



Recurrent Neural Networks

Processing Sequences

Summary

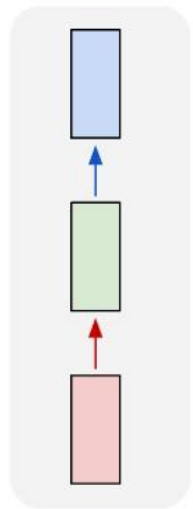
- Variable size input/output
- Recurrent neural networks
- Sequence to sequence learning
- LSTM
- GRU
- Attention
- Dynamic memory networks for question answering

The background features a collection of light gray geometric elements. In the top left, there is a set of vertical bars of varying heights. To their right is a 2x2 grid of circles. Further right is a large square, followed by a large circle. On the far right, there are horizontal bars. Below the top-left bars is a large circle. In the center, there is a large square. To its right are more vertical bars. Below the central square is another large square. On the right side, there are three circles stacked vertically. The text "Variable size input/output" is centered in the middle of the image.

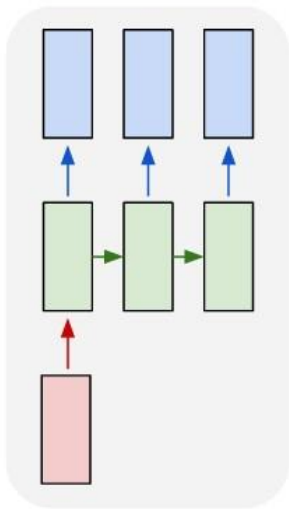
Variable size input/output

Different options for input/output cardinality

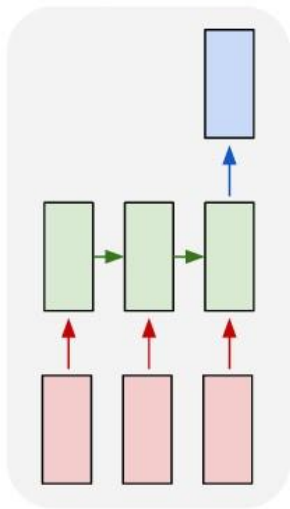
one to one



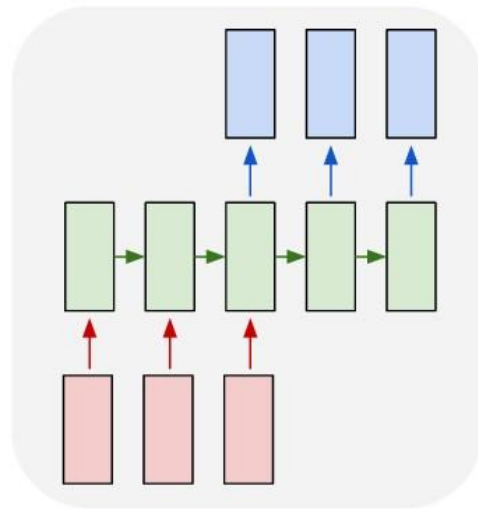
one to many



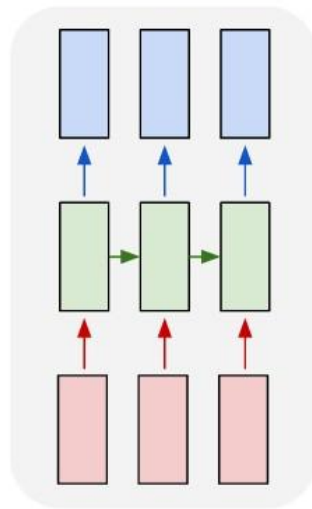
many to one



many to many



many to many



Examples

- One to many
 - Image captioning
- Many to one
 - Sentiment analysis
 - Question answering (single word)
- Many to many
 - Translation
 - Question answering
 - (synced) Video frames classification

Image captioning

Image source: Show and Tell

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A skateboarder does a trick on a ramp.



A dog is jumping to catch a frisbee.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.



A herd of elephants walking across a dry grass field.



A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.



A yellow school bus parked in a parking lot.



Describes without errors

Describes with minor errors

Somewhat related to the image

Unrelated to the image

Sentiment analysis

I would rather watch my avocado grow for 90 minutes than watching that match. (me)

Probably: negative

Translation

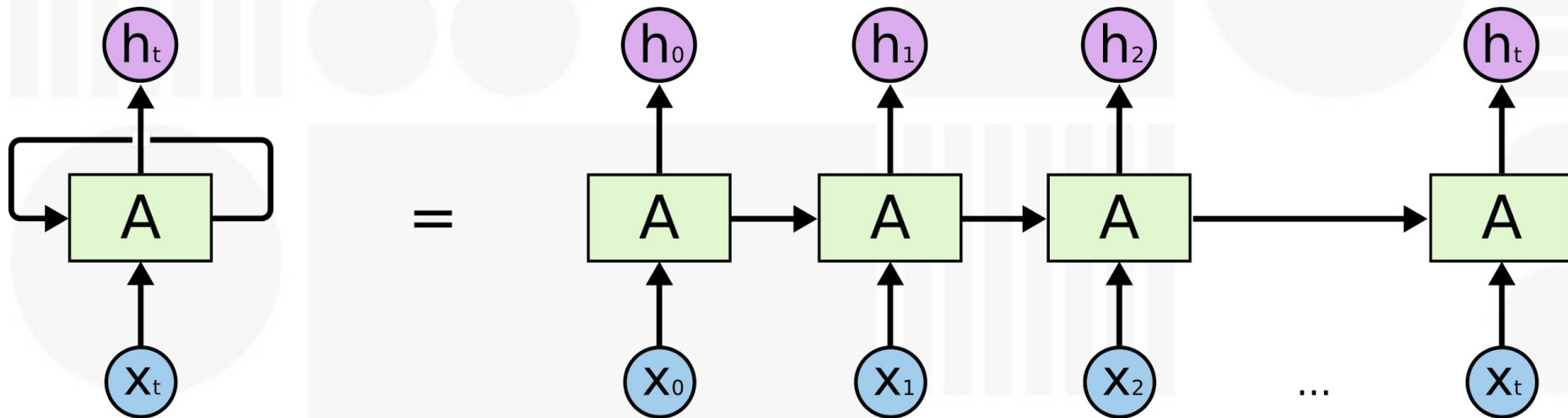
I don't know what I am talking about (English) =

Ich weiß nicht, wovon ich rede (German)

The background features a collection of light gray geometric shapes: a large square in the center, several smaller squares of varying sizes, a circle on the left, a large circle on the right, a group of four small circles in the top left, and several vertical bars of different heights and widths scattered throughout.

Recurrent Neural Networks

RNNs - the idea



- The one on the right is rolled out version of the network on the left
- All **A**-s on the right are the SAME network - they have the same weights

But how do we train such a thing?

- Typical approach would be back propagation
- Steps may be many, even ∞ (we'll see later)
- The solution:
 - Unroll for fixed number of steps
 - Backpropagate the error
 - You have to combine all the gradients!

