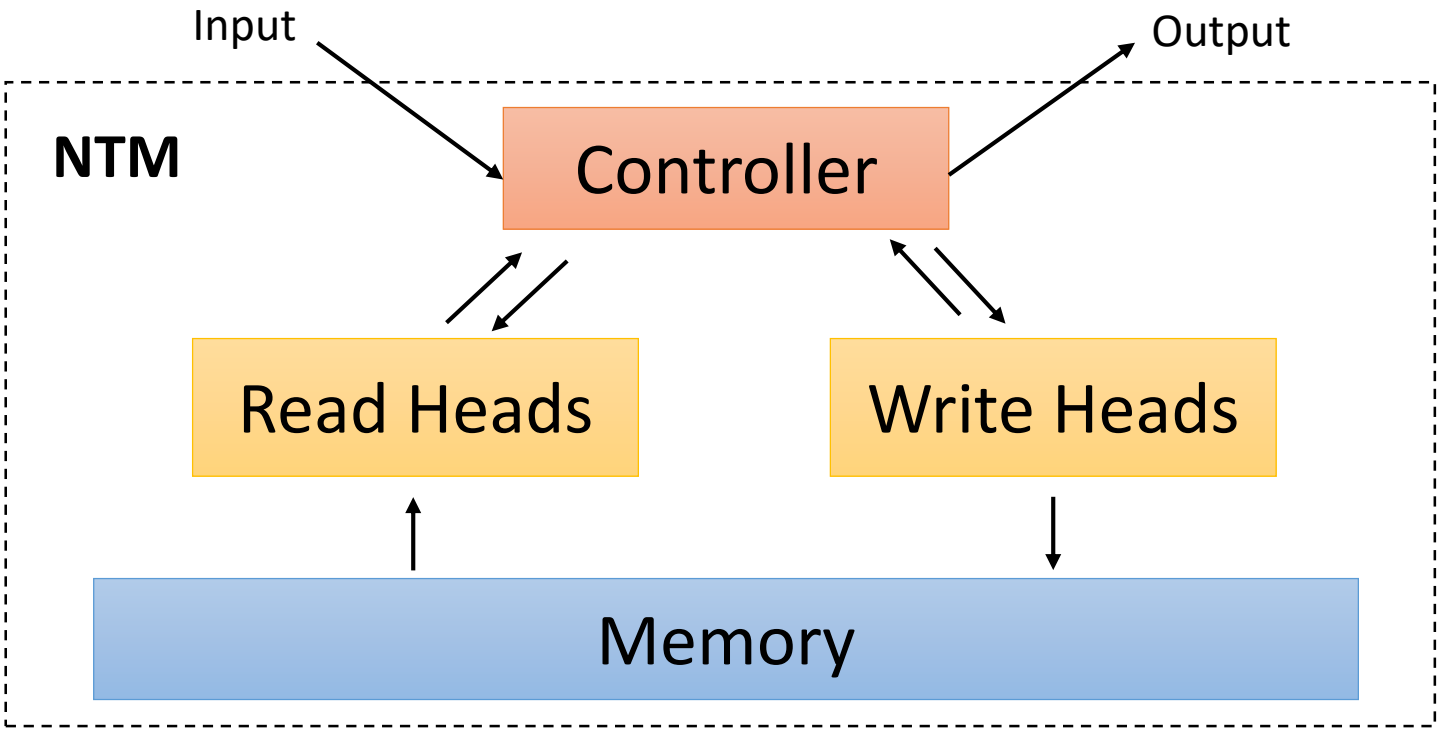


Yang et al, Stacked Attention Networks for Image Question Answering, CVPR 2016

Memory Nets for Visual QA with Attention



