Artificial Intelligence II: Deep learning methods

Dragoș Burileanu, Ana Neacșu & Vlad Vasilescu, Georgian Nicolae

Lecture 4: Recurrent Neural Networks

National University of Science and Technology POLITEHNICA Bucharest, Romania
BIOSINF Master Program

May 2024

Overview

Coping with sequential data

Spatialising the time : TDNN

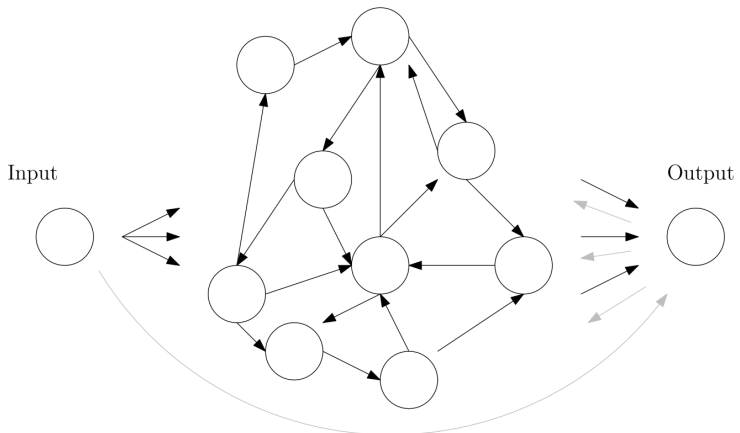
- The idea was introduced by Waibel et. al. in 1989 for phoneme recognition.

Some open questions

- What size to use for the time window?
- Should the history size be constant?
- Do we need the data over the whole time span ?
- How to share computations in time instead of using distinct weights per time instant?

Although Feedforward neural networks can still be efficient for processing sequential data, e.g. Gated ConvNet, Transformers, etc.

Recurrent Neural Network



Different types of weight matrices:

- W^{in} – input to hidden
- W^{out} – hidden to output
- W^{h} – hidden to hidden (applied multiple times)
- W^{back} – outputs to hidden

Different versions of RNNS

- if $W^{back} = 0 \rightarrow$ Elman networks
- if $W^{h} = 0 \rightarrow$ Jordan Networks
- if W^{h} is fixed $\rightarrow$ Echo state networks

Different paradigms formulation

Idea : Inputs and outputs can have variable size

Many to one

language models, sentiment analysis : multi class sequence classification

[‘this‘, ‘movie‘, ‘was‘, ‘awesome‘] $\mapsto$ 1
[‘you‘, ‘should‘, ‘was‘, ‘so‘ ‘bad‘ ‘it’s‘ ‘not‘ ‘worth‘ ‘the‘ ‘money‘] $\mapsto$ 0

Many to many

Neural machine Translation, e.g. RO $\rightarrow$ EN
“*Războiul afectează economia globală* ” $\mapsto$ “The war is affecting the global economy”

Different paradigms formulations

One to many

image captoning, language model with probabilistic sampling

How to train a RNN?

- Unrolled in time, RNN appear as very deep networks
- It is computationally expensive to do reverse-mode differentiation
- They are prone to problems like exploding or vanishing gradients
- Initially were difficult to train

Some ideas to improve:

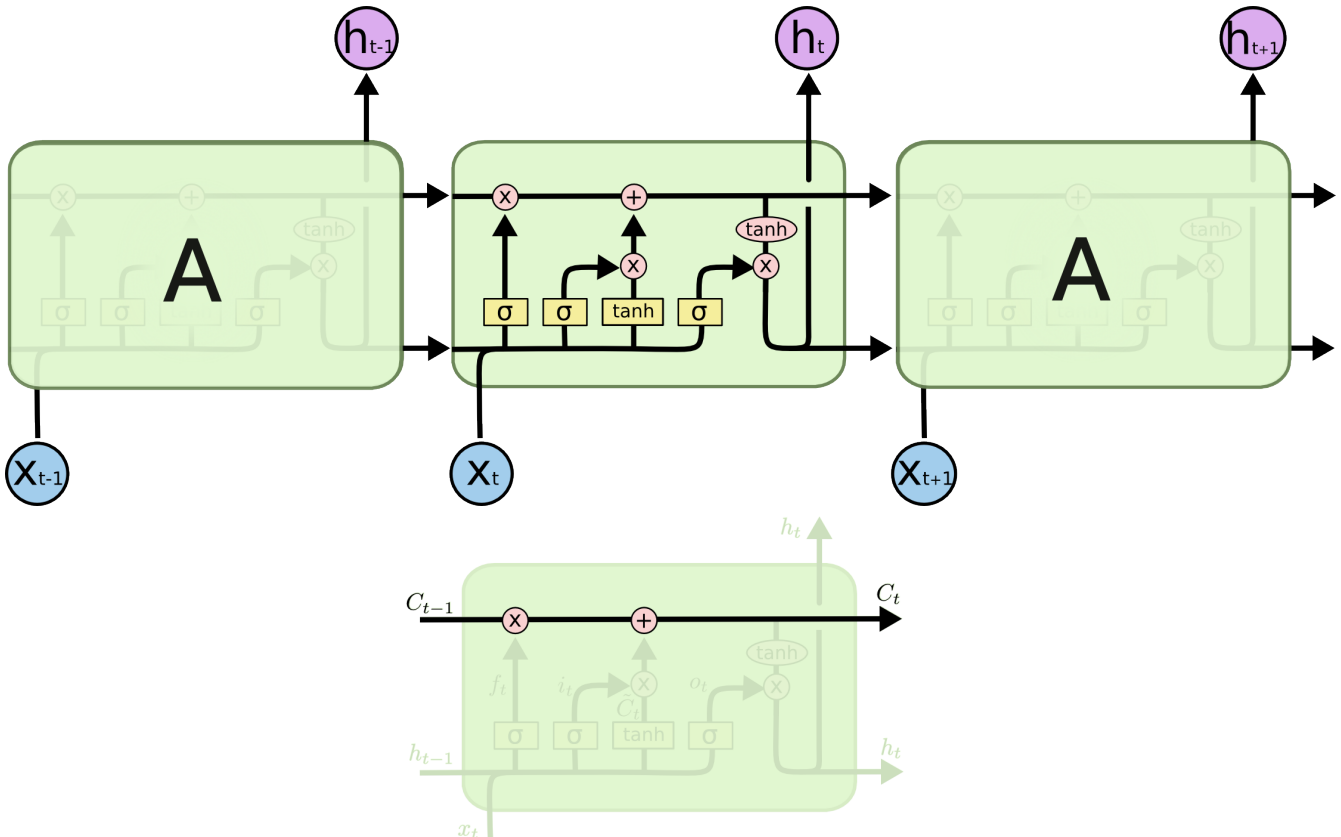
- Unfold in time the computational graph and do forward mode differentiation.
- Idea: Work on small windows to reduce the computational overhead

Using memory cells in Neural Networks

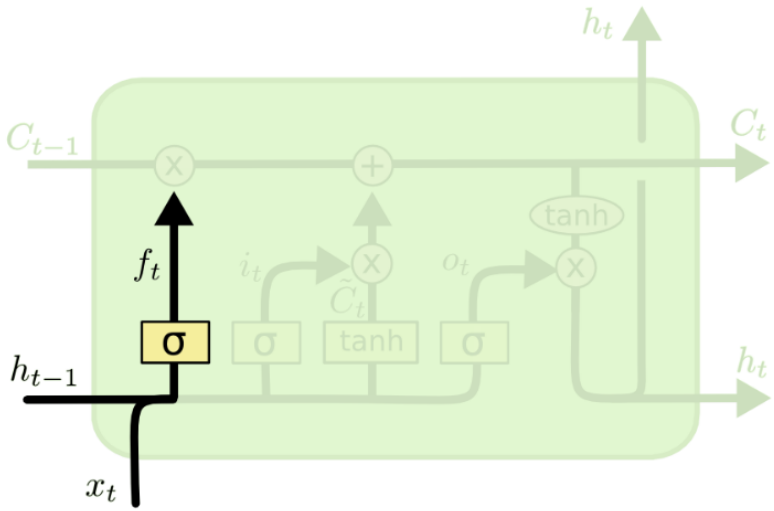
LSTM

Problem : RNNs have problems learning long range dependencies.

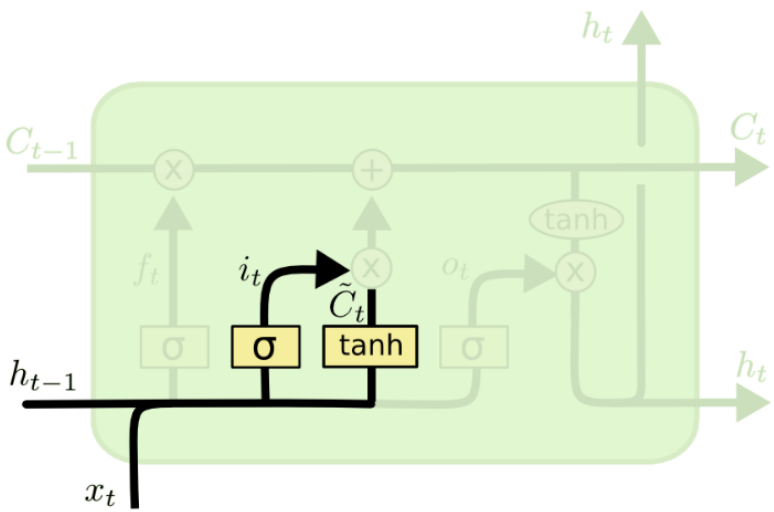
LSTM → Long-Short Term Memory (introduced by Schmidhuber în 1997) use memory cells to address this problem.