Notes

- With many steps the gradient for the first few either:
 - Vanishes (most of the time)
 - Explodes (less frequently)
 - A solution - gradient clipping (or later slides)
- We typically use tanh for activation function
- Take extra care with batches
- Deep RNNs
- Bi-directional RNNs

Gradient clipping

$$g = \frac{df}{dg}$$

if $g > \alpha$

$$\hat{g} = \frac{\alpha}{||g||} g$$

Deep RNNs

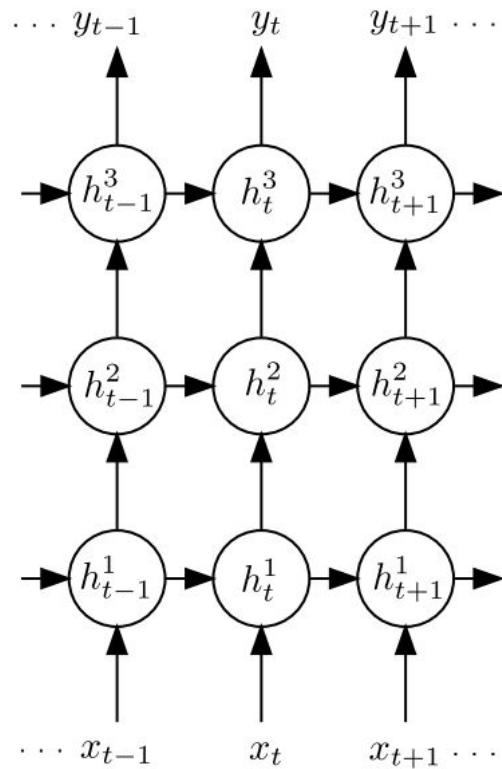
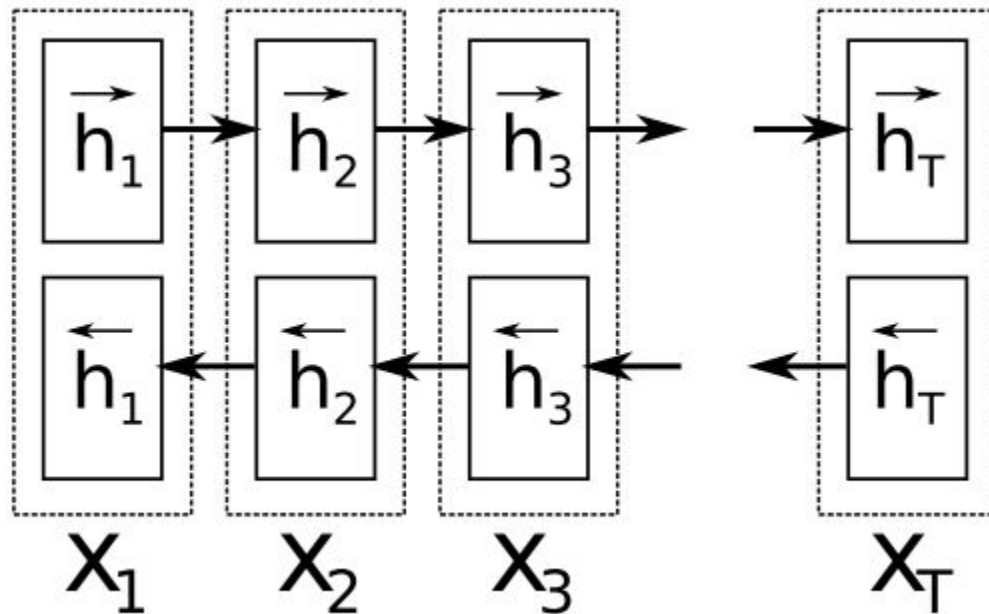


Image source: Hybrid speech recognition with deep bidirectional LSTM

Bidirectional RNN





Sequence to Sequence learning

The task

- We are given a sequence
- Our target is another sequence
- The two sequences may be of different size and order

Example

I don't know what I am talking about (English) =

Ich weiß nicht, wovon ich rede (German)

Note that:

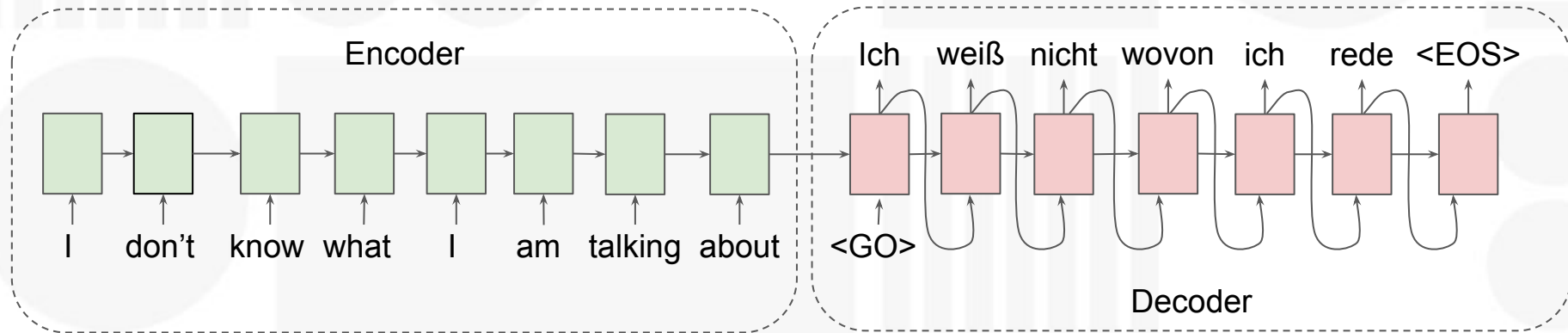
- The length is different
- The order is different

How do we do it

Two networks:

- Encoder that converts the input to machine-readable
- Decoder that produces the desired output

Encoder / Decoder



NOTE: in fact we are using embeddings not the words

Notes

- Experiments proved better results if we reverse input
- Same problems as with RNNs
 - Solved using LSTM/GRU
- We have problems with long sentences
 - Solved using attention

The background is a light gray canvas featuring several abstract geometric elements. In the top left, there is a vertical stack of eight thin, light gray rectangles. To their right is a 2x2 grid of four light gray circles. Further right is a single light gray square. In the top right corner, there are five horizontal light gray stripes. On the left side, there is a single light gray circle. The center of the image is dominated by a large, light gray square. To its right, there are five vertical light gray stripes. In the bottom left, there is a light gray square. In the bottom right, there is another light gray square. Along the far right edge, there are three light gray circles stacked vertically, with the bottom one partially cut off.