Neural Turing Machines

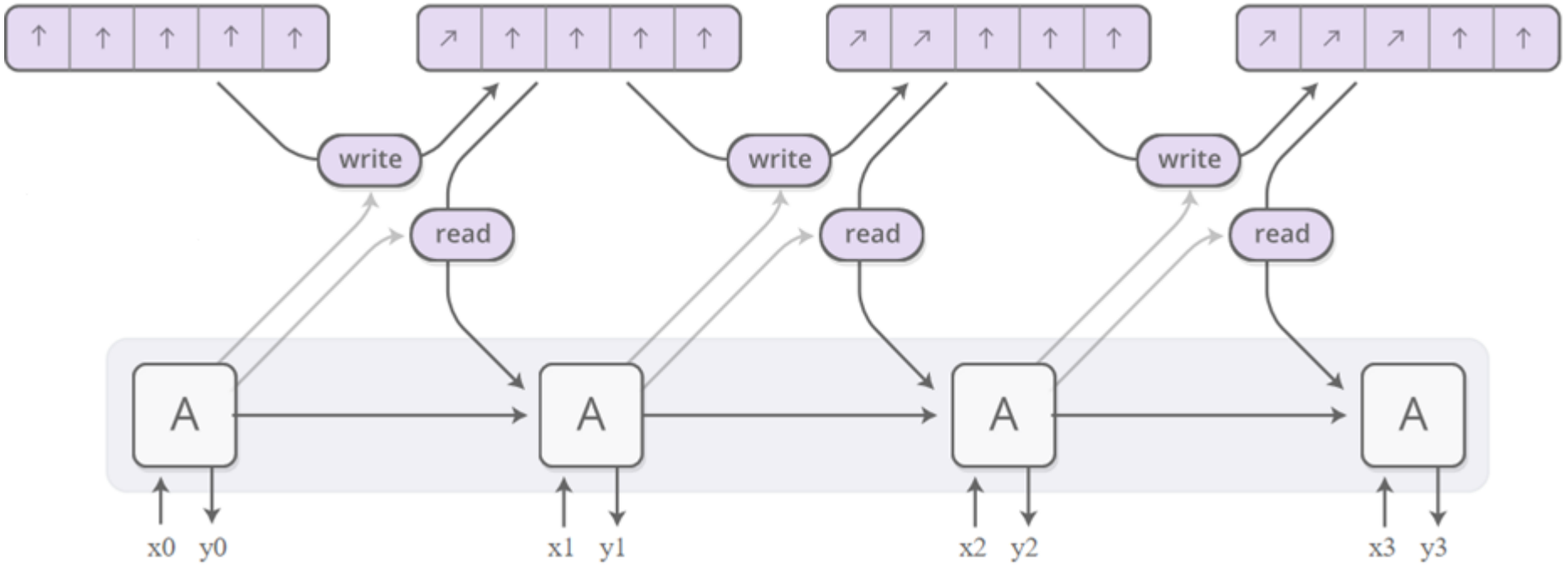


- Memory networks that can read and write memories at both training and test
- End-to-end differentiable