LSTM – step by step



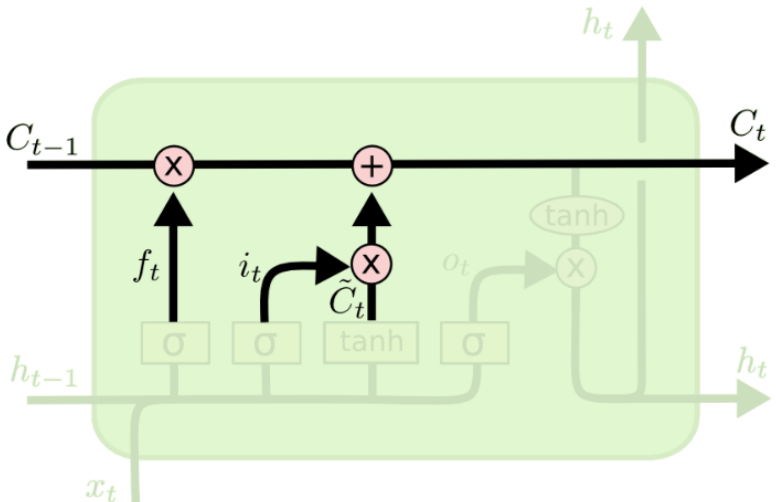
$$f_t = \sigma(W_f^x x_t + W_f^h h_{t-1} + b_f) \quad \in [0, 1]$$

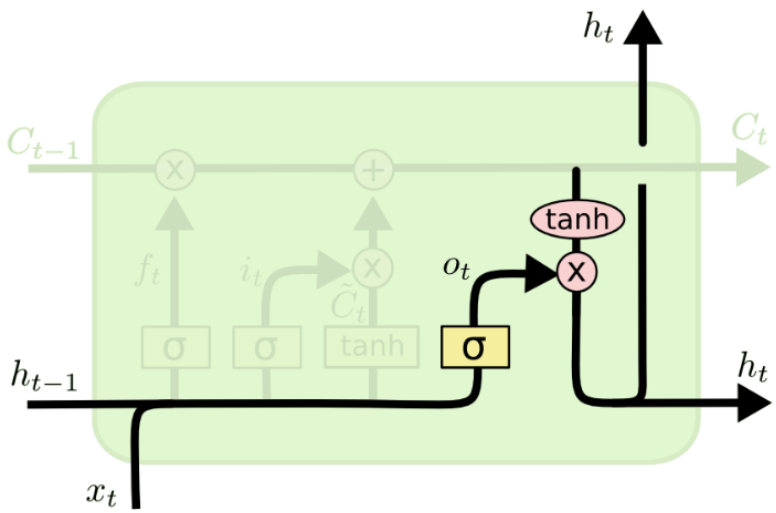


$$i_t = \sigma(W_i^x x_t + W_i^h h_{t-1} + b_i) \quad \in [0, 1]$$

$$\tilde{C}_t = \tanh(W_C^x x_t + W_C^h h_{t-1} + b_C)$$

LSTM – step by step





$$C_t = F_t \odot C_{t-1} + i_t \odot \tilde{C}_t$$

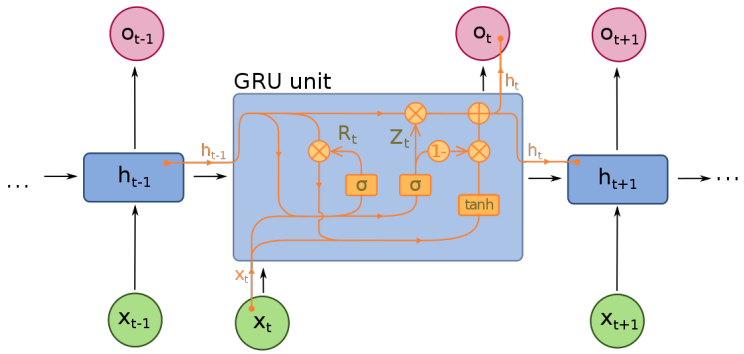
$$o_t = \sigma(W_o^x x_t + W_o^h h_{t-1} + b_o) \quad \in [0, 1]$$

$$h_t = o_t \odot \tanh(C_t)$$

- Next layers integrate h_t, not c_t
- If $f_t = 1$ and $i_t = 0 \rightarrow$ *constant error carrousel*
- Images and more info can be found here.

Gated Recurrent Units – GRU

Idea : a simplfied versions of LSTM, introduced by Cho et. al. in 2014.