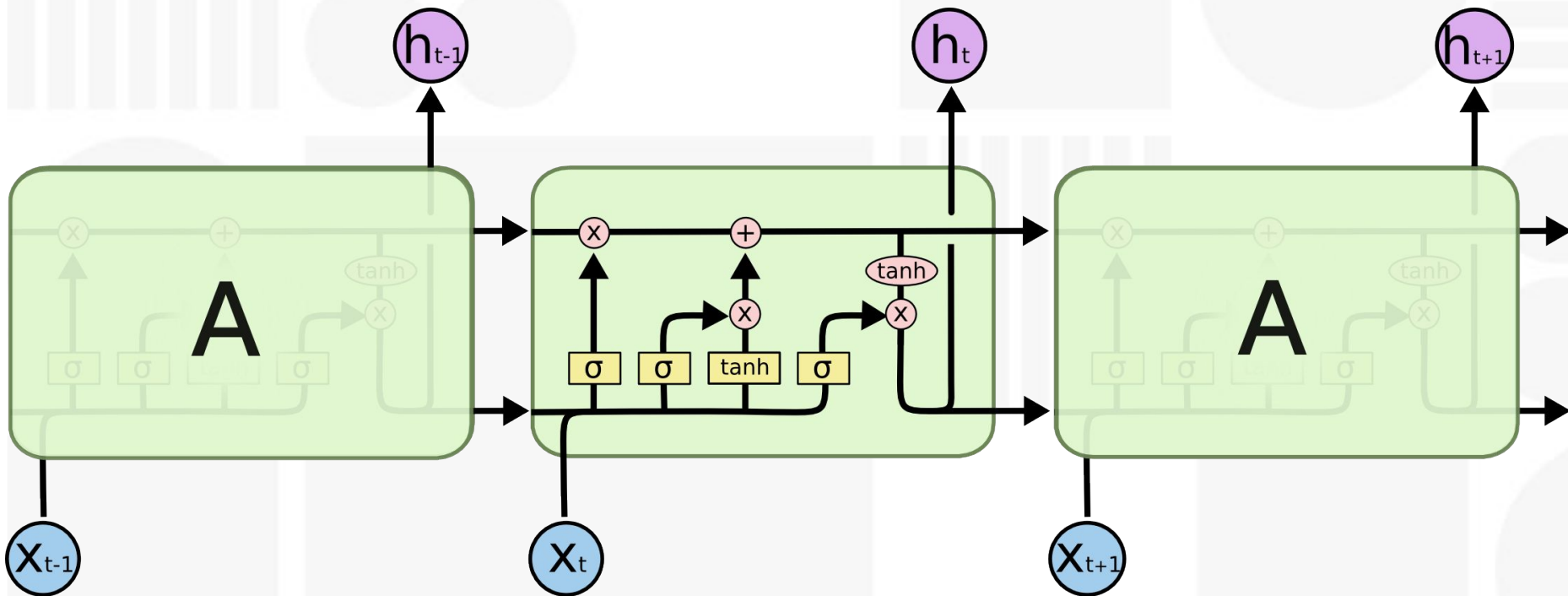
LSTM

What is it?

- Stands for “long short term memory”
- Special type of RNN cell
- Able to decide what to forget/remember
- Alleviates the vanishing/exploding gradient problem

Picture or it did not happen!

source: Cris Olah's blog

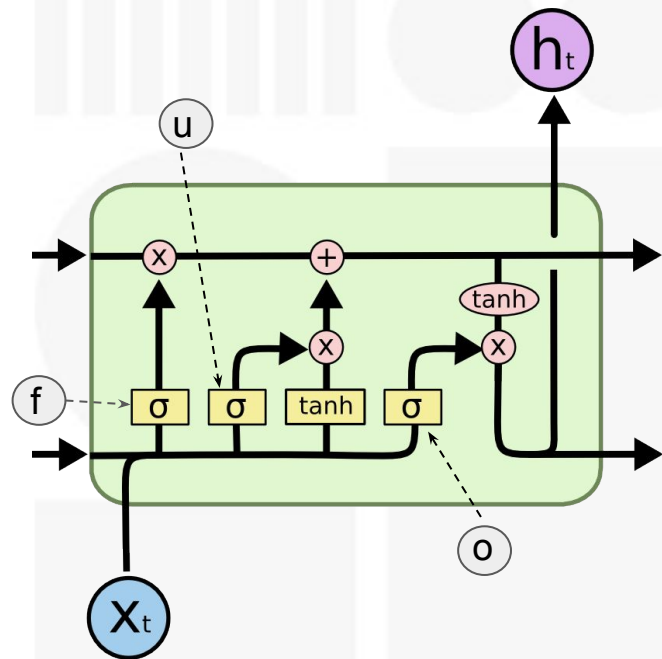


Doesn't help much, does it?

Components

- We have two states - hidden state and cell state
 - Hidden state equivalent to the typical RNN state
 - Cell state 'flows freely' and propagates gradients
- Three gates:
 - Forget gate
 - Update gate
 - Output gate

Step by step



Input: $X = \text{concat}(X_t, H_{t-1})$
(size: $p + n$)

Forget gate: $f = \sigma(X * W_f + b_f)$ (size n)

Update gate: $u = \sigma(X * W_u + b_u)$ (size n)

Output gate: $o = \sigma(X * W_o + b_o)$ (size n)

Input: $X' = \tanh(X * W_c + b_c)$ (size n)

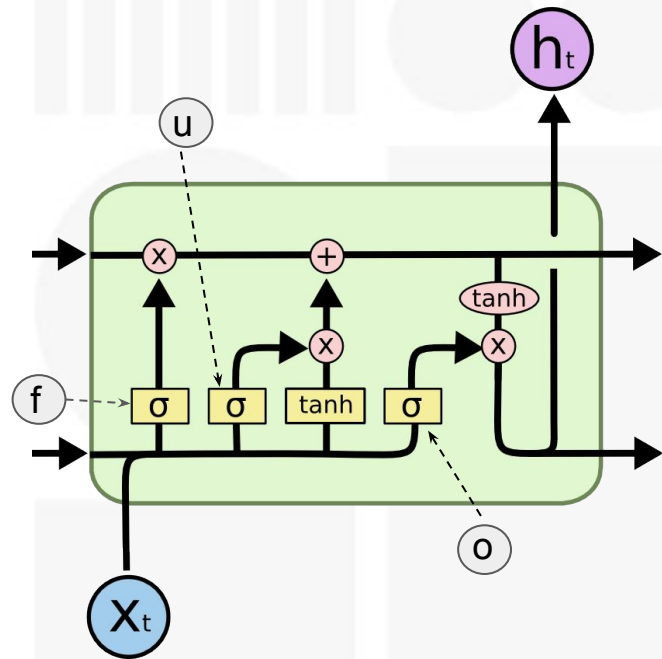
New C: $C_t = f * C_{t-1} + u * X'$ (size n)

New H: $H_t = o * \tanh(C_t)$ (size n)

Output (if exists):

$Y_t = \text{softmax}(H_t * W + b)$ (size m)

How does it help?



- There is a direct path for the gradient - no activation applied on C
- The cell 'learns' what to remember for longer period

Image Source: Chris Olah's blog (labels added)

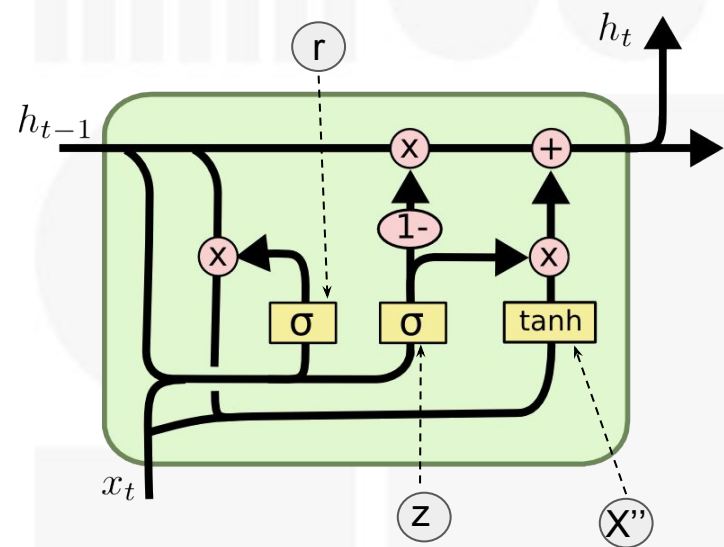
The background is a light gray canvas featuring several abstract geometric elements. In the top left, there is a cluster of eight vertical gray bars of varying heights. To their right is a 2x2 grid of four identical light gray circles. Further right is a large, solid light gray square. In the top right corner, there is a large light gray circle and a series of five horizontal gray bars. On the left side, there is a large light gray circle and a solid light gray square. In the center, a large light gray square serves as a backdrop for the text. To the right of this central square is another set of eight vertical gray bars. At the bottom right, there is a solid light gray square and a partial view of a light gray circle on the edge.