Neural Controller

Image credits @ colah.github.io

Typically an RNN emitting vectors to control read and write from the memory

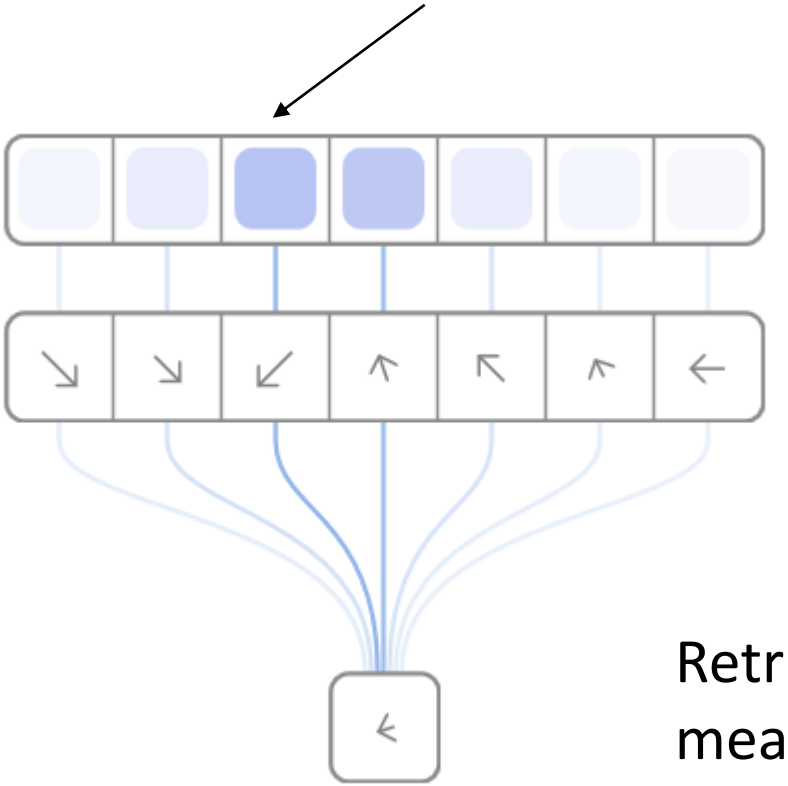


The key to differentiability is to always read and write the whole memory

Use soft-attention to determine how much to read/write from each point

Memory Read

Interest in the i-th memory