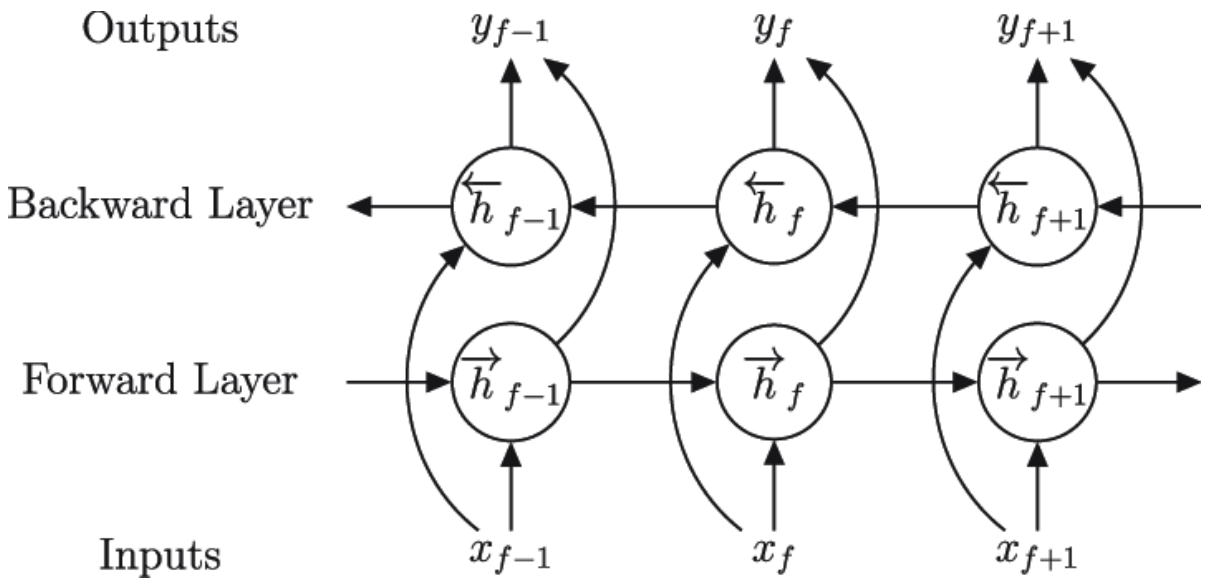


$$
\begin{aligned}
R_t &= \sigma(W_i^x x_t + W_i^h h_{t-1} + b_i) \\
Z_t &= \sigma(W_z^x x_t + W_z^h h_{t-1} + b_z) \\
n_t &= \tanh(W_n^x x_t + b_{nx} + R_t \odot (W_n^h h_{t-1} + b_{nh})) \\
h_t &= Z_t \odot h_{t-1} + (1 - Z_t) \odot n_t
\end{aligned}
$$

- If $Z_t = 1 \rightarrow h_t$ is not modified
- If $Zt = 0, R_t = 1 \rightarrow$ cell state is updated
- If $R_t = 0 \rightarrow$ hidden state reset

Bidirectional RNN/LSTM/GRU

Idea : Take into account both past and future contexts to classify current sample.



Stacked RNNs

Idea : Stack more cells to allow the layers to operate an increasing time-scales.
Downscale time: concatenate consecutive hidden stated before feeding those to the next layer.

Most notable works:

- Listen, Attend ans Spell.
- Machine translation with stacked unidirectionnal LSTMs (Seq2Seq
- Saddle point detection

Training strategies

Initialization

- LSTM : high forget bias $\rightarrow$ by default favor to remember everything
- Identity weight initialization – proposed by Le et. al.
- Orthonormal weight initialization – proposed by Henaff et. al., aims to keep: $\|Wh\|_2^2 = h^\top W^\top W h = h^\top h = \|h\|_2^2$

Training

- Layer Normalization – mean and variance are computed independently per sample over the whole layer

Regularization techniques

- gradient and activation clipping
- cannot use regular dropout – see (ZoneOut, rnnDrop)