GRU

What is it?

- Stands for “gated recurrent unit”
- Similar to LSTM
- Has only two gates instead of 3
- Exposes its whole state to the outside

GRU - how does it work?



Input: $X = \text{concat}(X_t, h_{t-1})$ (size: $p + n$)

Update gate: $z = \sigma(X * W_z + b_z)$ (size n)

Reset gate: $r = \sigma(X * W_r + b_r)$ (size n)

Input:

$X' = \text{concat}(X_t, (r * h_{t-1}))$ (size $p + n$)

$X'' = \tanh(X' * W_c + b_c)$ (size n)

New h: $h_t = (1 - z) * h_{t-1} + z * X''$ (size n)

Output (if exists):

$Y_t = \text{softmax}(h_t * W + b)$ (size m)

LSTM vs GRU (1 / 2)

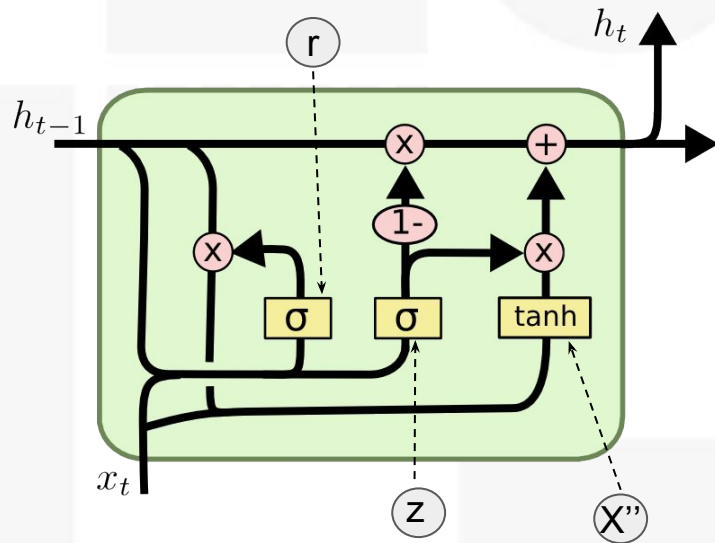
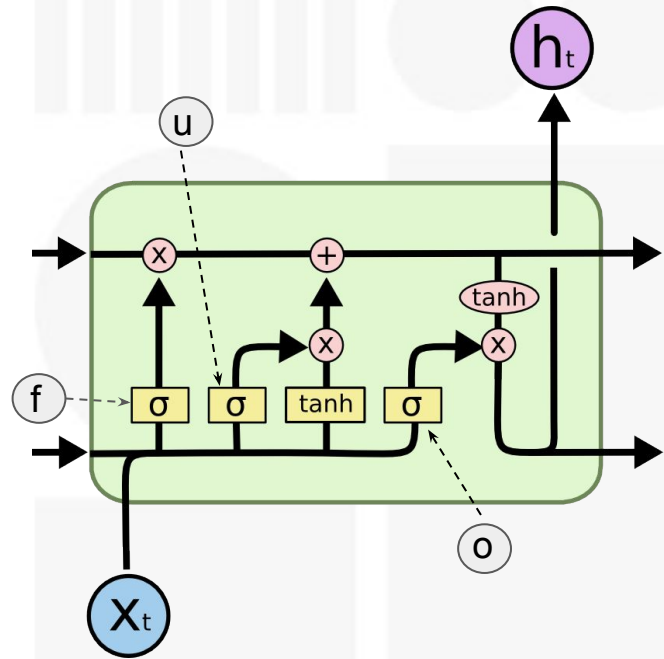


Image Source: Chris Olah's blog (labels added)

LSTM vs GRU (2 / 2)

Input: $X = \text{concat}(X_t, H_{t-1})$

Forget gate: $f = \sigma(X * W_f + b_f)$

Update gate: $u = \sigma(X * W_u + b_u)$

Output gate: $o = \sigma(X * W_o + b_o)$

Input: $X' = \tanh(X * W_c + b_c)$

New C: $C_t = f * C_{t-1} + u * X'$

New H: $H_t = o * \tanh(C_t)$

Output (if exists):

$$Y_t = \text{softmax}(H_t * W + b)$$

Input: $X = \text{concat}(X_t, H_{t-1})$

Update gate: $z = \sigma(X * W_z + b_z)$

Reset gate: $r = \sigma(X * W_r + b_r)$

Just two gates!

Input:

$$X' = \text{concat}(X_t, (r * H_{t-1}))$$

$$X'' = \tanh(X' * W_c + b_c)$$

No C!

$$\text{New H: } H_t = (1 - z) * H_{t-1} + z * X''$$

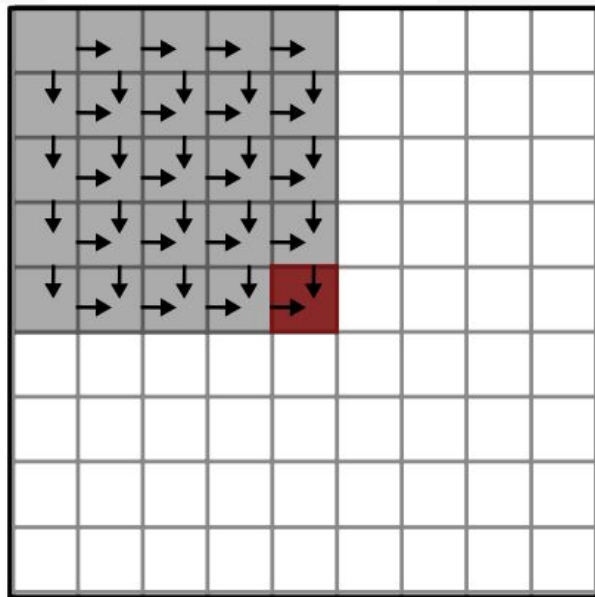
Output (if exists):

$$Y_t = \text{softmax}(H_t * W + b)$$

Notes on GRU and LSTM

- GRU has fewer parameters
 - faster to compute
 - faster to train
- Empirically GRU performs on par with LSTM
- GRU is getting more popular recently
- Other architectures for recurrent units exist

MDLSTM



(a) Standard MD-LSTM

Image Source: Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation (cropped)



ATTENTION

What is attention

- Trainable function
- We enable the decoder to look at encoder outputs
- No longer rely on single encoder hidden state to encode all information
- We can visualize the information used for a prediction

What is attention

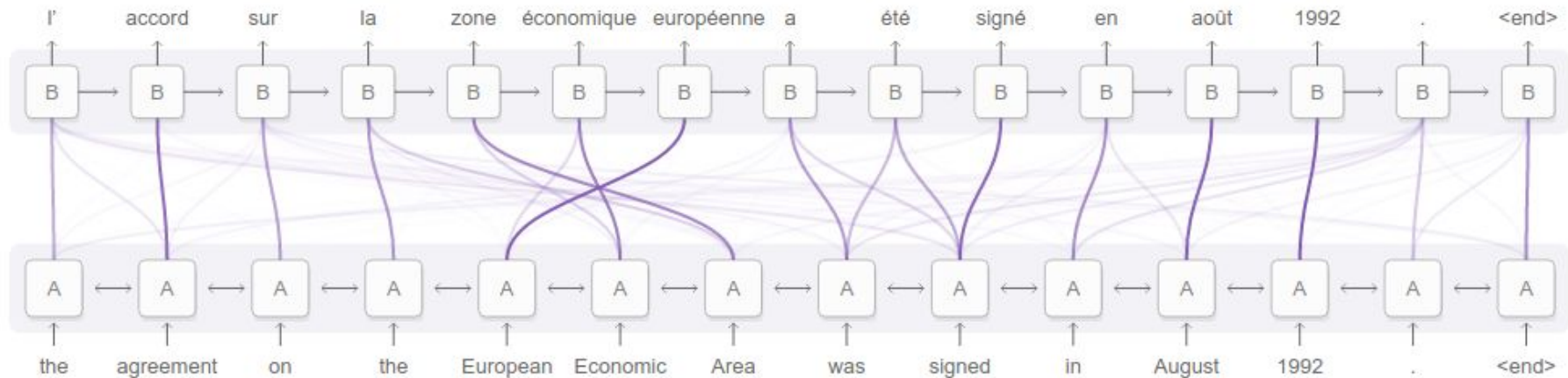


Image Source: Attention and Augmented Recurrent Neural Networks (colah's blog)

