Neural Memories

INTELLIGENT SYSTEMS FOR PATTERN RECOGNITION (ISPR)

DAVIDE BACCIU – DIPARTIMENTO DI INFORMATICA - UNIVERSITA’ DI PISA

DAVIDE.BACCIU@UNIPI.IT

Lecture Outline

- Dealing with very long-term dependencies
  - Multiscale networks
  - Adding memory components
- Neural reasoners
  - Neural memories
  - Differentiable memory read, write, indexing

UNIVERSITÀ DI PISA

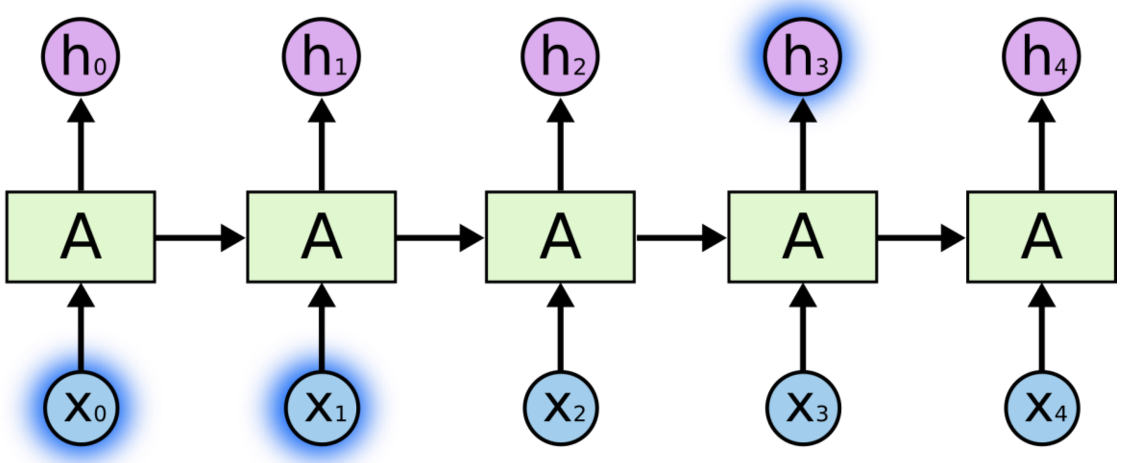
Hierarchical and Multiscale Recurrent Networks

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

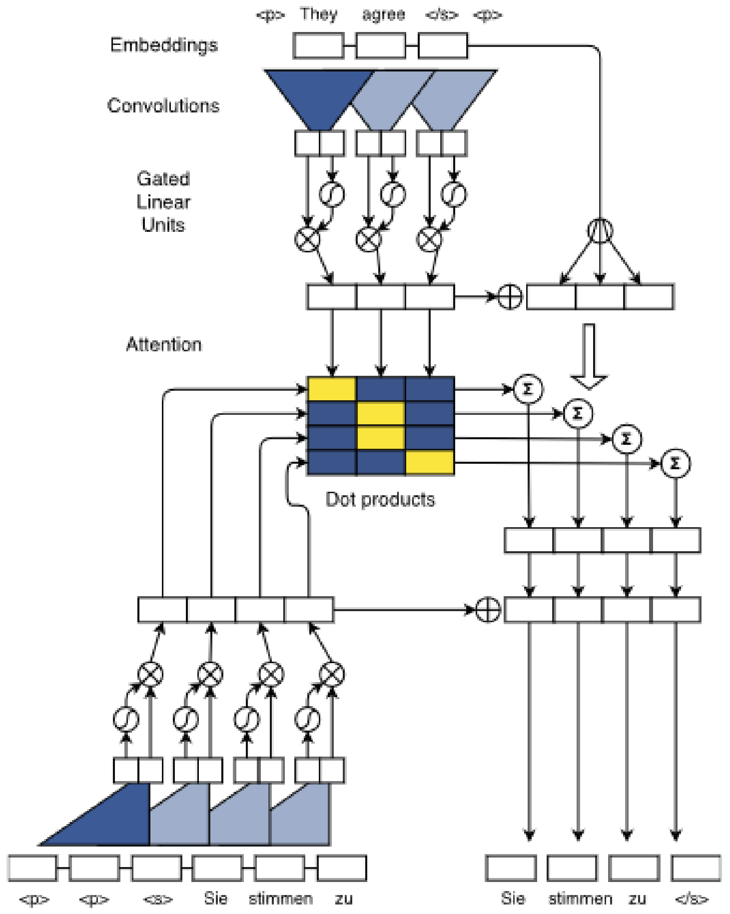
Skipping State Updates Mitigates Vanishing Gradients

- Vanishing effects are exponential w.r.t. the unrolled network’s depth.
- Skipping updates entirely reduces the vanishing effect.
- How can we shorten the path between long-range dependencies?
- How can we skip updates? Any idea/intuitions?



Convolutional Seq2Seq

- Use convolution instead of recurrence
- Better parallelization on GPUs
- Smaller distance between long-range dependencies



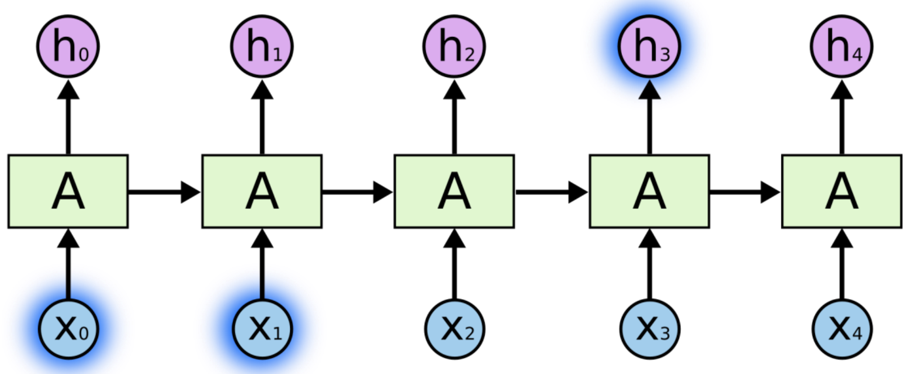
Depth of different architectures

- Vanishing gradients depend on the depth of the network

Layer Type	Complexity per Layer	Sequential Operations	Maximum Path Length
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

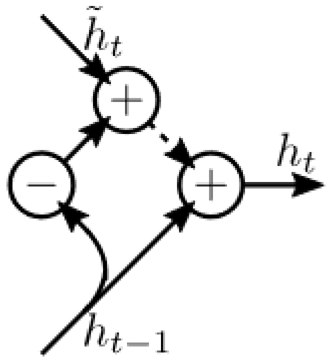
Hierarchical RNNs

- Add skip connections to the model (static skip)
- Learn when to skip updates (adaptive skip)
- Skip units, blocks of units, or entire layer