Sequence Analysis

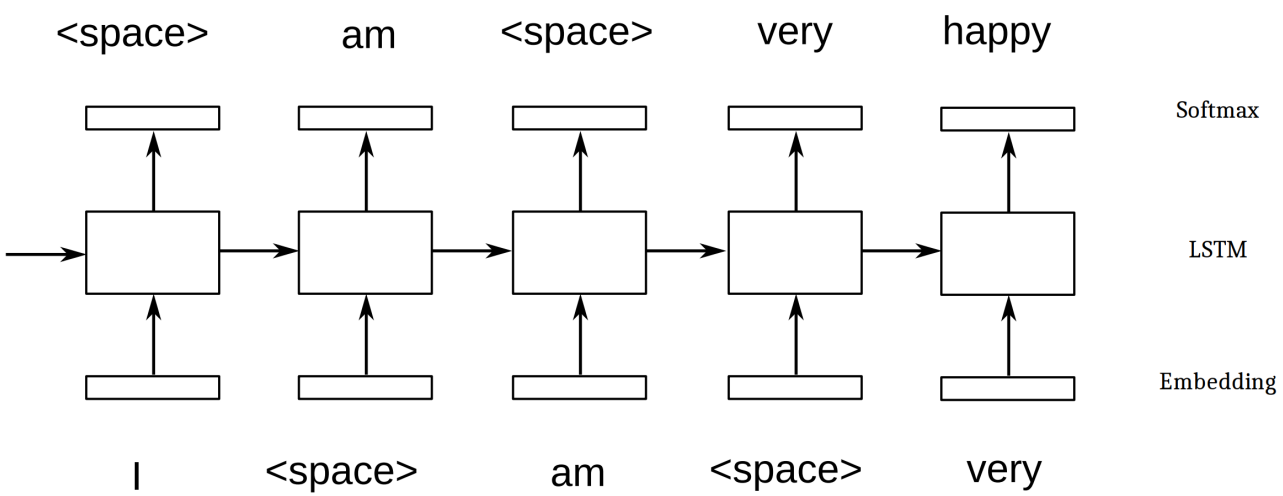
Possible problems

1. **One to One**
   - Example: Simple Classification
   - Solution: Vanilla RNN
2. **One to Many**
   - Example: Image Captioning
   - Solution: Show and Tell
3. **Many to One**
   - Example: Sentiment Analysis
   - Solution: LSTM for Sentiment Classification
4. **Many to Many**
   - Aligned sequences of identical sizes
     - Example: Video Classification
     - Solution: Time-Distributed CNN with RNN
   - Unaligned sequences of different sizes
     - Example: Machine Translation, Automatic Speech Recognition
     - Solution: Seq2Seq Models, Listen, Attend and Spell

Language model using charRNN

Idea : given some fixed-length words, predict the next word/character

- Predict $p(x_t|x_0, \ldots x_{t-1})$
- It's a many-to-many problem during training, but many-to-one during the inference
- Check The Unreasonable Effectiveness of Recurrent Neural Networks blog post.



Let us take an example

Example from CentraleSupelec course.

- Vocabulary : each element has a code associated.
  e.g. `'\t':0`, `'\n':1`, `'!':2`, `'.':3` ... `'A':25`, `'B':26`, ..., `'Z':51`, ...
- Dataset: windows containing 60 characters:
  e.g. Input: [4, 71, 67, 45, 36, 14, ... 100,65,36,75,89,57,70]
  Output : [71, 67, 45, 36, 14, ... 100,65,36,75,89,57,70, **63**]
- Architecture: Embedding, LSTM ($\times 2$), ReLU, Softmax
- Loss : cross-entropy averaged over batch_size $\times$ seq_len
- Training: ADAM(0.01)

The training process

- To sample from a LM, you start from a context e.g. [‘L’, ‘A’, ‘G’, ‘R’, ‘E’, ‘N’, ‘O’, ‘U’, ‘I’, ‘L’, ‘L’, ‘E’ ’ ‘]
- Note that here the text is generated character by character!

Initialization sample (200 characters):

LA GRENOUILLE y
-Ô)asZYc5[h+IÉë?8—>Y.bèqp;ÎzÇÇ<)!|f]Lt+«-u
XûoÜ:!ïgVùb|Ceü9ùÈ«à
6)ZàÀçBJi)X:ZÛdzxQ8PcviV]O]xPX,Înc.è’Pâs:X;ûfjBâ?X
*ç’ED’fSOl*Z(È’È1SnjàvPïLoUÊêàDgùO9z8eJûRYJ?Yg*
Uâp|jCbû—HxBràZBMZÛPCGuR’]ÀiÊÂSBF4D),û

After 30 epchs (200 characters):

LA GRENOUILLE ET MOURE ET LA RENARDIER
Quel Grâce tout mon ambassade est pris.
L’un pourtant rare,
D’une première
Qu’à partout tout en mon nommée et quelques fleuris ;
Vous n’oserions les Fermerois, les heurs la

After 50 epochs (200 characters):

LA GRENOUILLE D’INDÉTES
[Phèdre]
Tout faire force belle, commune,
Et des arts qui, derris vôtre gouverne a rond d’une partage conclut sous besort qu’il plaît du lui dit Portune comme un Heurant enlever bien homme,

More details about this in Joel Legrand’s course.