Hard Attention

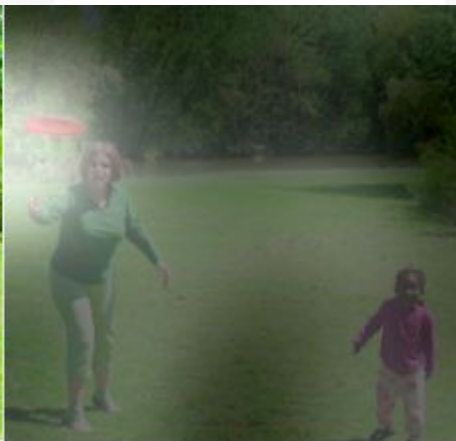
- Binary value - either 1 or 0 for each feature
- Fast and easy to compute during forward pass
- Non-differentiable -needs different mechanism to train
 - reinforcement learning
 - variance reduction

Soft attention

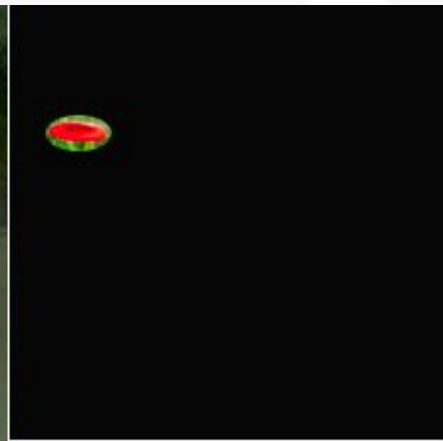
- Real value between 0 and 1 for each feature
- Each feature is included to some extent
- Differentiable
 - can use backpropagation as usual
- More common

An example

Soft attention



Hard attention



A woman is throwing a frisbee in a park.

Image Source: Show attend and tell (hard attention version added separately)

Bahdanau attention

$$c_i = \sum_{j=1}^{T_x} a_{ij} h_j$$

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^T \exp(e_{ik})}$$

$$e_{ij} = a(s_{i-1}, h_j)$$

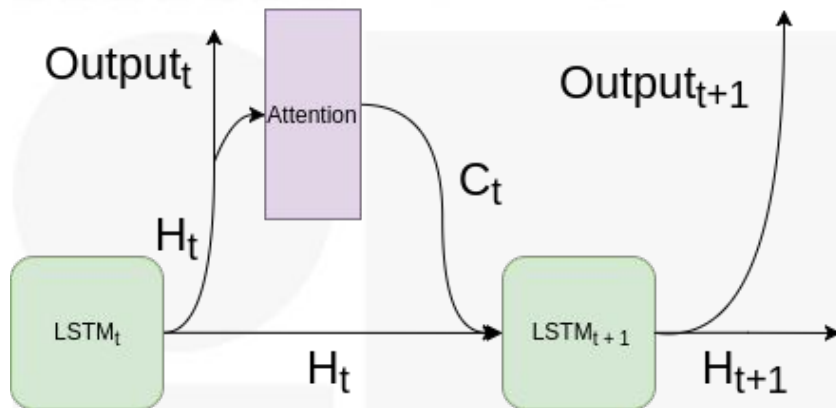
- New input component c_i called “context”
- h_j - encoder output at step j
- a_{ij} - weight of the j -th annotation(encoder output)
- e_{ij} - attention energy for the j -th annotation
- a - an attention function(a fully connected layer)
- s_{i-1} - decoder hidden state after step $i - 1$
- NOTE: for each step we iterate over **all** encoder outputs

Luong attention

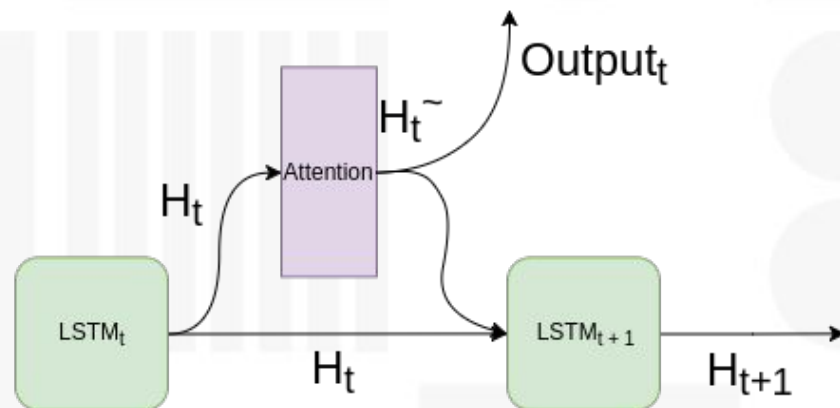
- Proposes more attention functions
- Attention layer comes **after** the RNN cell
- Proposes “global” and “local” versions
 - global is more similar to Bahdanau
 - local: p_i for each decoder step, consider $[p_i - D, p_i + D]$
 - monotonic version $p_i = i$
 - predictive - p_i is computed by a trained function
 - still differentiable