Regularization – Zoneout

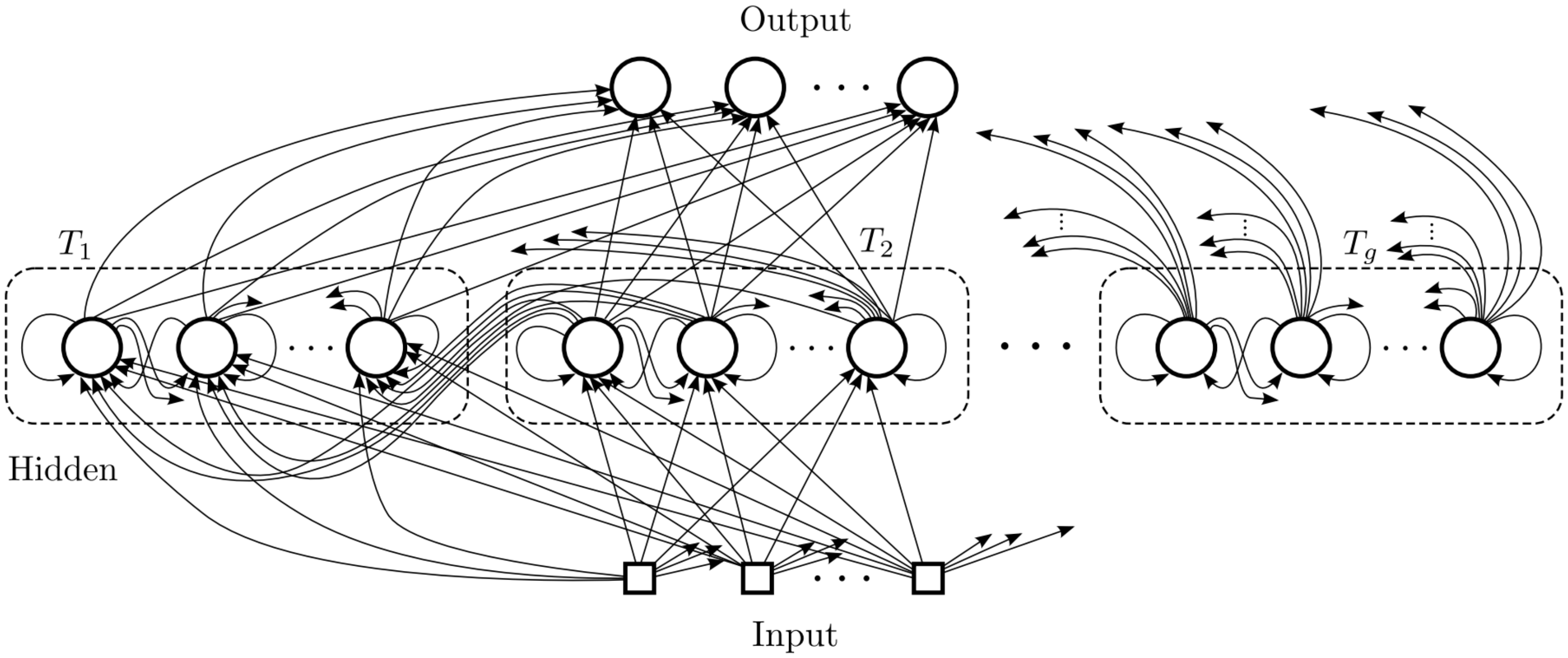
- At each timestep, force some units to keep the same value
- Random sampling
- Hard gate
- Avoiding the update improves gradient propagation



Zoneout: $\mathcal{T} = d_t \odot \tilde{\mathcal{T}} + (1 - d_t) \odot 1$ **Dropout:** $\mathcal{T} = d_t \odot \tilde{\mathcal{T}} + (1 - d_t) \odot 0$