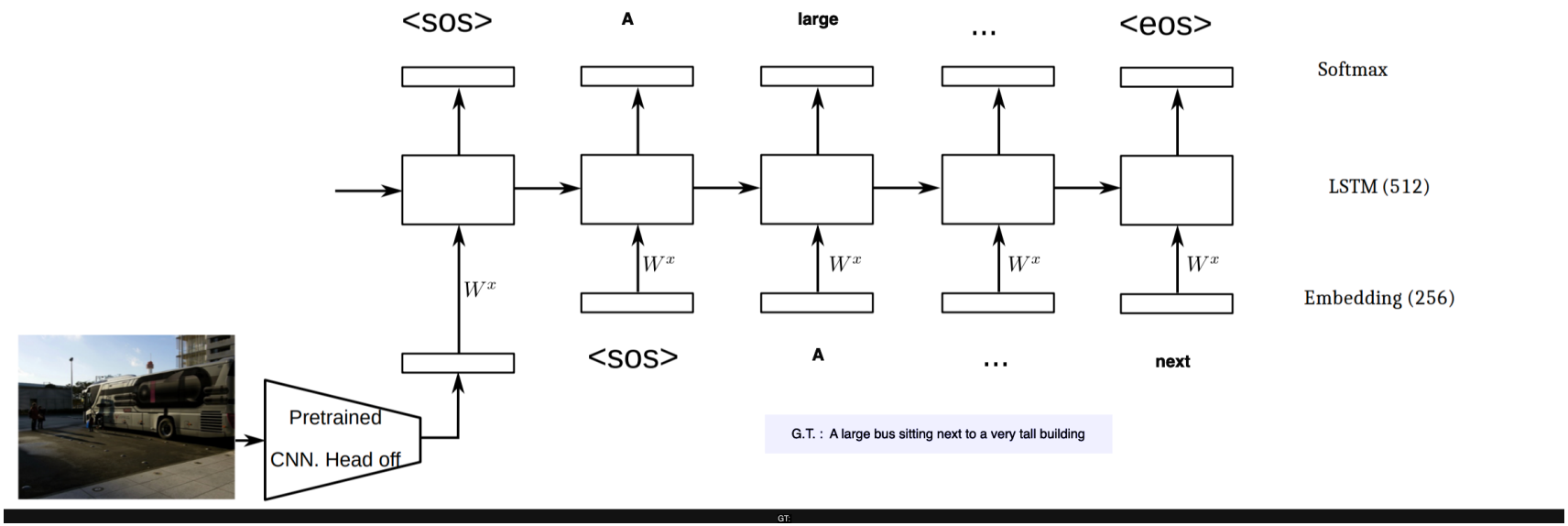
Image captioning

Idea : Based on an input image, generate a short description of it.

- One of the first works was introduced by Erhan et. al.

Show and Tell

Idea : introduce an end-to-end neural network model that combines a convolutional neural network (CNN) for image feature extraction with a recurrent neural network (RNN) for sequence generation.



Minimization problem:

$$-log(P(S_0, S_1, S_2, \dots S_T|I, \theta)) = -\sum_j log(P(S_j|S_0, \dots S_{j-1}, I, \theta))$$

Show and Tell

Training :

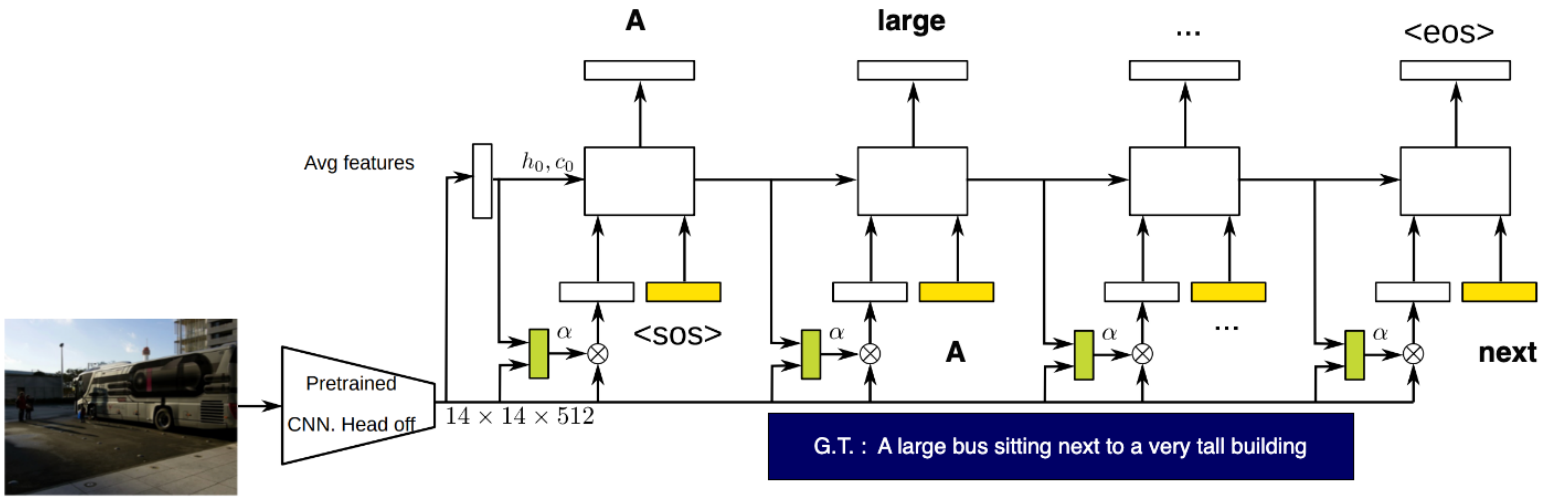
- Trained on COCO Caption dataset
- The CNN part was pretrained on Image-Net
- Words embeddings randomly initialized (The also tried a pre-trained version)
- 512 - LSTM cells and embeddings
- Training the LSTM with frozen CNN then finetuning everything, using SGD (0.01)
- End-to-end training fails
- Scheduled sampling

Inference :