Bahdanau vs Luong

Bahdanau Attention

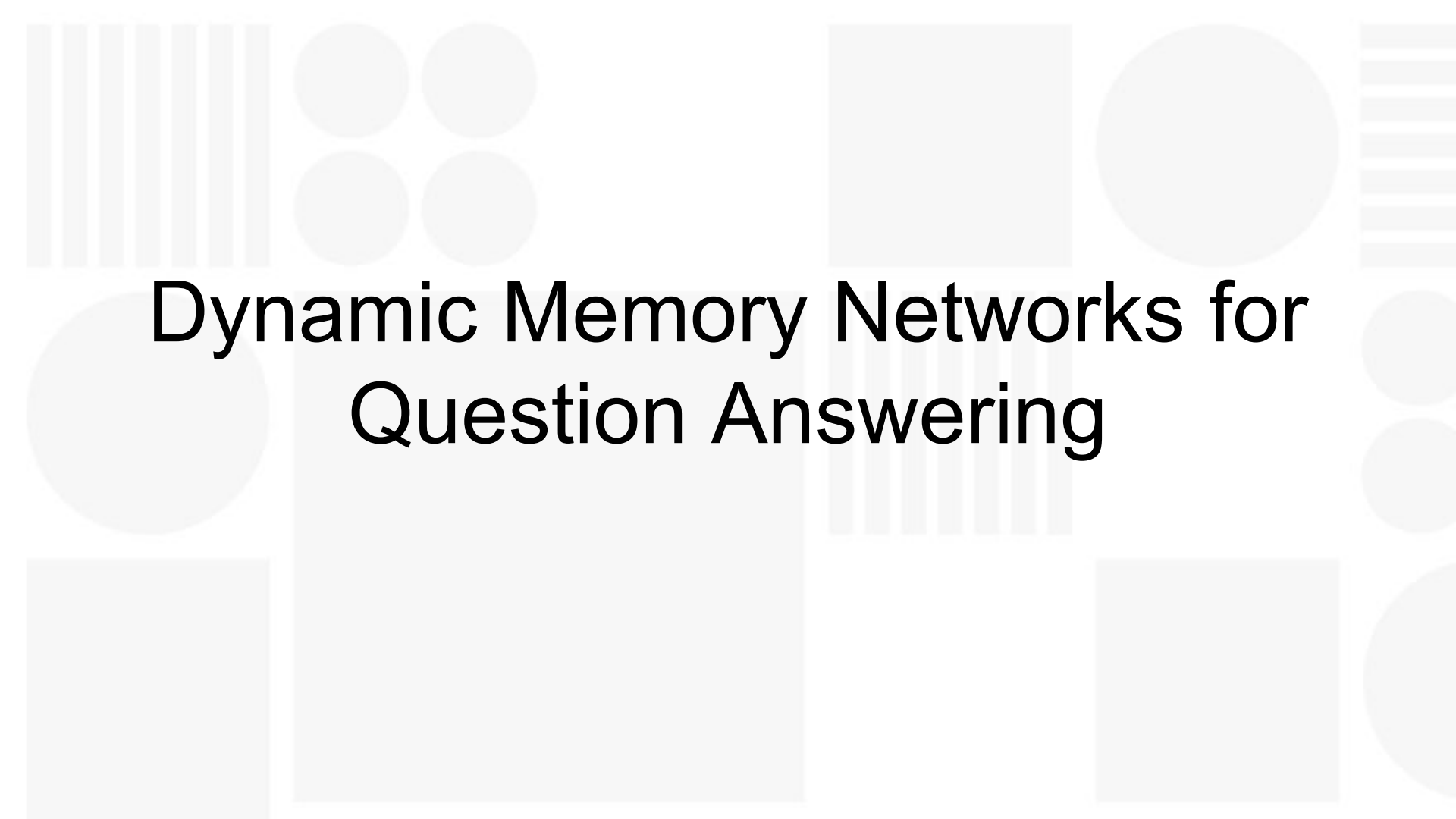


Luong Attention



Different attentions

- Neural turing machines - using external memory
- Adaptive computation time - dynamically deciding how many cells we need
- Neural Programmer - dynamically decide **what** cells to run



Dynamic Memory Networks for Question Answering

Question answering - the problem

- We are given a set of “facts”
- We are given a single question
- We need to produce an answer to that question

Question answering - example

Two supporting fact example(from the bAbI dataset):

- 1 Mary got the milk there.
- 2 John moved to the bedroom.
- 3 Sandra went back to the kitchen.
- 4 Mary travelled to the hallway.
- 5 Where is the milk? hallway 1 4

Dynamic Memory Networks for question answering

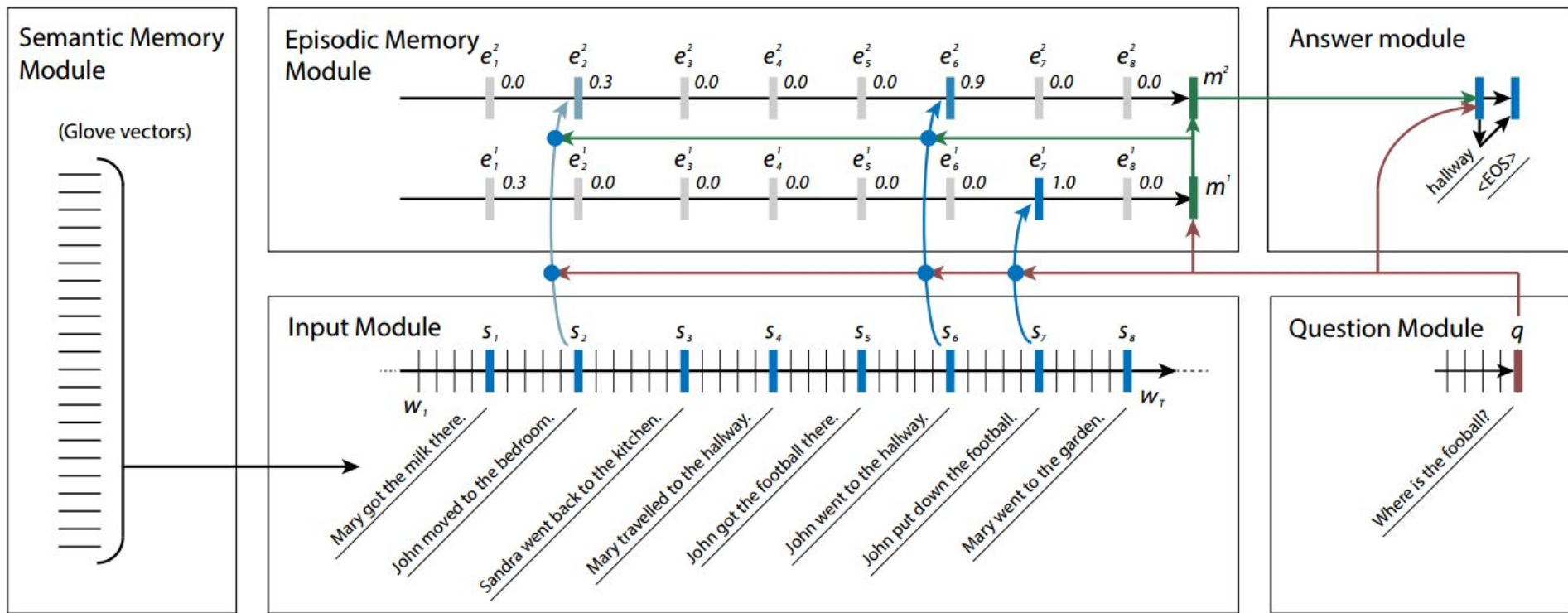
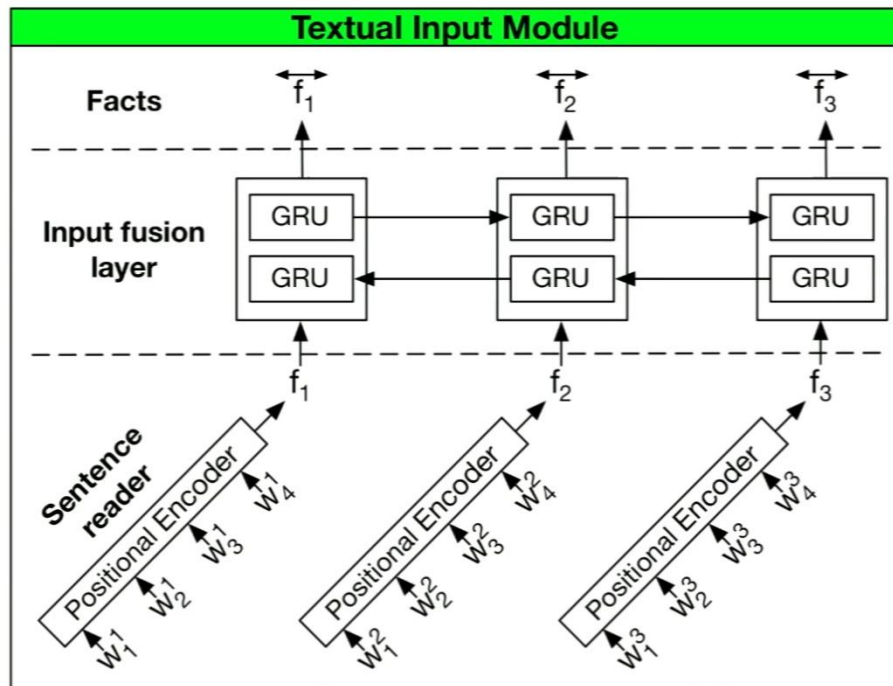