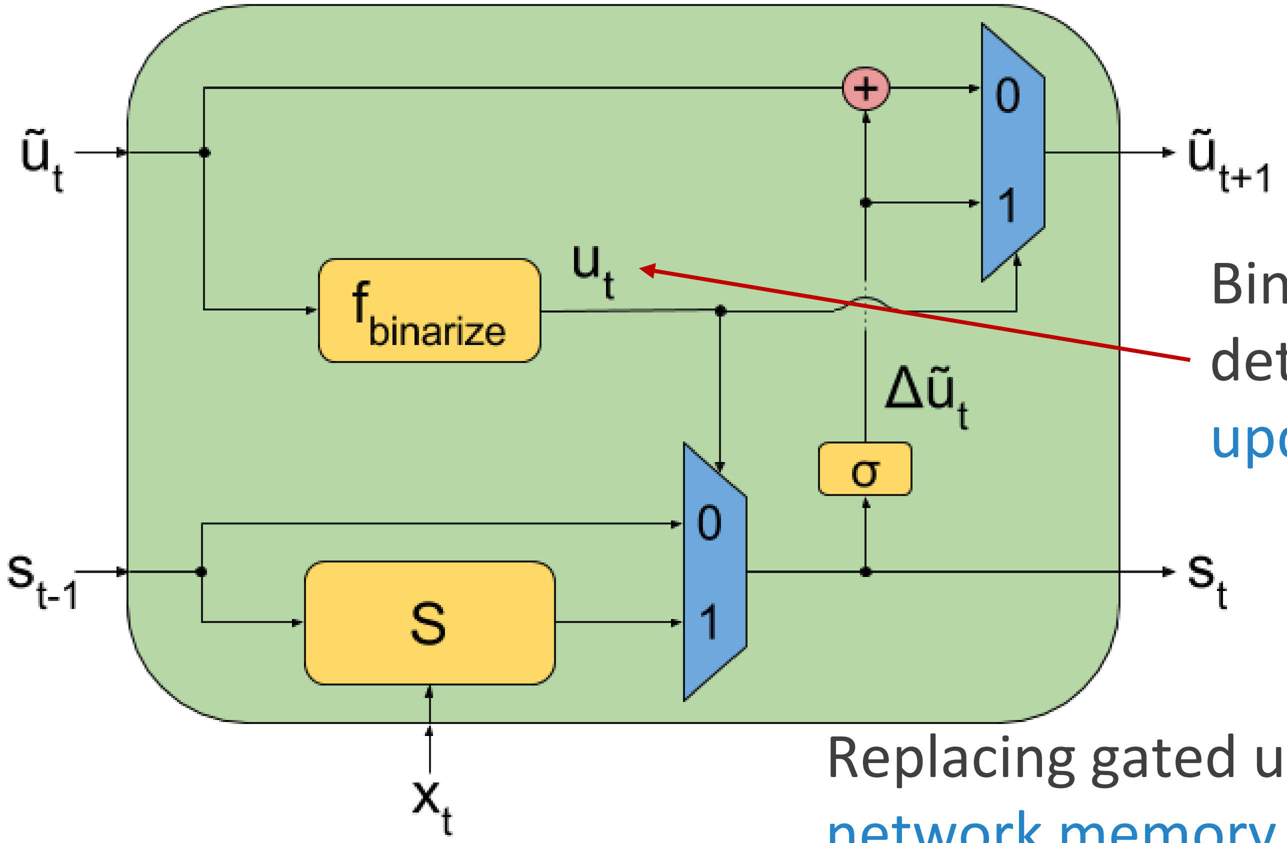
Clockwork RNN

Modular recurrent layer where each module is updated at different clock



Modules interconnected only when destination clock time is larger

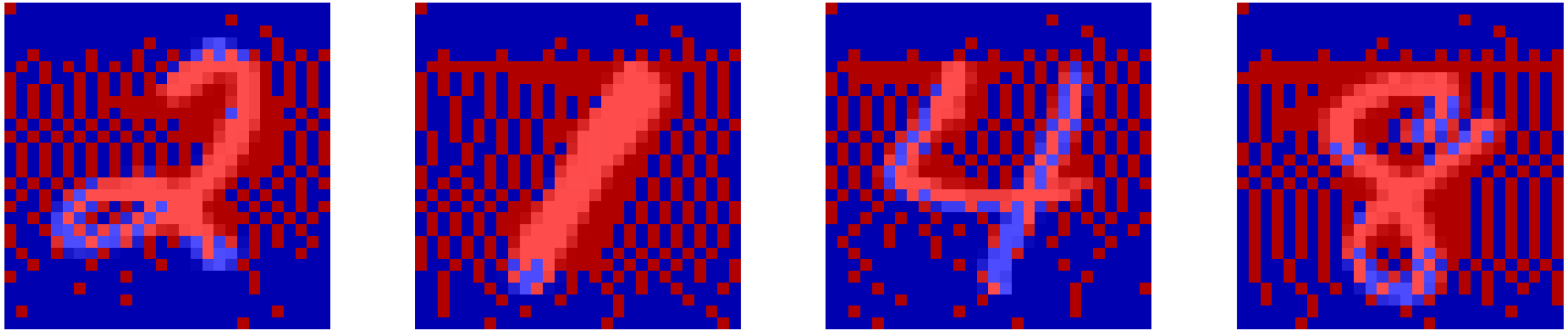
Skip RNN



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

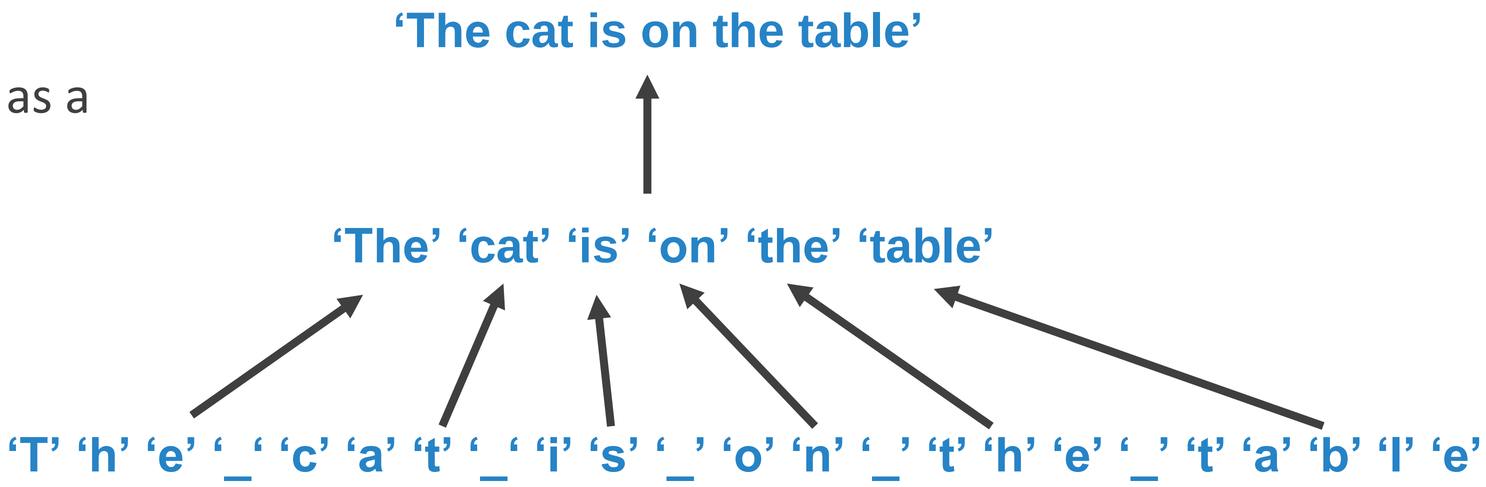
Ignored pixels

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

UNIVERSITÀ DI PISA

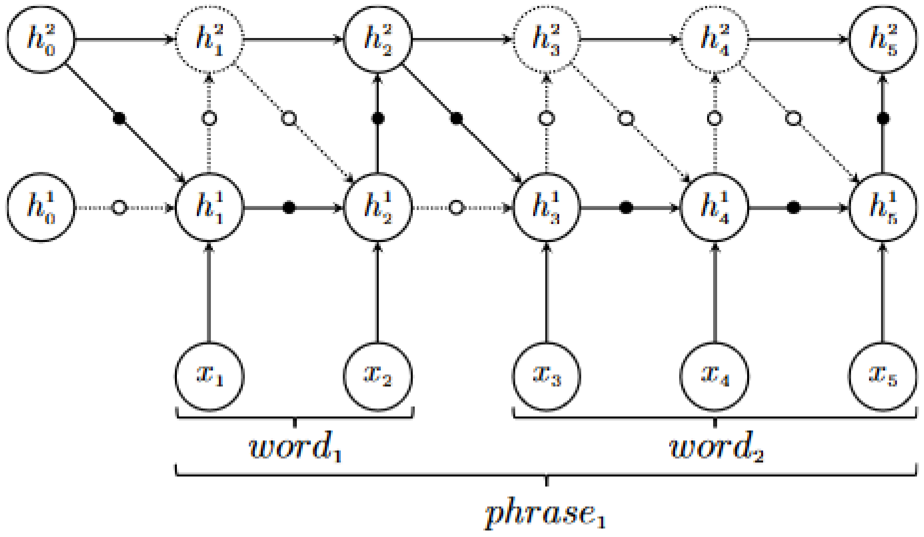
Hierarchical Sequential Structure

- Many sequences have a latent hierchical structure
- Example: wikipedia, represented as a sequence of characters
- Hierarchy
  - Characters
  - Words
  - Sentences
  - Paragraphs
  - Documents
- We want to model the hierarchy explicitly



Hierarchical Multiscale RNN (HM-RNN)

- ***UPDATE***: state update (LSTM cells) according to boundary detector.
- ***COPY***: copies cell and hidden states from the previous timestep to the current timestep.
- ***FLUSH***: sends summary to next layer and re-initializes current layer's state.



Junyoung Chung, HIERARCHICAL MULTISCALE RECURRENT NEURAL NETWORKS, https://arxiv.org/pdf/1609.01704.pdf

Recap (1)

- **Recurrence**: update at each timestep, linear scan of the sequence. Path length=n
- **Convolution**: update at each timestep but look at the last k timestep. Path length=log_k(n)
- **Attention**: update the entire sequence in parallel. Path length=1

Recap (2)

- **Zoneout**: randomly disable unit update.
- **CW-RNN**: blocks of units with different update frequencies. Static.
- **Skip-RNN**: adaptive gates learns to skip entire updates. Also saves computation
- **HM-RNN**: each layer models more abstract features by learning the boundaries. Adaptive.