- Use beam-search for decoding – more on that later

Attention based show and tell

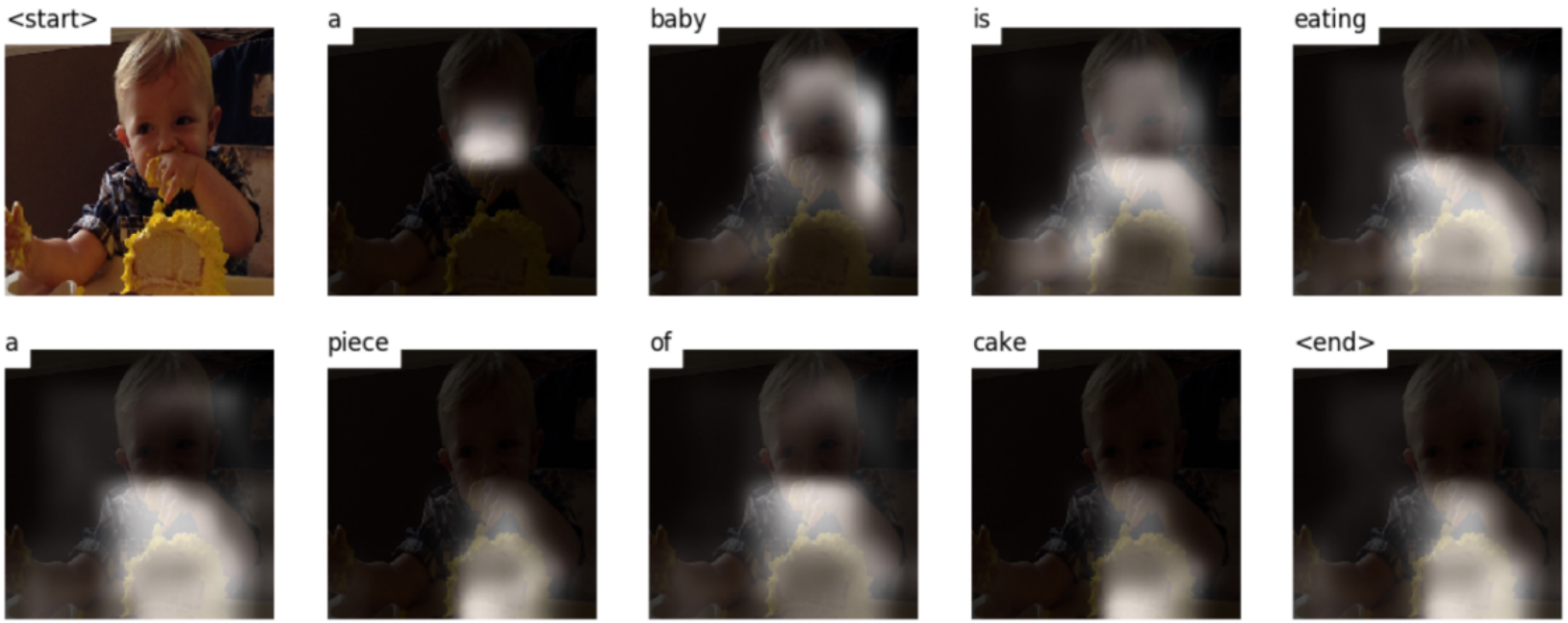
Idea : Introduces an attention mechanism into the image captioning task, allowing the model to focus on different parts of the image when generating captions.
Introduced by Xu et. al.



Stochastic Attention

- $\sum_i \alpha_{t,i} = 1 \rightarrow$ the coefficients are normalized by design
- force the model to pay attentions to all locations by using a regularization term, i.e. $\lambda \sum_l = (1 - \sum_t \alpha_{t,l})^2$, where l is the location.

Attention example



Results

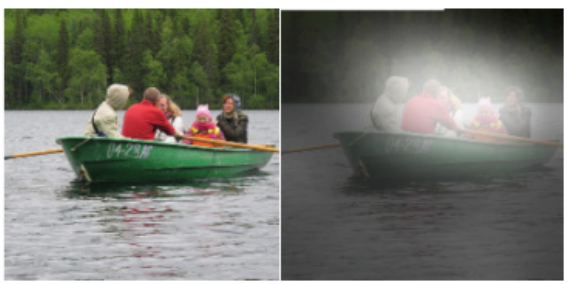
Results taken from Xu et. al.

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.

A little girl sitting on a bed with a teddy bear.



A group of people sitting on a boat in the water.