Attention distribution vector $\boldsymbol{a}$ from the RNN

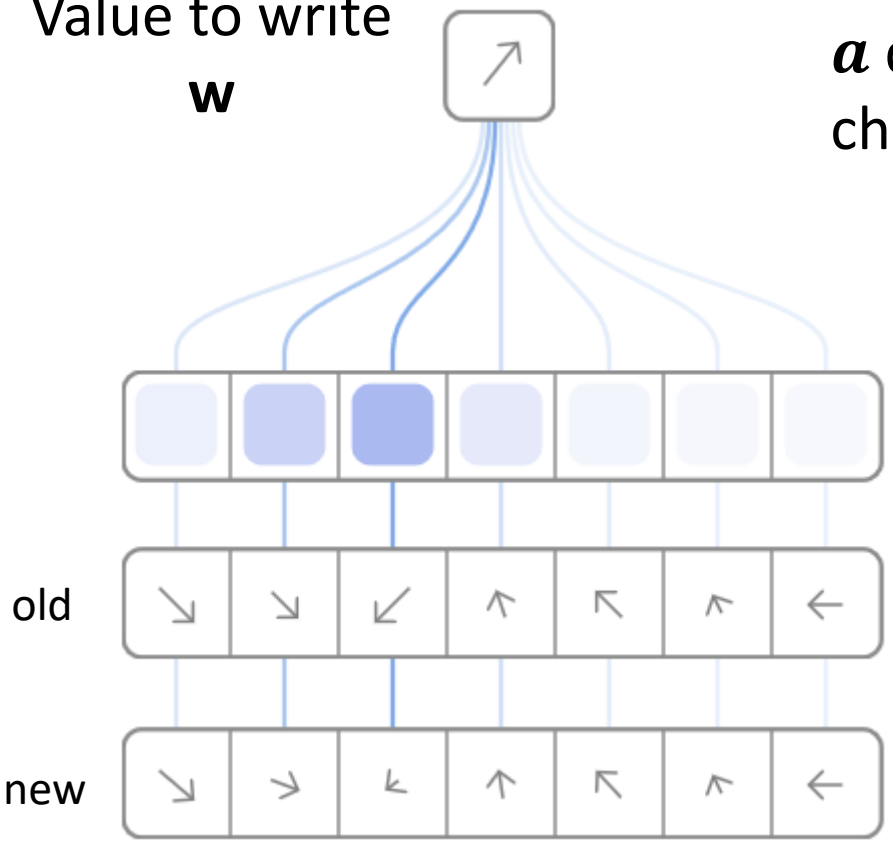
Memories $\boldsymbol{M}_i$

Retrieved memory is a weighted mean of all memories

$$\boldsymbol{r} = \sum_i a_i \boldsymbol{M}_i$$

Memory Write

Value to write
w



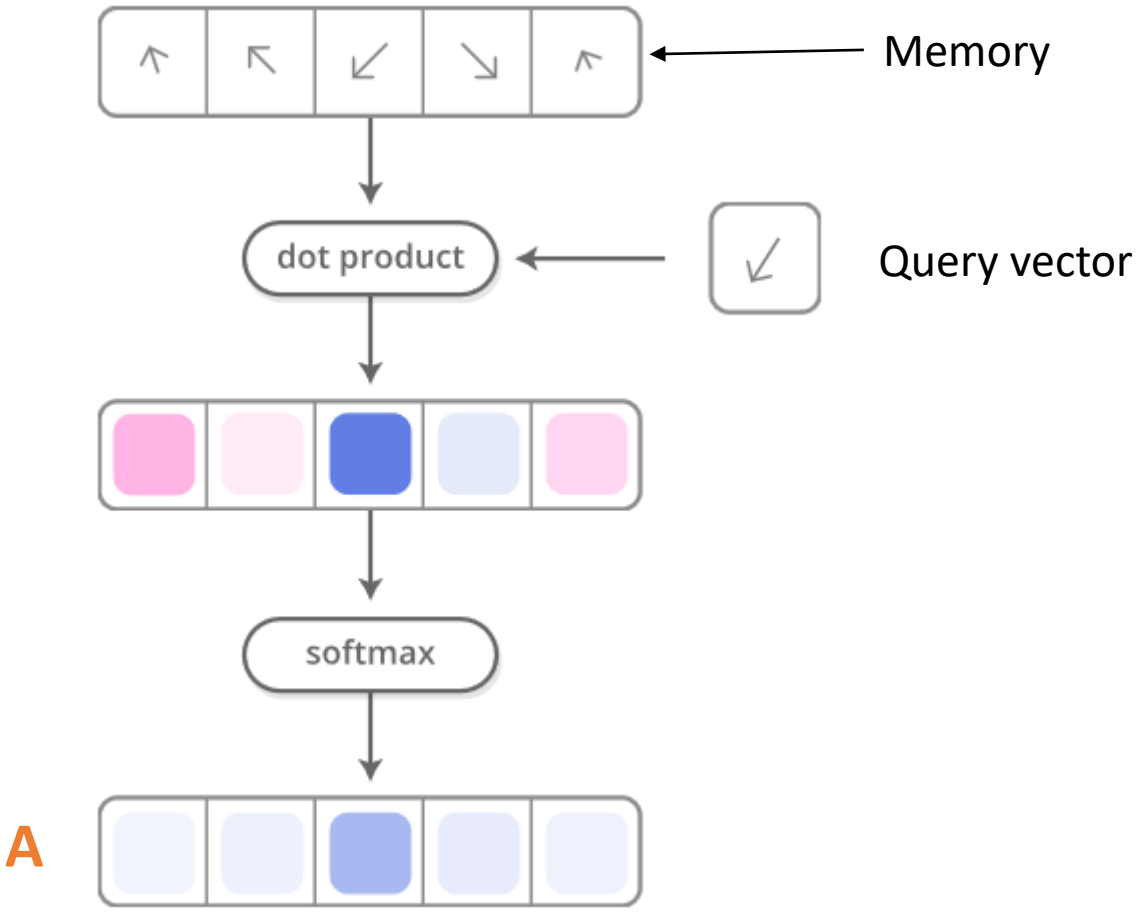
Attention distribution vector $\boldsymbol{a}$ describing how much we change each memory