Neural Reasoning

Neural Reasoning

- Recurrent (sequential) models as general programs
- Reasoning abilities, typical of classic algorithms or classic AI
- Why don’t we use a «classic» algorithm? Because we may not have a proper input (e.g. pathfinding graph) but only sensor information.
    - Example: robot navigation
    - The model needs to learn to encode the structure from the raw data and then solve the problem

RNN and Memory – Issue 2

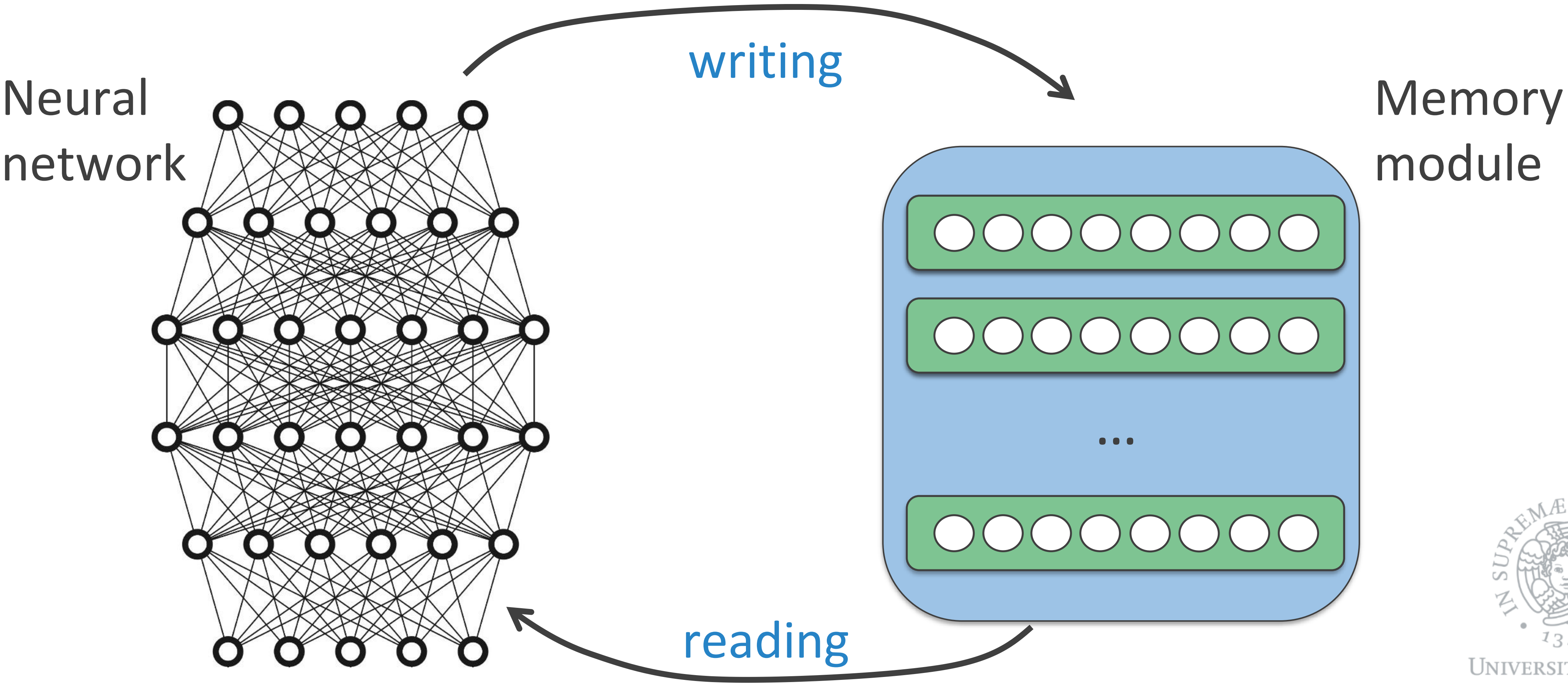
A motivating example:

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

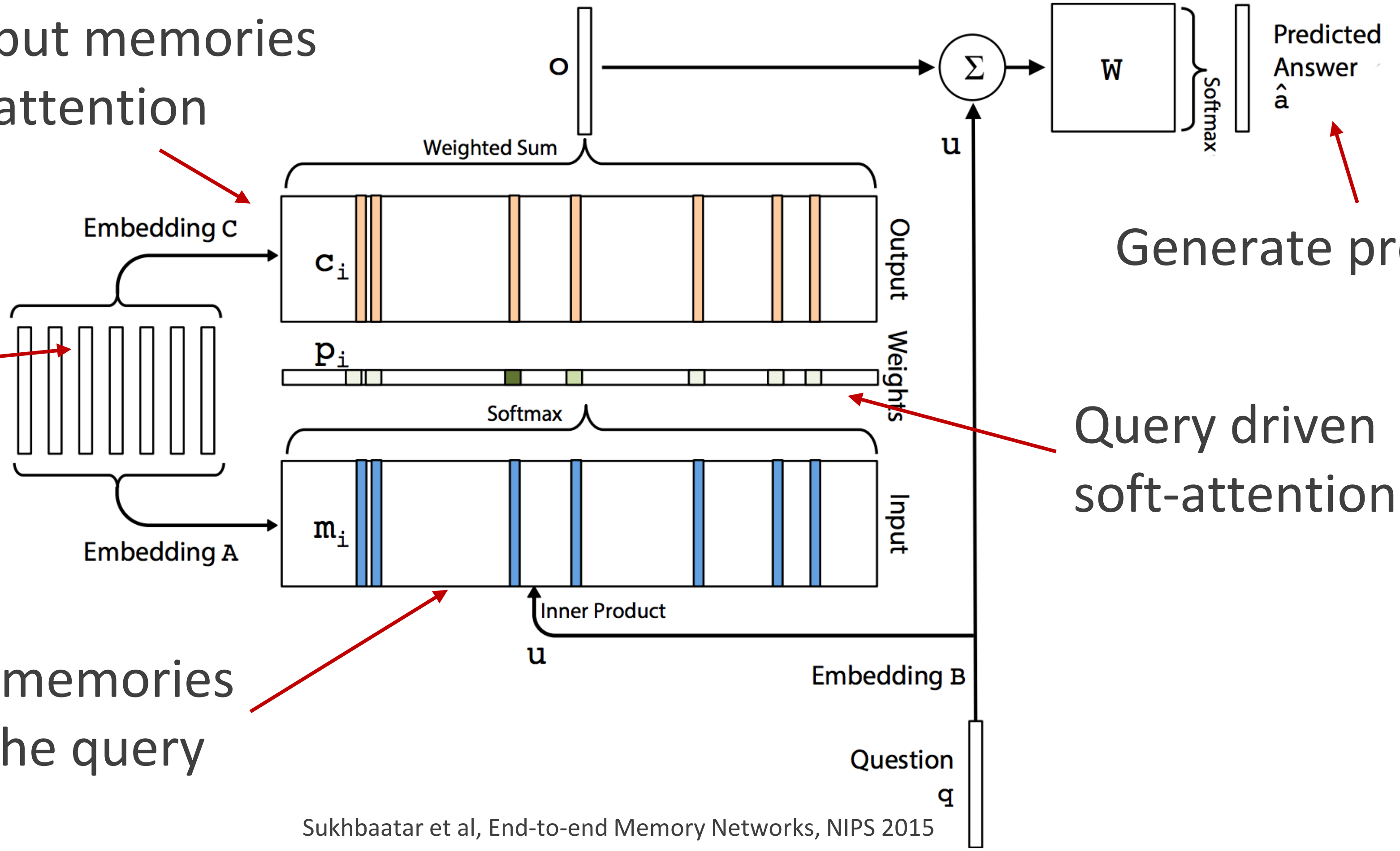
Weston et al, Memory Networks, ICLR 2015

