Lecture 14. Recurrent Neural Networks, Attention, and the Transformer

COMP90051 Statistical Machine Learning

Zahra Dasht Bozorgi

This lecture

- Recurrent networks for modelling sequences
  - recurrent units
  - back-propagation through time
  - long-short term memory
- Transformers and attention

Recurrent Networks

A DNN tailored to variable length sequential inputs

Sequential input

- Until now, we have assumed fixed-sized input
  - Vectors of features $\boldsymbol{x}$ in d dimensions
  - Matrices of pixels in an image
- What if our input is a sequence?
  - Frames in a video clip
  - Time steps in an audio clip
  - Words in a sentence
  - A protein sequence
  - Stock prices over time …
- How can we model this in a DNN?

FCNNs poor for sequences

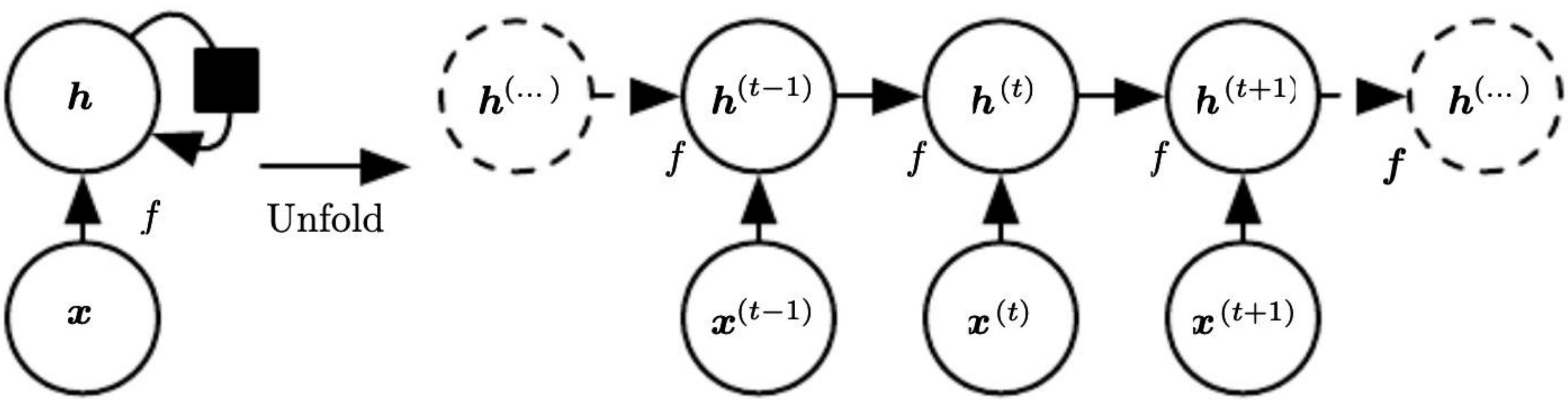
- Consider classifying sentences
  - "This is the worst movie of all time, a real stinker" → ☹
  - "The movie is a real stinker" → ☹
- Issue: inputs are *different lengths*
  - pad them with empty "words" to be a fixed size
- Issue: how do we *represent words* as vectors?
  - learn an "embedding" vector for each word
- Issue: phrases have *similar meaning even when at different locations*
  - "a real stinker" is a key predictive feature
  - if we naively apply FCNN needs to learn this concept repeatedly

ConvNets for Sequences?

- Sequences are just rectangular shaped images (e.g., embedding dim. times length): apply CNNs
  - With 1D filters
  - The filter parameters are shared across time, and can find patterns in the input
- This is called the *time delay neural network*
- Downside:
  - receptive field of filters are limited to finite size, i.e., the width of the convolutional filters, which can be expanded with deeper networks

Recurrent Neural Nets (RNNs)