Image Source: Dynamic Memory Networks for Question Answering
(Chris Manning & Richard Socher)

Dynamic Memory Networks for question answering



Concatenate the encodings in the two directions

Image Source: *Dynamic Memory Networks for Visual and Textual Question Answering*
(Xiong et al)

DMNs - The question

- Standard GRU RNN

$$q_t = GRU(v_t, q_{t-1})$$

DMNs - Episodic Memory

$$z_i^t = [s_i \circ q; s_i \circ m^{t-1}; |s_i - q|; |s_i - m^{t-1}|]$$

$$Z_i^t = W^{(2)} * \tanh(W^{(1)} * z_i^t + b^{(1)}) + b^{(2)}$$

$$g_i^t = \frac{\exp(Z_i^t)}{\sum_{k=1}^{M_i} \exp(Z_k^t)}$$

$$h_i^t = g_i^t * GRU(c_t, h_{t-1}^i) + (1 - g_i^t) * h_{t-1}^i$$

Episodic Memory - notes

- When g_i^t is close to zero, we entirely ignore the GRU, when it is close to 1, we entirely ignore the hidden state
- We measure cosine similarity between vectors
- On the first layer m^{t-1} is the question itself
- We could replace the softmax with sigmoids at each point
- Another GRU over the memory states

DMNs - The answer

- For the bAbI dataset the answer is a single word, so we use softmax
- For variable length answers we use a decoder much like in the translation case

Dynamic Memory Networks - the example

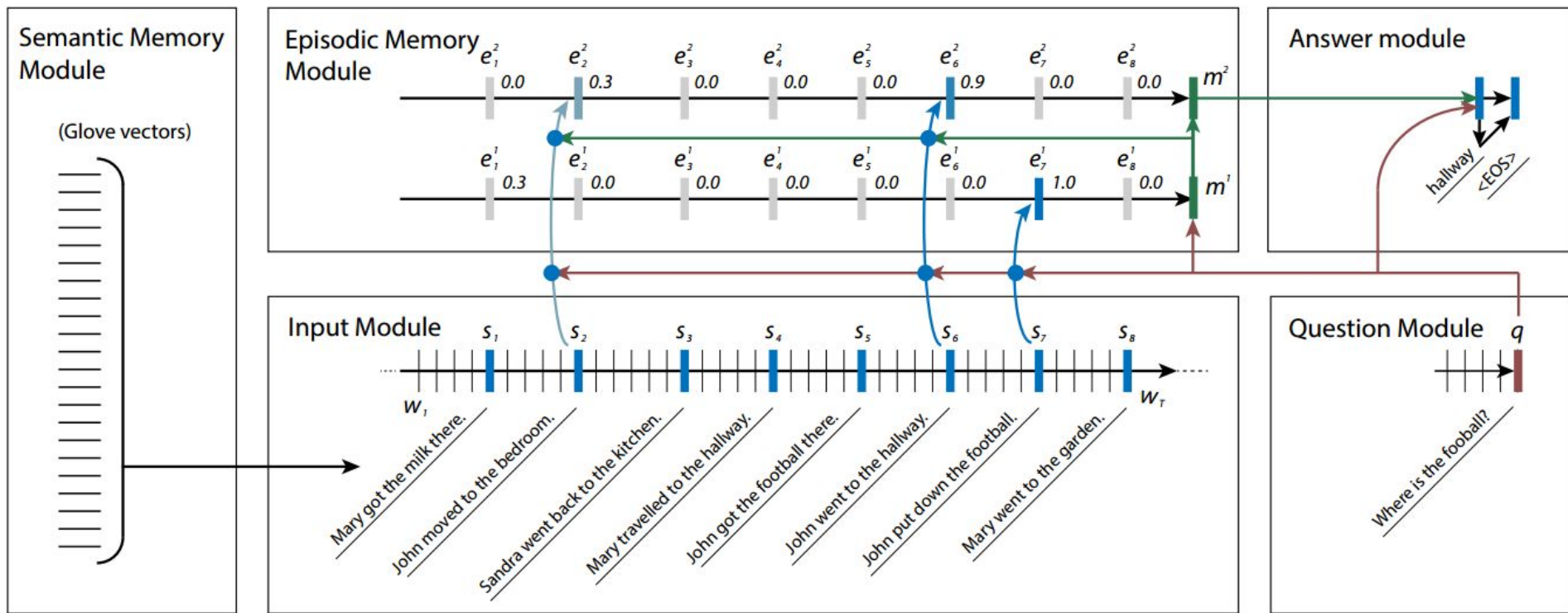
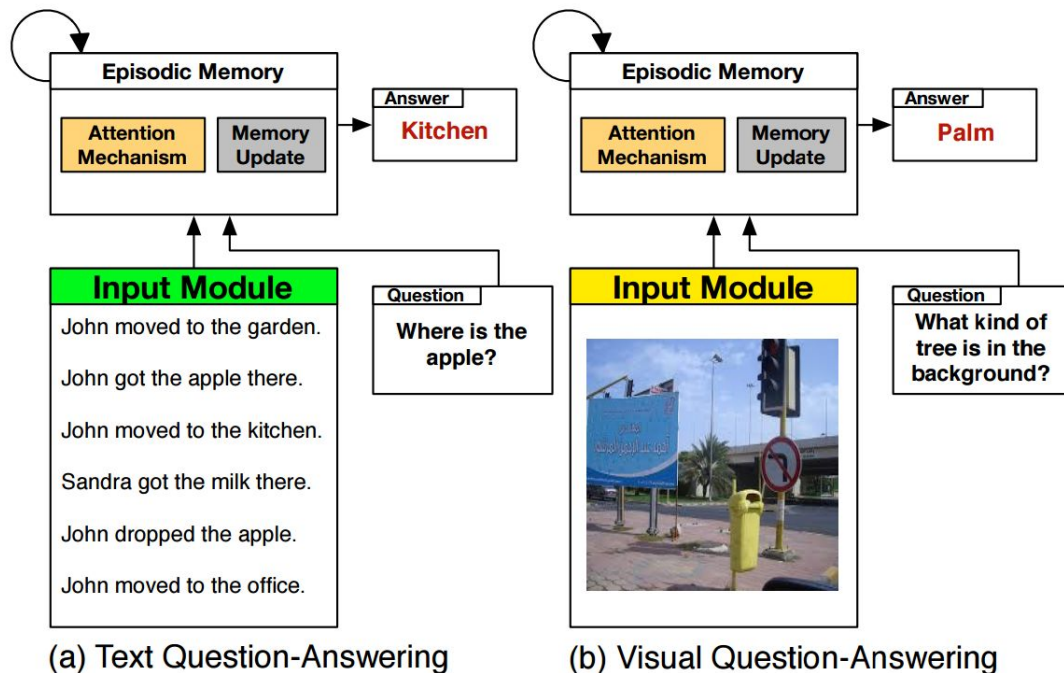