$$\boldsymbol{M}_i = \mathrm{a_i}\mathbf{w} + (1 - \mathrm{a_i})\mathbf{M_i}$$

Write operation is actually performed by composing erasing and adding operations

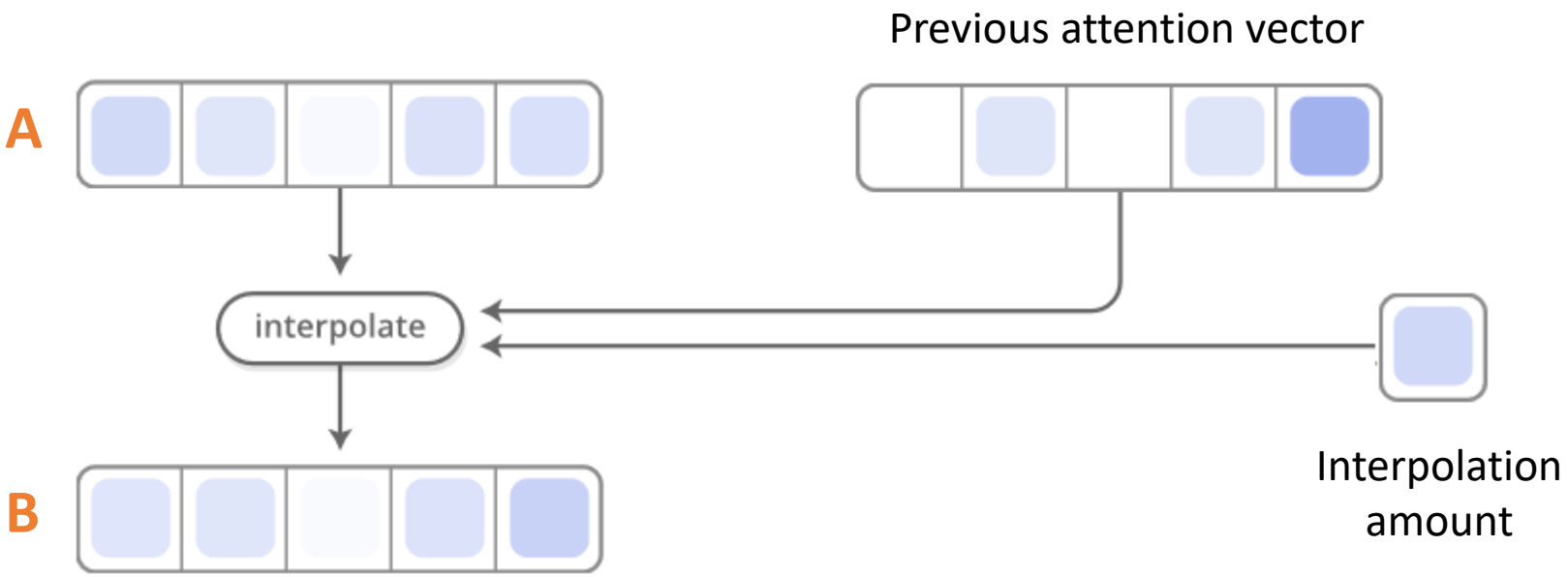
NTM Attention Focusing

1. Generate content-based memory indexing



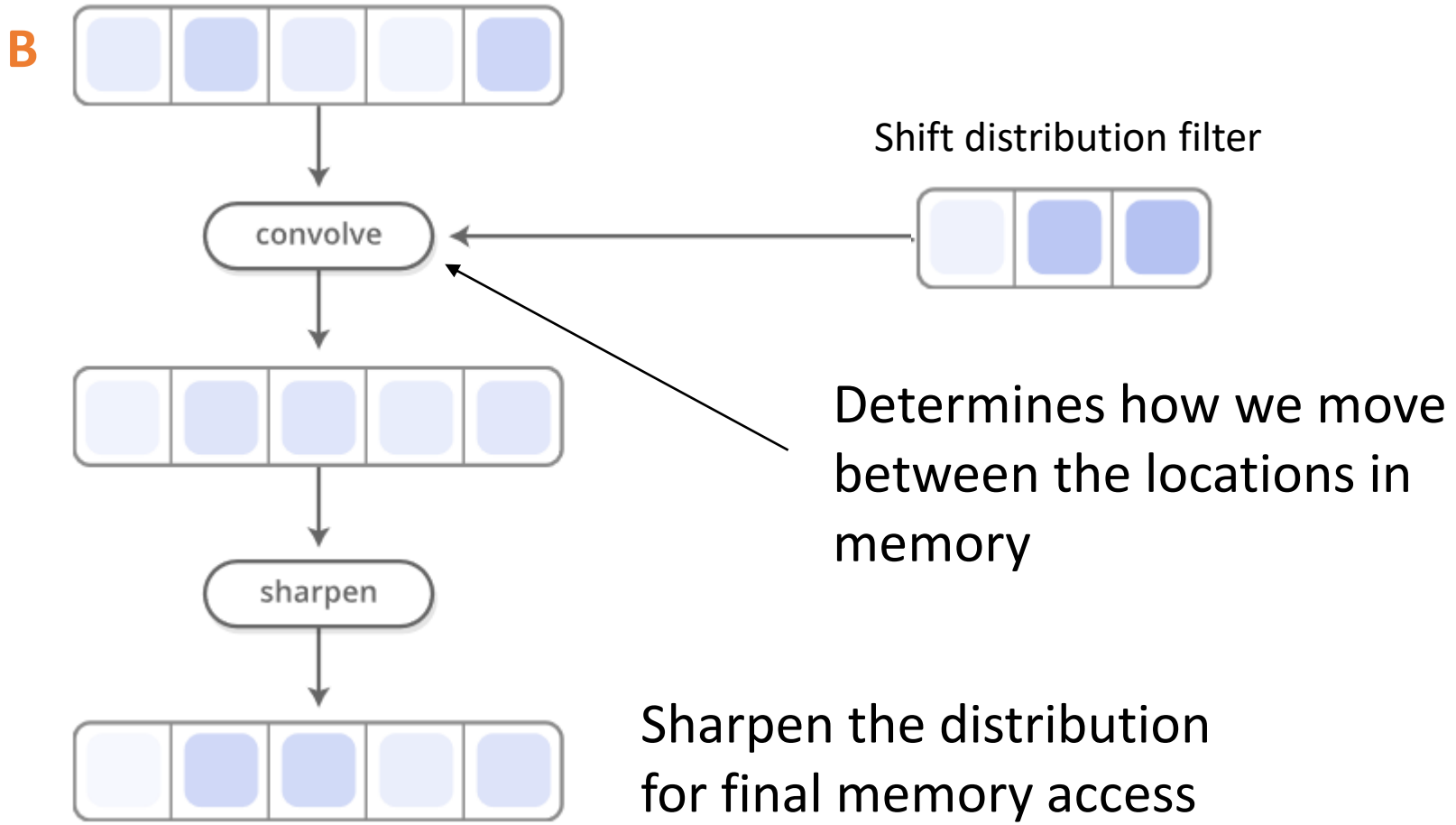
NTM Attention Focusing

2. Interpolate with attention from previous time



NTM Attention Focusing

3. Generate location-based indexing



Practical Use?

- Still missing the killer application
- Not straightforward to train
- Superior to GRNN when it comes to learn to program
  - Copy task
  - Repeat copy
  - Associative recall
  - Sorting
- Foundations for neural reasoning
  - Pondering networks

Software

- Complete sequence-to-sequence tutorial (including attention) on Tensorflow
  - A shorter version in Keras
- Github project collecting several memory augmented networks
- Pytorch implementation of stacked attention for visual QA (originally Theano-based)
- Many implementations of the NTM (Keras, Pytorch, Lasagne,…): none seemingly official, but this recent one is supposedly stable

Take Home Messages

- Attention.. Attention.. and, again, attention
  - Soft attention is nice because makes everything fully differentiable
  - Hard attention is stochastic hence cannot Backprop
  - Empirical evidences of them being sensitive to different things
- Encoder-Decoder scheme
  - A general architecture to compose heterogeneous models and data
  - Decoding allows sampling complex predictions from an encoding conditioned distribution
- Memory and RNN
  - Efficient use of dynamic memory
  - External memory for search and recall tasks
  - Read/write memory for neural reasoning