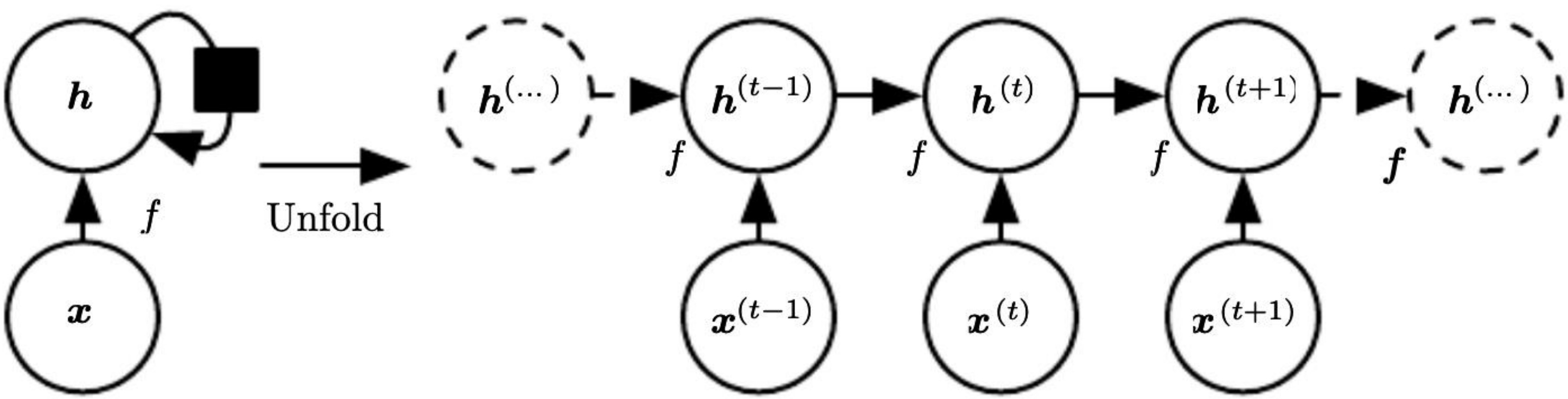
- RNNs create networks dynamically, based on input sequence
  - given sequence of inputs $\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \dots, \boldsymbol{x}^{(t)}$
  - process each symbol from left to right, to form a sequence of hidden states $\boldsymbol{h}^{(t)}$
  - each $\boldsymbol{h}^{(t)}$ encodes all inputs up to t



Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 10

RNNs as Very Deep Networks

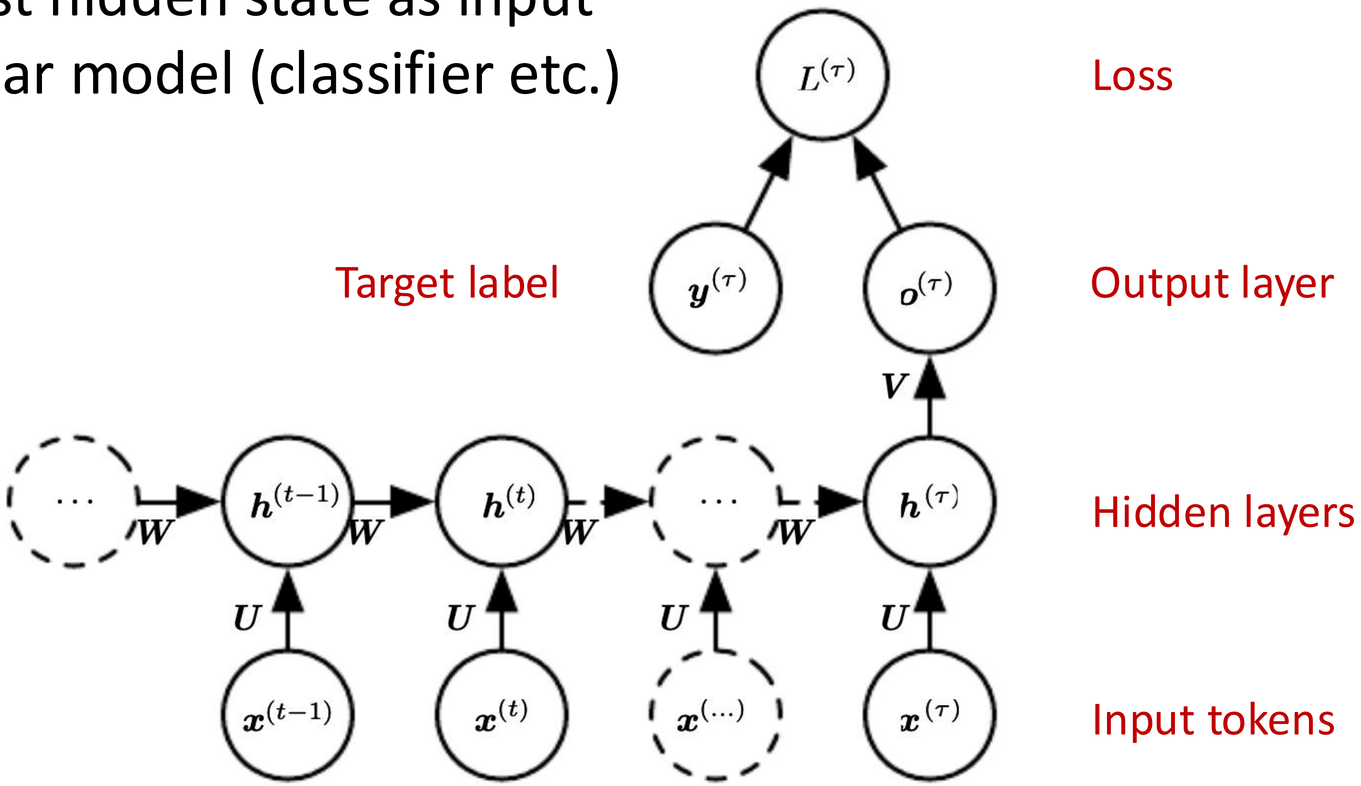
- Compared to NNets we've seen before:
  - unfolded RNN has depth equal to input sequence length
  - parameters shared between layers
- Can easily be 'unrolled' to cater to any input length



Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 10

RNN Applications: Seq. Classification

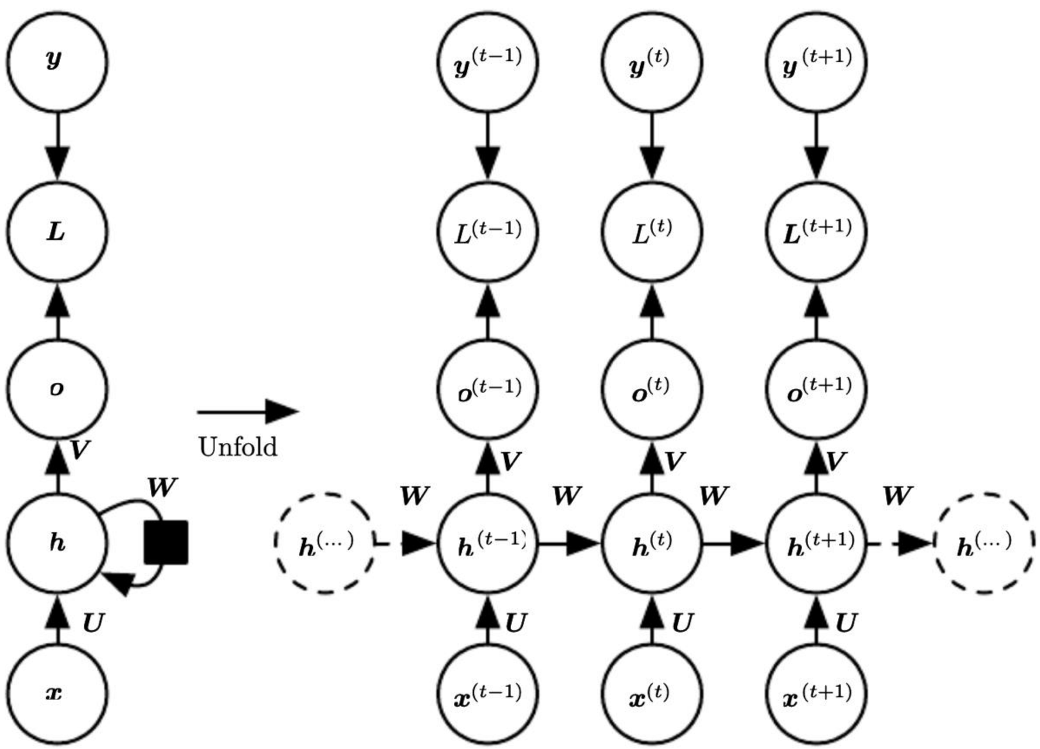
- Sequence classification: labelling sequence
  - use last hidden state as input to linear model (classifier etc.)



Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 10

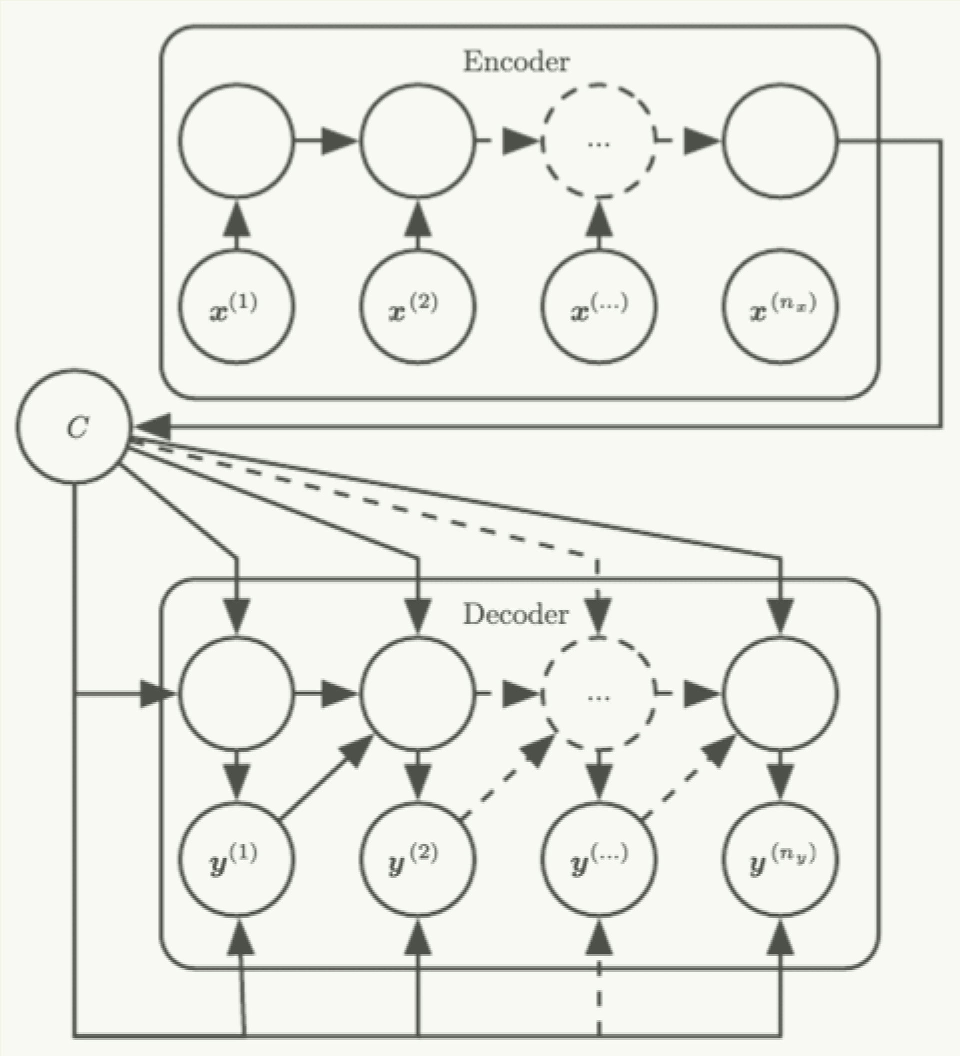
Sequence Tagging RNN

- Assign each item/token a label in sequence
  - Given targets per item, can measure loss per item



Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 10

Encoder-Decoder for Sequence Translation



E.g., English to French

Encoder RNN encodes input sequence into a context

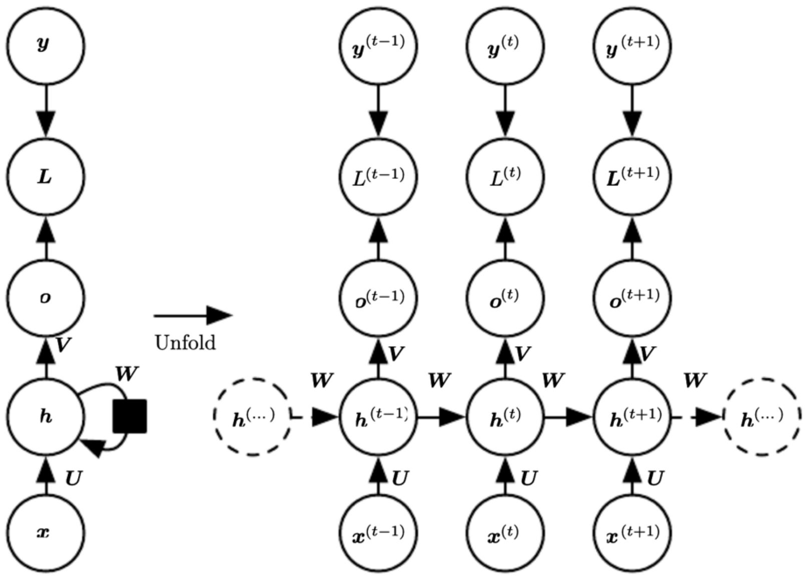
Decoder RNN acts like a tagger, where we're trying to (re)generate next inputs

Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 10

RNN Parameterisation

- Consider tagging RNN: define *f* as follows

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$



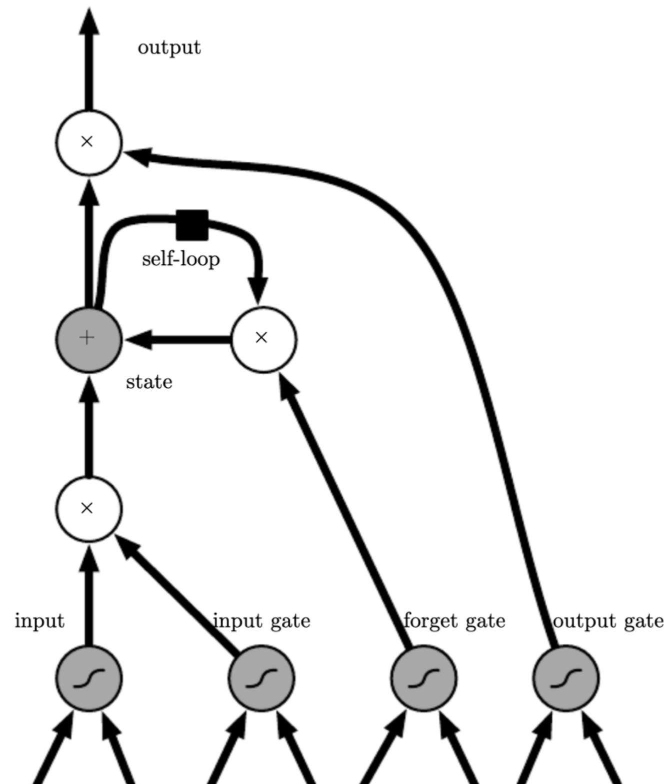
- Parameters are ***b, W, U, c, V***
  - not specific to timestep *t*, but shared across all positions
  - this “template” can get unrolled arbitrarily

Training RNNs: Backprop. Thru. Time

- Backpropagation algorithms can be applied to network
  - Called backpropagation through time (BPTT)
  - Gradients from the loss at every position must be propagated back to the very start of the network
- Suffers from gradient vanishing problem
  - Consider linear RNN, gradients of $\frac{\partial \boldsymbol{h}^{(T)}}{\partial \boldsymbol{h}^{(1)}} = \boldsymbol{W}^{T-1}$, thus can explode or vanish with large *T*, depending on largest eigenvalue of **W** (i.e., greater than / less than one).
  - Can't *learn* long distance phenomena (over 10+ steps)

Long Short-Term Memory (LSTM)

- In RNN, previous state is provided as an input
  - Multiplied by weight matrix, and non-linearity applied
- LSTM introduces state self-loop, based on copying
  - Takes copy of previous state, scaled by sigmoid forget gate
- Gradient magnitude now maintained
  - Can handle 100+ distance phenomena (vs 5-10 for RNN)



Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 10

Mini-summary

- Recurrent networks for modelling sequences
  * recurrent units
  * back-propagation through time
  * long-short term memory
  * applications

next: Transformers

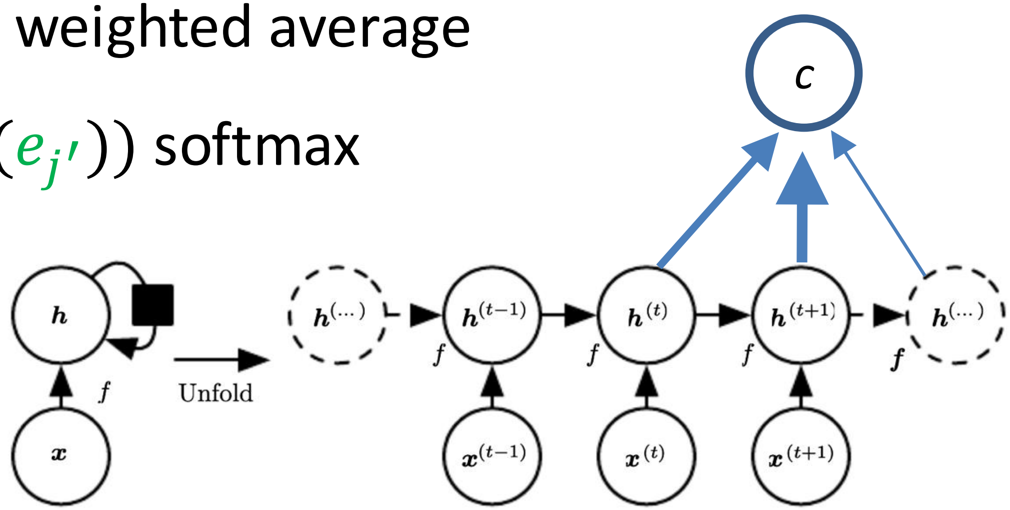
Transformers

~~More than meets the eye.~~

*A method for processing sequence inputs in highly parallelizable manner, using **attention**.*

Attention

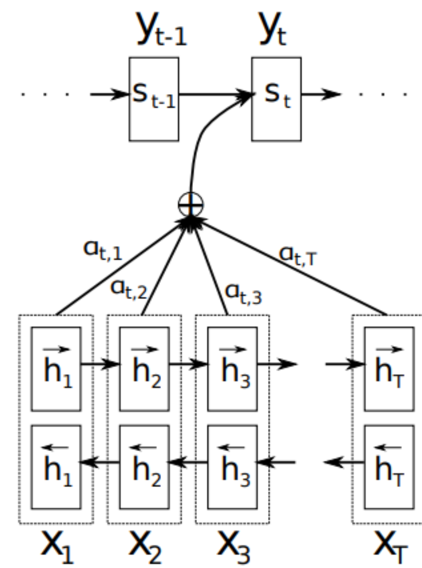
- RNNs over long sequences not to good at representing properties of the full sequence
  - **Biased** towards the end (or ends) of the sequence
  - Last hidden layer / context: A **bottleneck**!
- Attention averages over hidden sequence
  - $\boldsymbol{c} = \sum_j \alpha_j \boldsymbol{h}^{(j)}$ summary weighted average
  - $\alpha_j = \exp(e_j)/(\sum_{j'} \exp(e_{j'}))$ softmax
  - $e_j = f(\boldsymbol{h}^{(j)})$ focuses at
- E.g., key phrase in review