End-to-End Memory Networks

Combine output memories through soft attention

Facts x_i

Search for memories matching the query



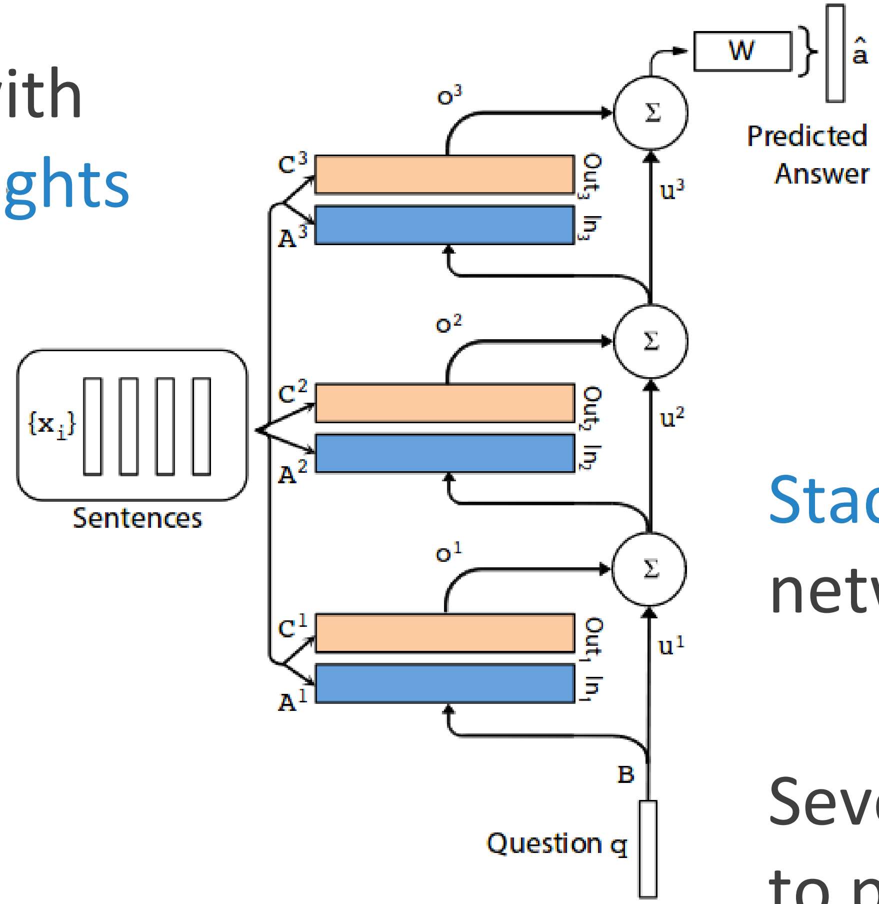
Generate prediction

Query driven soft-attention

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

Memory Network Extensions

Often with tied weights

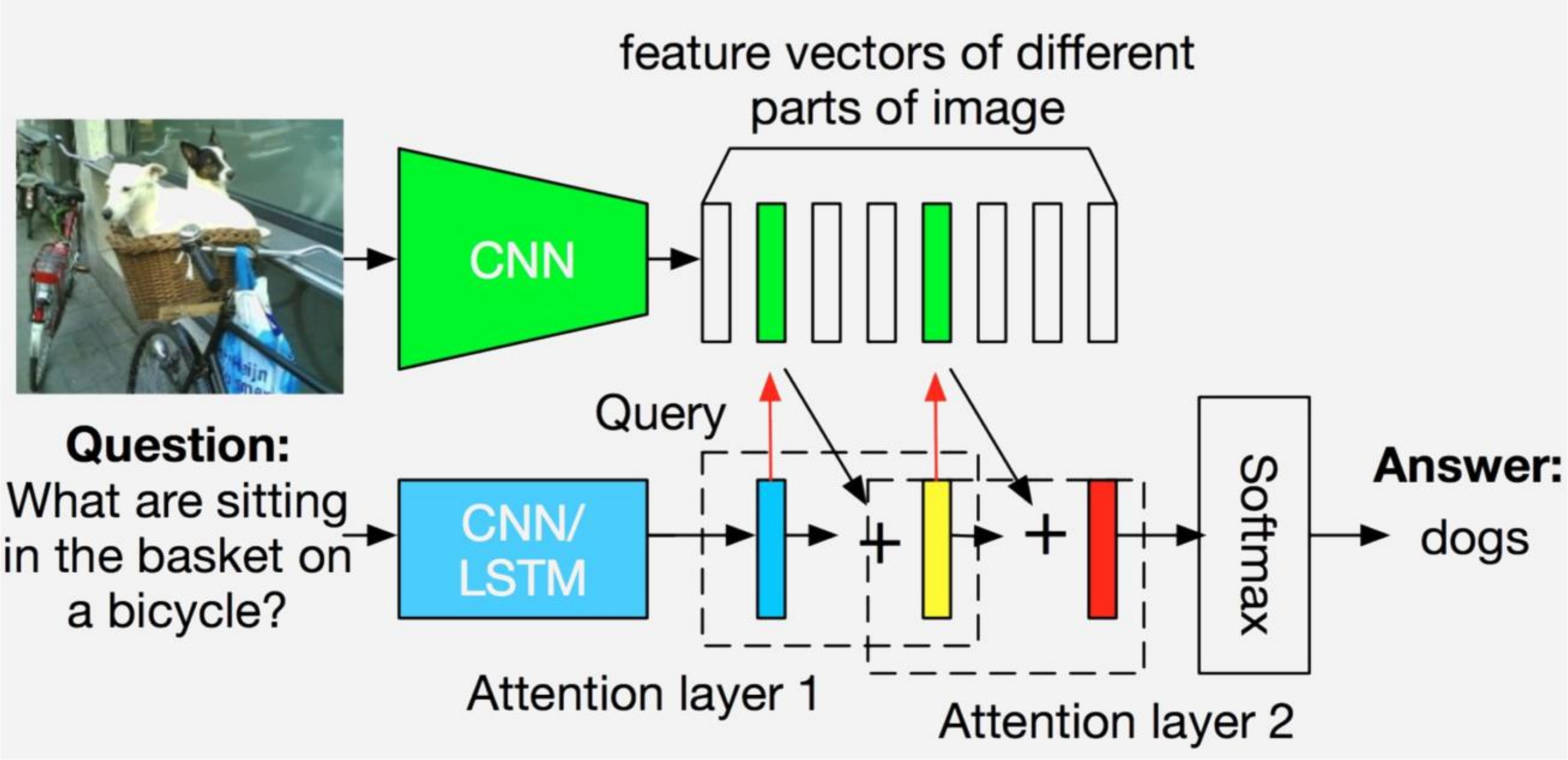


Use more complex output components, e.g. RNN to generate response sequences

Stack multiple memory network layers

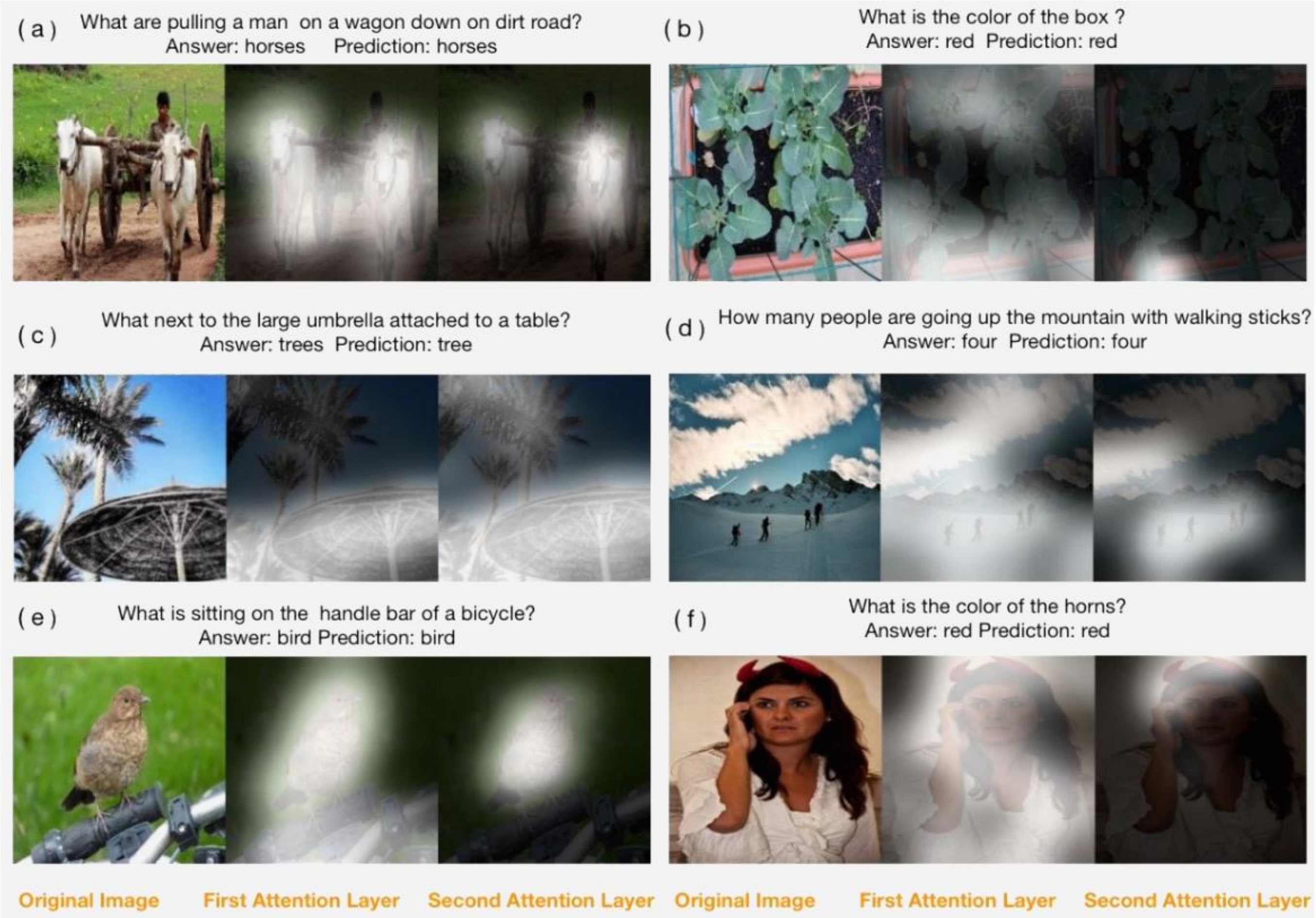
Several iterations of reasoning to produce a better answer