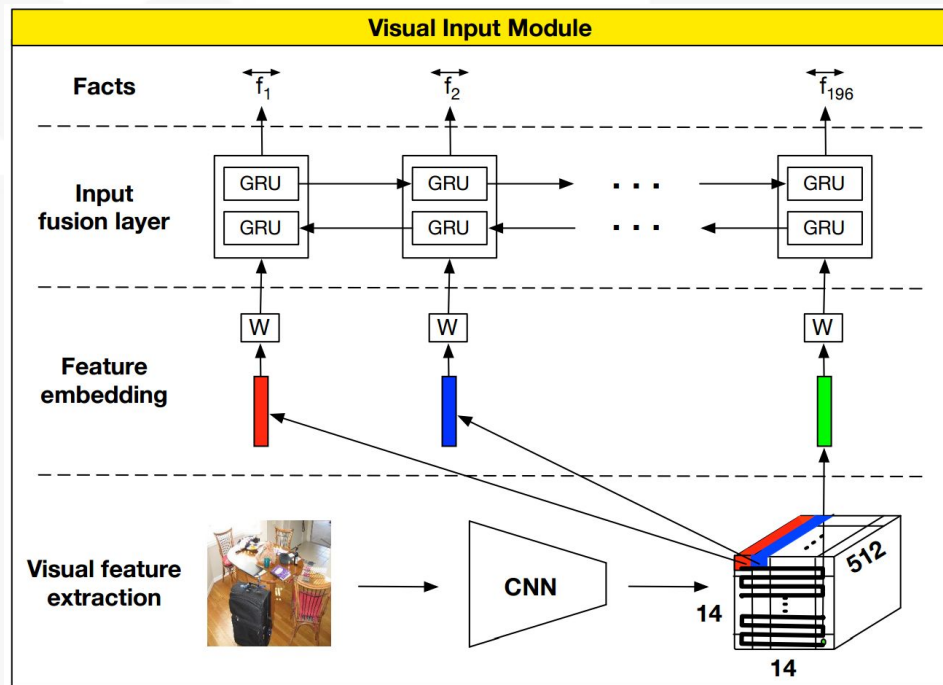
Image Source: Dynamic Memory Networks for Question Answering
(Chris Manning & Richard Socher)

DMNs for visual question and answering (VQA)



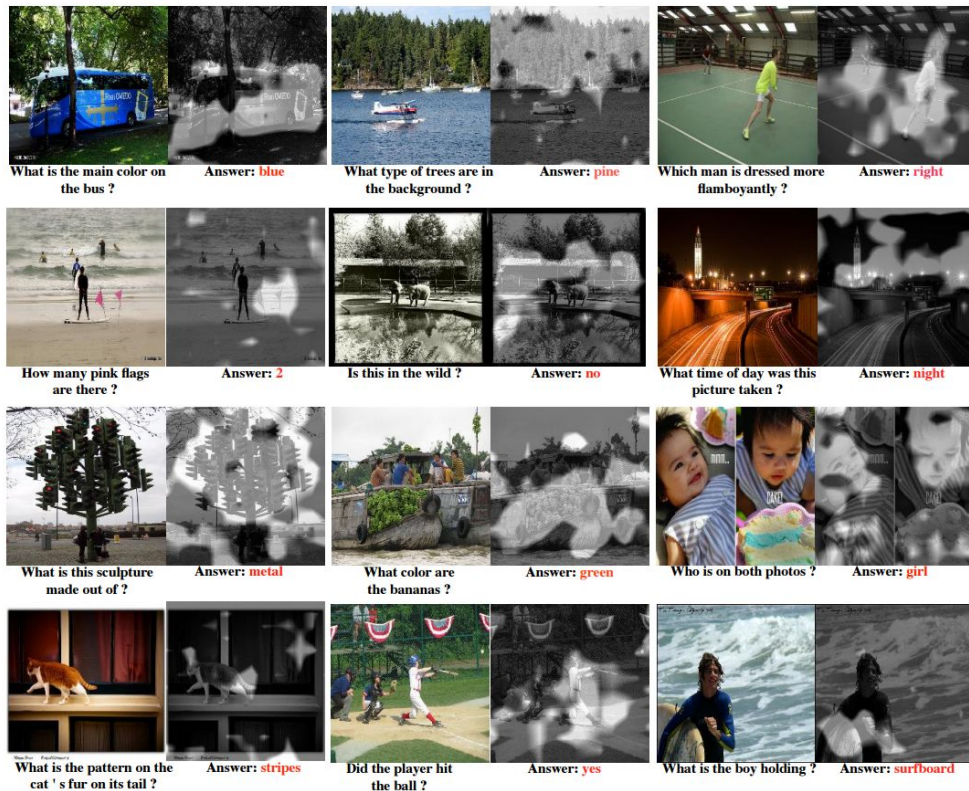
*Image Source: Dynamic Memory Networks for Visual and Textual Question Answering
(Xiong et al)*

VQA input module for DMNs



*Image Source: Dynamic Memory Networks for Visual and Textual Question Answering
(Xiong et al)*

VQA - visualizing the attention



*Image Source: Dynamic Memory Networks for Visual and Textual Question Answering
(Xiong et al)*

Visual Question answering with DMNs - notes

- Instead of word embeddings we use the feature vectors
- The network is the same but for the input module
- We introduce a “snake-like” order to the feature vectors
- Attention is actually meaningful



questions any have you do

References (1/2)

- Tensorflow and deep learning - without a PhD by Martin Görner(RNNs): <https://www.youtube.com/watch?v=vq2nnJ4g6N0&t=107m25s>
- Chris Olah's blog - RNNs and LSTM <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- Andrej Karpathy- The Unreasonable Effectiveness of Recurrent Neural Networks: <http://karpathy.github.io/2015/05/21/rnn-effectiveness/>
- Cs231n - Recurrent Neural Networks: <https://youtu.be/yCC09vCHzF8>
- Sequence to Sequence Learning <https://arxiv.org/pdf/1409.3215.pdf>
- Comparison of LSTM and GRU <https://arxiv.org/pdf/1412.3555v1.pdf>
- Chris Olah - Attention and Augmented Recurrent Neural Networks <https://distill.pub/2016/augmented-rnns/>

References (2/2)

- Practical PyTorch: Translation with a Sequence to Sequence Network and Attention: <https://github.com/spro/practical-pytorch/blob/master/seq2seq-translation/seq2seq-translation.ipynb>
- Attention and memory in deep learning and nlp <http://www.wildml.com/2016/01/attention-and-memory-in-deep-learning-and-nlp/>
- Neural Machine Translation by Jointly Learning to Align and Translate (Bahdanau et al.)- <https://arxiv.org/abs/1409.0473>
- Effective Approaches to Attention-based Neural Machine Translation (Luong et al.) - <https://arxiv.org/pdf/1508.04025.pdf>
- Show attend and tell (Xu et al.) - <https://arxiv.org/pdf/1502.03044.pdf>
- Parallel Multi-Dimensional LSTM, With Application to Fast Biomedical Volumetric Image Segmentation - <https://arxiv.org/pdf/1506.07452.pdf>
- Dynamic Neural Networks for Question Answering (Stanford University lecture): <https://youtu.be/T3octNTE7ls>
- Dynamic Memory Networks for Question Answering (Raguvanshi & Chase): <https://cs224d.stanford.edu/reports/RaghuvanshiChase.pdf>
- The bAbI dataset: <https://research.fb.com/downloads/babi/>
- Dynamic Memory Networks for Visual and Textual Question Answering: <https://arxiv.org/pdf/1603.01417.pdf>