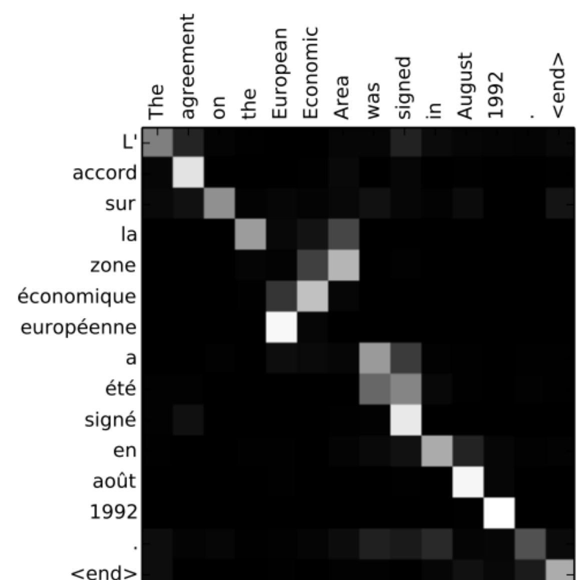


Deep Learning. Goodfellow, Bengio, Courville (2016). Ch 10

Repeated attention in Seq2seq models

- Consider multiple sequential outputs, $\boldsymbol{s}^{(i)}$
  - $\boldsymbol{c}_i = \sum_j \alpha_{ij} \boldsymbol{h}^{(j)}$
  - $\boldsymbol{s}^{(i)} = f(\boldsymbol{s}^{(i-1)}, y^{(i-1)}, \boldsymbol{c}_i)$
  - $\alpha_{ij} = \exp(e_{ij}) / (\sum_{j'} \exp(e_{ij'}))$
  - $e_{ij} = f(\boldsymbol{s}^{(i-1)}, \boldsymbol{h}^{(j)})$
- Avoids bottleneck, and uncovers meaningful structure

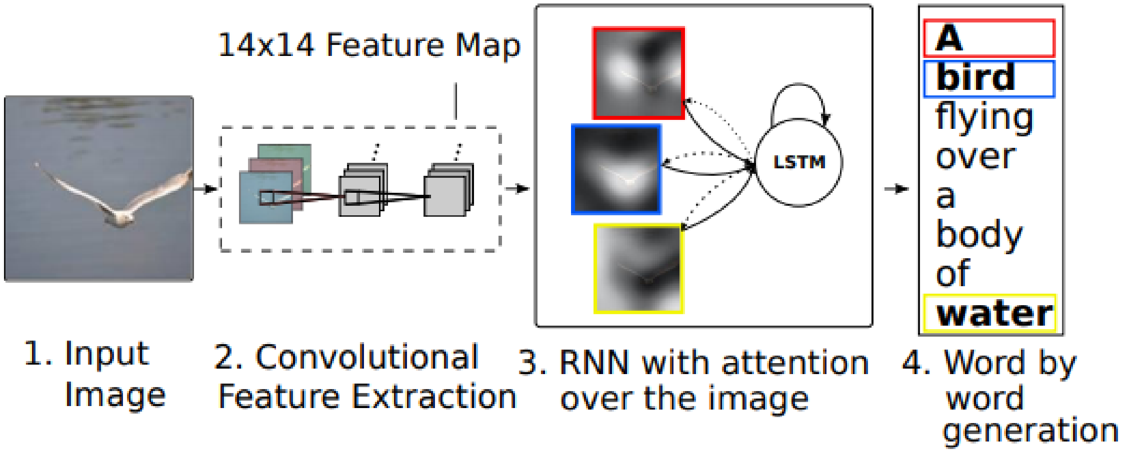


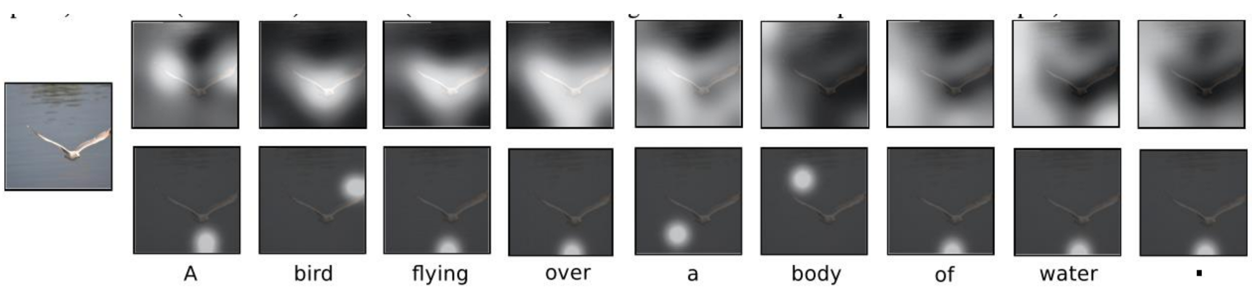


Neural Machine Translation by Jointly Learning to Align and Translate.
Bahdanau, Cho, Bengio, ICLR 2015