Memory Nets for Visual QA with Attention

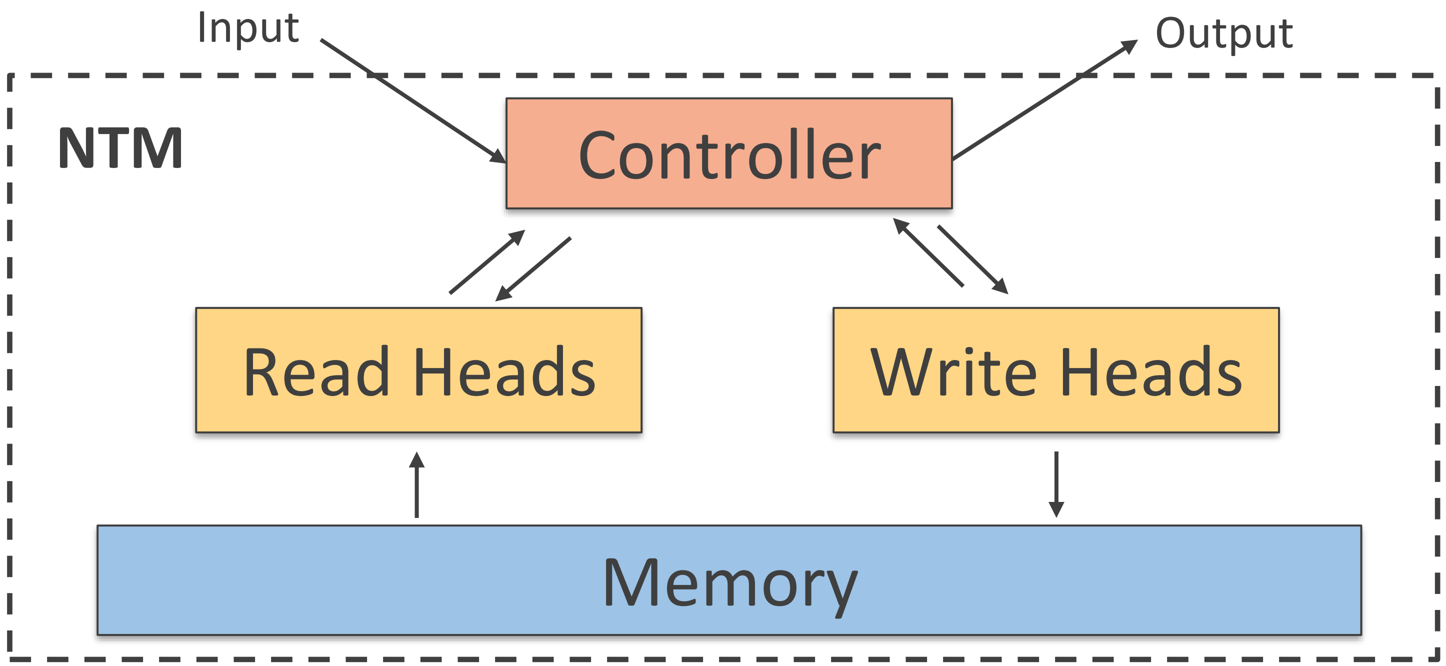


Yang et al, Stacked Attention Networks for Image Question Answering, CVPR 2016

Memory Nets for Visual QA with Attention



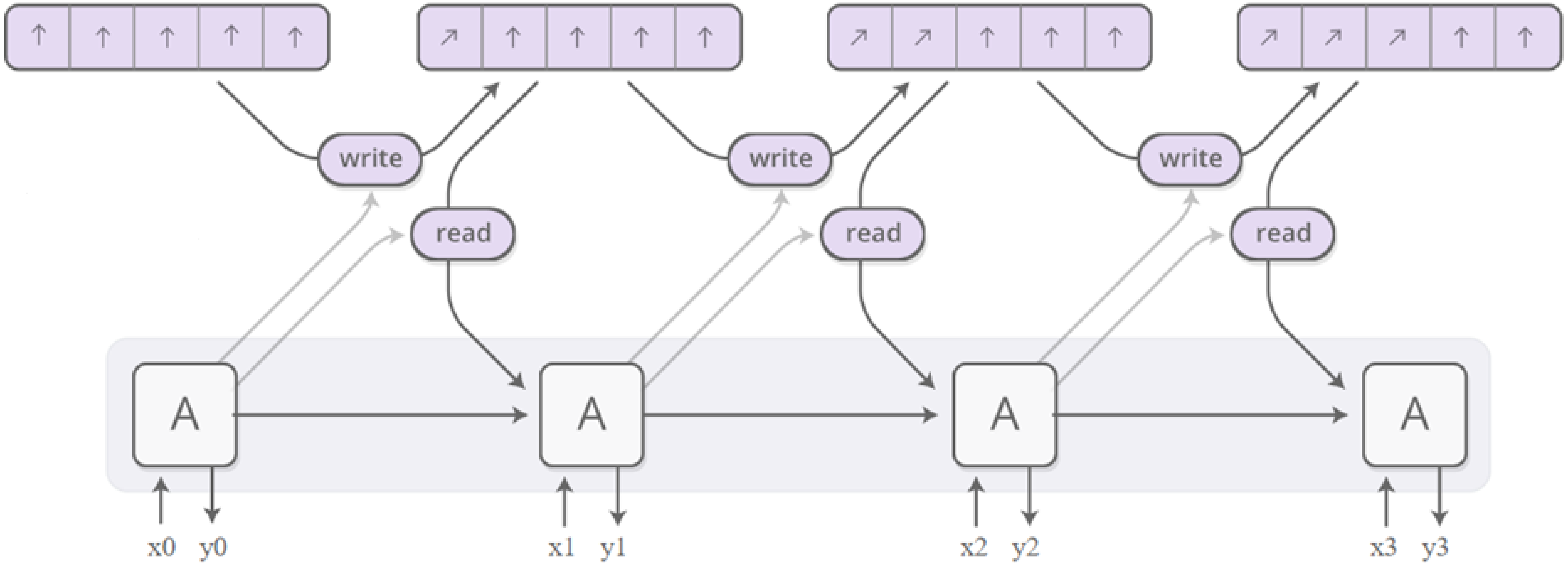
Neural Turing Machines



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

Neural Controller

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

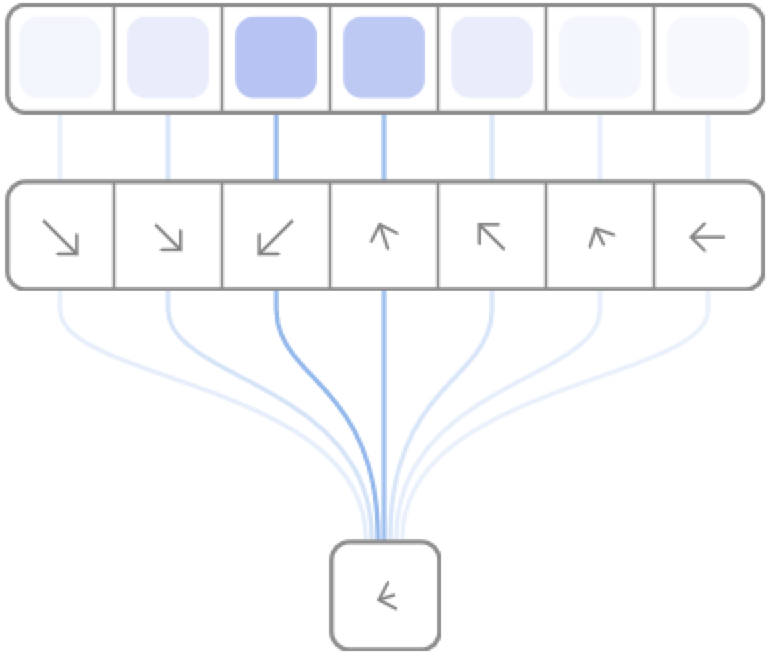
Use soft-attention to determine how much to read/write from each point