Attention in Vision

- Can attend to other representations, e.g., images
  - Attention over matrix input
  - Roves during generation of caption

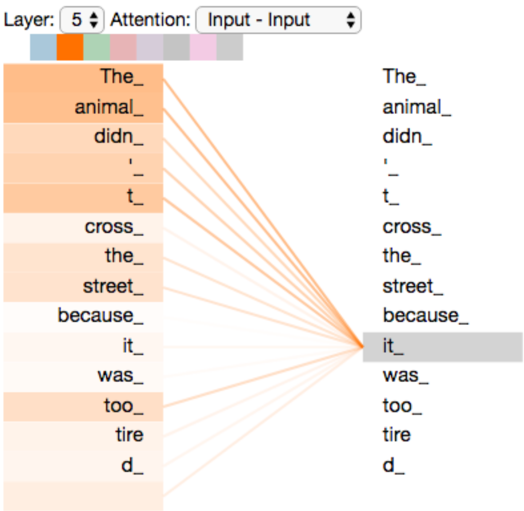




Show, Attend and Tell: Neural Image Caption Generation with Visual Attention, *Xu et al, ICML 2015*

Self-attention

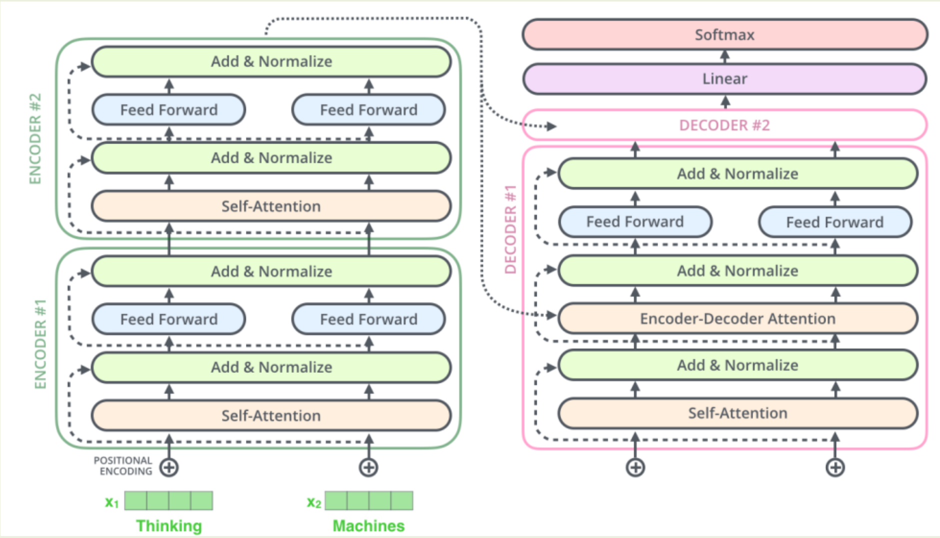
- Transformers use attention as means of representing sequences directly, instead of RNN
  - ∗ Representation of item *i* is based on attention to the rest of the sequence
  - ∗ Use item i as the query in attention against all items $j \neq i$
- Compared to RNNs
  - ∗ No explicit position information (add to each symbol position index)
  - ∗ Cheap: easily done in parallel



The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

Transformer

- The Transformer uses self-attention as its main step
  - Alongside residual, and normalization layers
  - Using multiple “attention heads”, and deep stacking
- Applied first to translation
  - Then raw text, e.g., BERT, RoBERTa, GPT
  - Highly scalable
  - Large performance gains over RNN models



The Illustrated Transformer, http://jalammar.github.io/illustrated-transformer/

This lecture

- Recurrent networks for modelling sequences
  - recurrent units
  - back-propagation through time
  - long-short term memory
- Transformers and attention

next: Cross-validation