Memory Read

Interest in the i-th memory

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

Memories $\boldsymbol{M}_i$

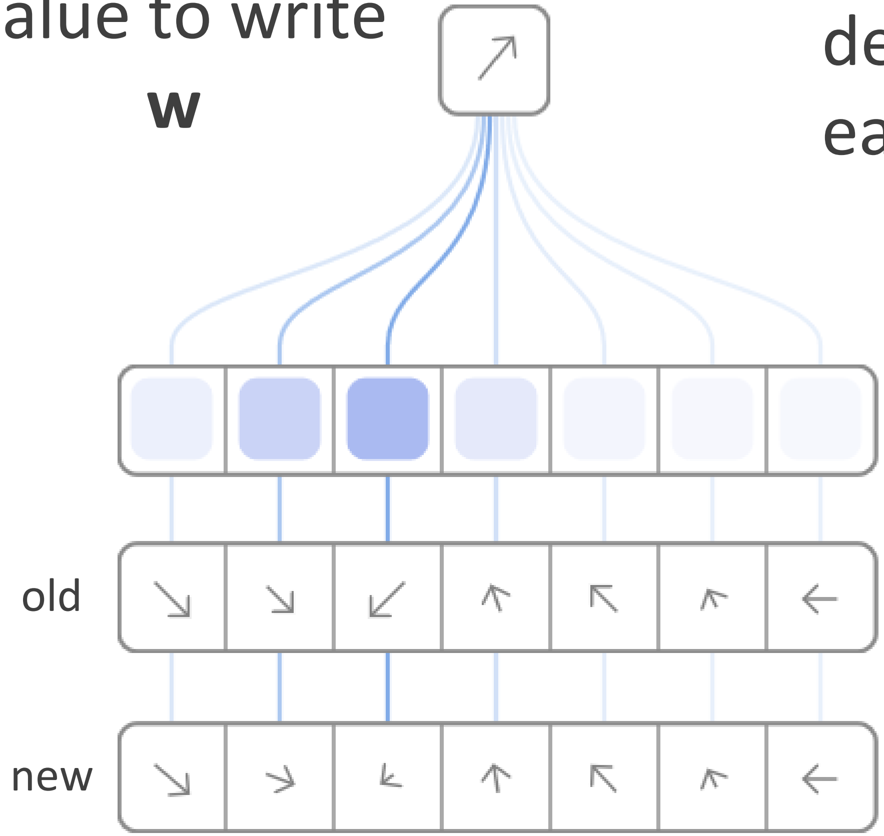


Retrieved memory is a weighted mean of all memories

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

Memory Write

Value to write
w



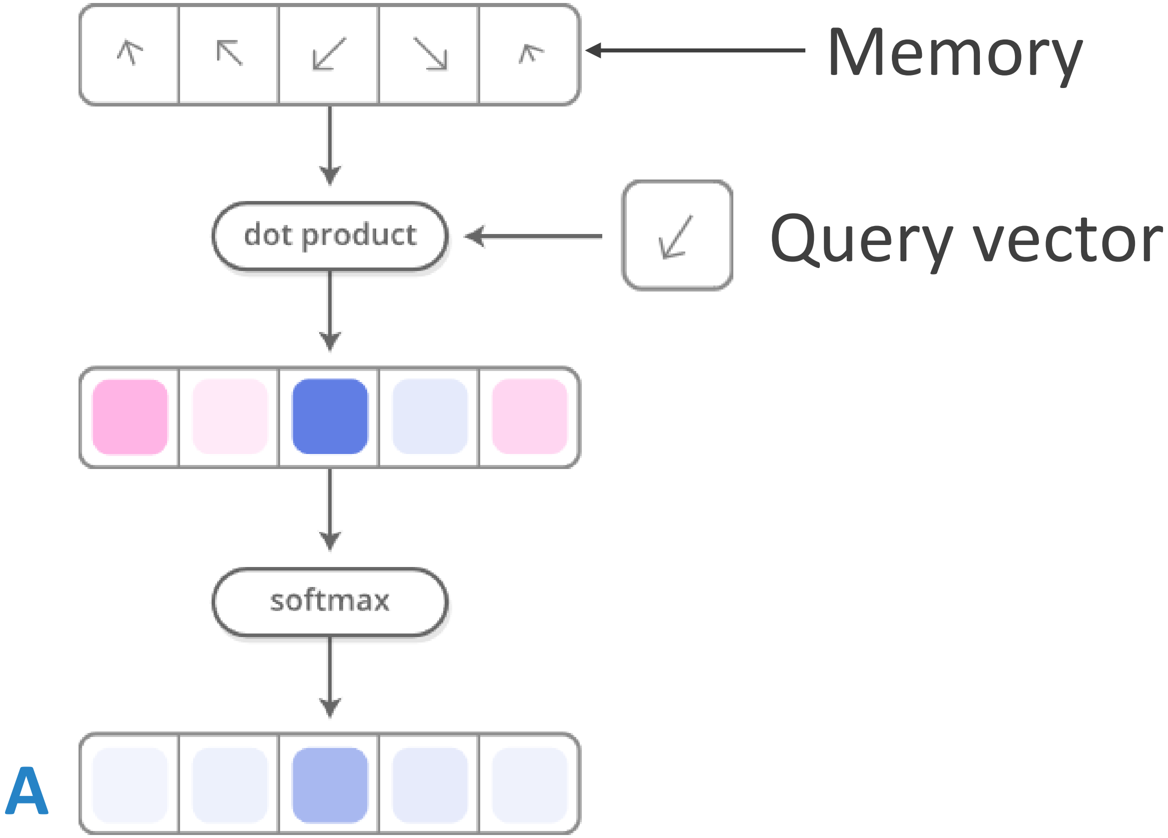
Attention distribution vector $\boldsymbol{a}$ describing how much we change each memory

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

Write operation is actually performed by composing erasing and adding operations

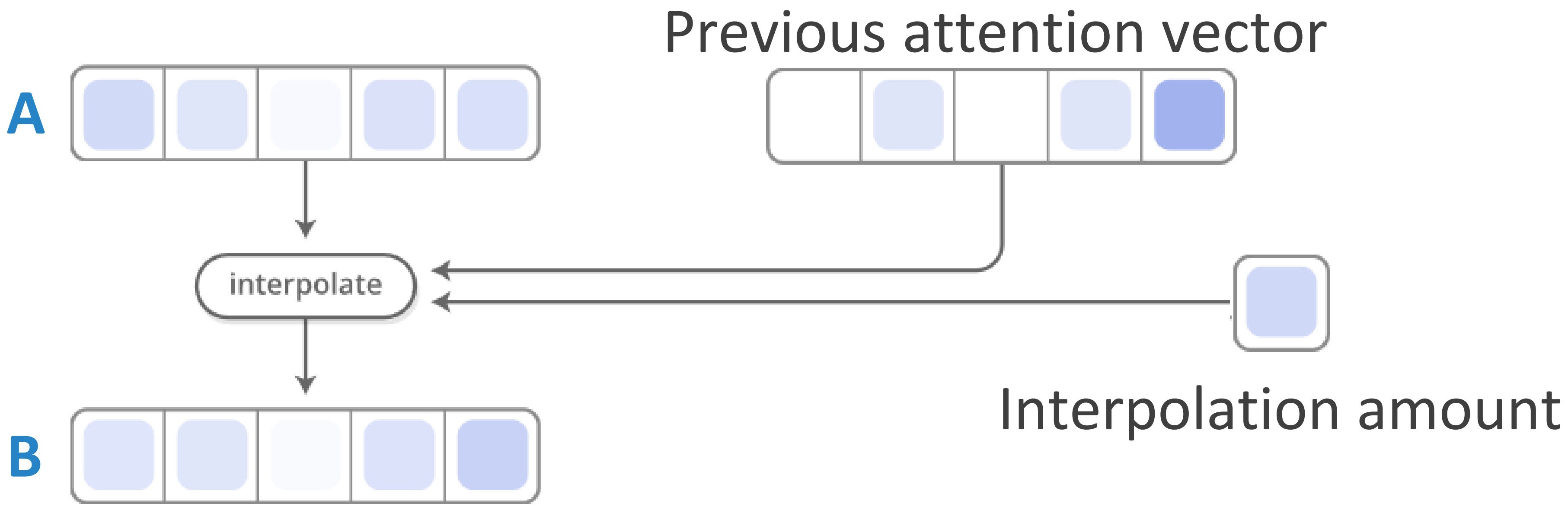
NTM Attention Focusing

1. Generate content-based memory indexing



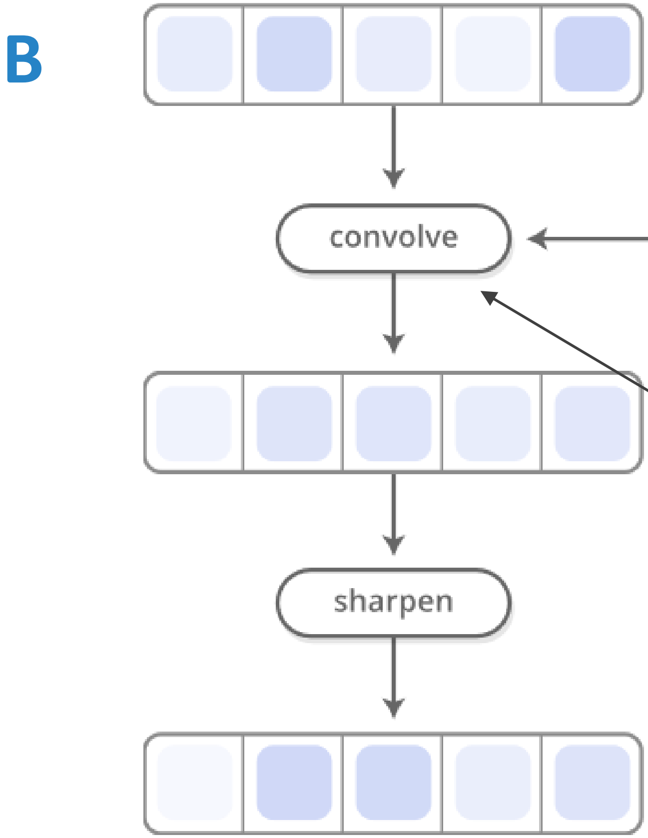
NTM Attention Focusing

2. Interpolate with attention from previous time

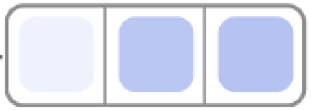


NTM Attention Focusing

3. Generate location-based indexing



Shift distribution filter



Determines how we move between the locations in memory

Sharpen the distribution for final memory access

Practical Use?

- Not really of these models (but inspired Neural Algorithmic Reasoning)
- Not straightforward to train
- Advantages over 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 (enough for TF to list it as official)

Take Home Messages

- Attention.. Attention.. and, again, attention
- Hierarchy
  - Can be directly encoded in RNN, or learned adaptively
  - Used to mitigate vanishing gradients and learn hierarchical encodings
- Memory and RNN
  - Efficient use of dynamic memory
  - External memory for search and recall tasks
  - Read/write memory for neural reasoning

Next Weeek

Generative Deep Learning module

- Explicit density models
- Implicit/sampling models
- Neural diffusion
- …

After Next Weeek

…we will continue on April 30th!

- No lecture 23, 24 and 25th April
- Lecture on 30th April, 02 May