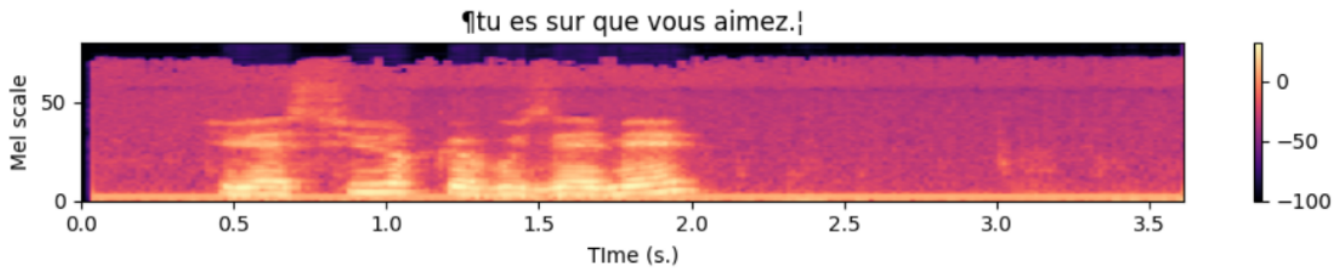
A giraffe standing in a forest with trees in the background.

Unalignment issue

Problem : How to handle Input/Output Sequences of **variable sizes** and **misaligned lengths**?

This is the case in applications such as:

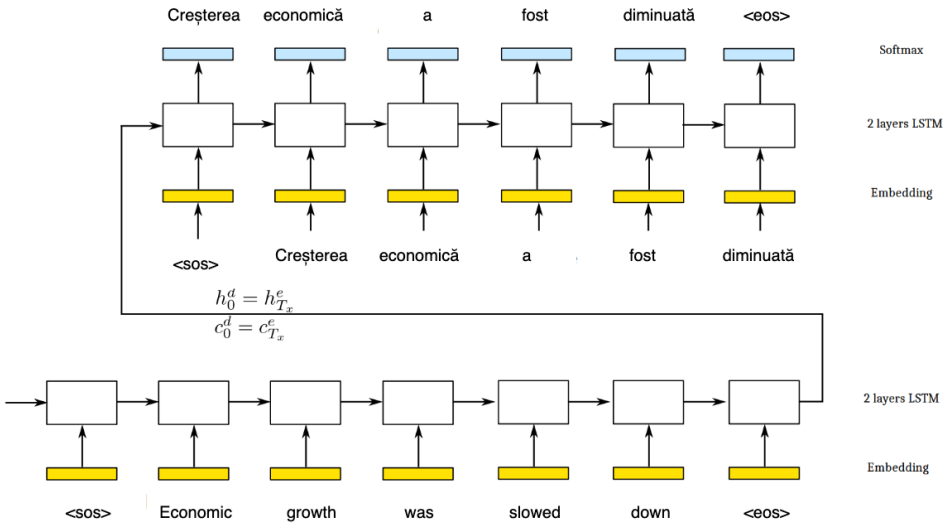
- Automatic Speech Recognition (ASR)
  - Input: Mel Spectrogram
  - Output: associated text



- Machine Translation (MT)
  - Input: *The economic growth was slowed down.*
  - Output:
    Creșterea economică a fost diminuată.
    La croissance économique a été ralentie.

Sequence to sequence using Encoder-Decoder Architecture

Idea : Encode the input sequence into a hidden state and decode the output sequence from it. Introduced in Vinyals et. al. for Neural Machine Translation.



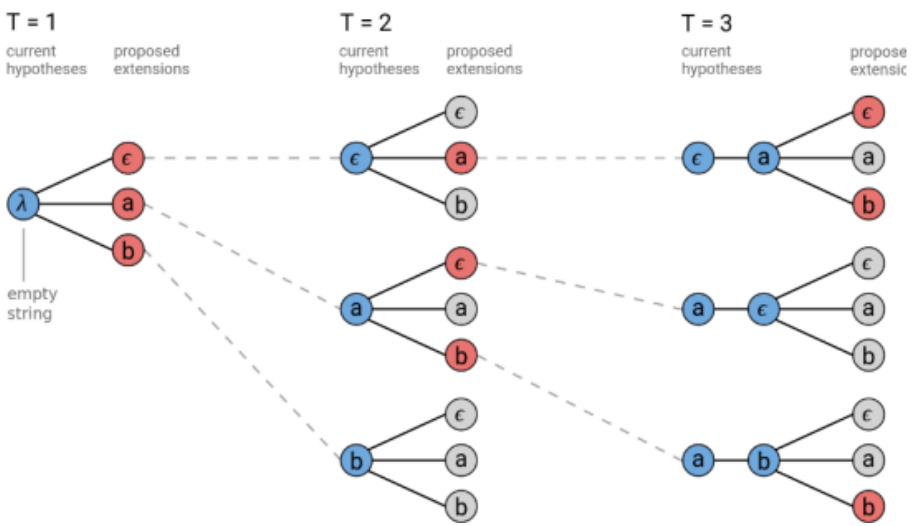
Training startegy

- Beam search decoding
- Input fed in reverse order
- Scheduled sampling
- Professor forcing
- Check this blog

Beam Search Decoding

$$p(y|x) = p(y_0|x,\theta)\prod_t p(y_t|y_0,/dots,y_{t-1}x\theta)$$

- minimize $p(y|x) \rightarrow$ obtain the translation with the highest probability
- the probability distribution over the labels is dependent on the previously generated label $\rightarrow$ approximate a search by maintaining a set of M candidates
- check this for more details.



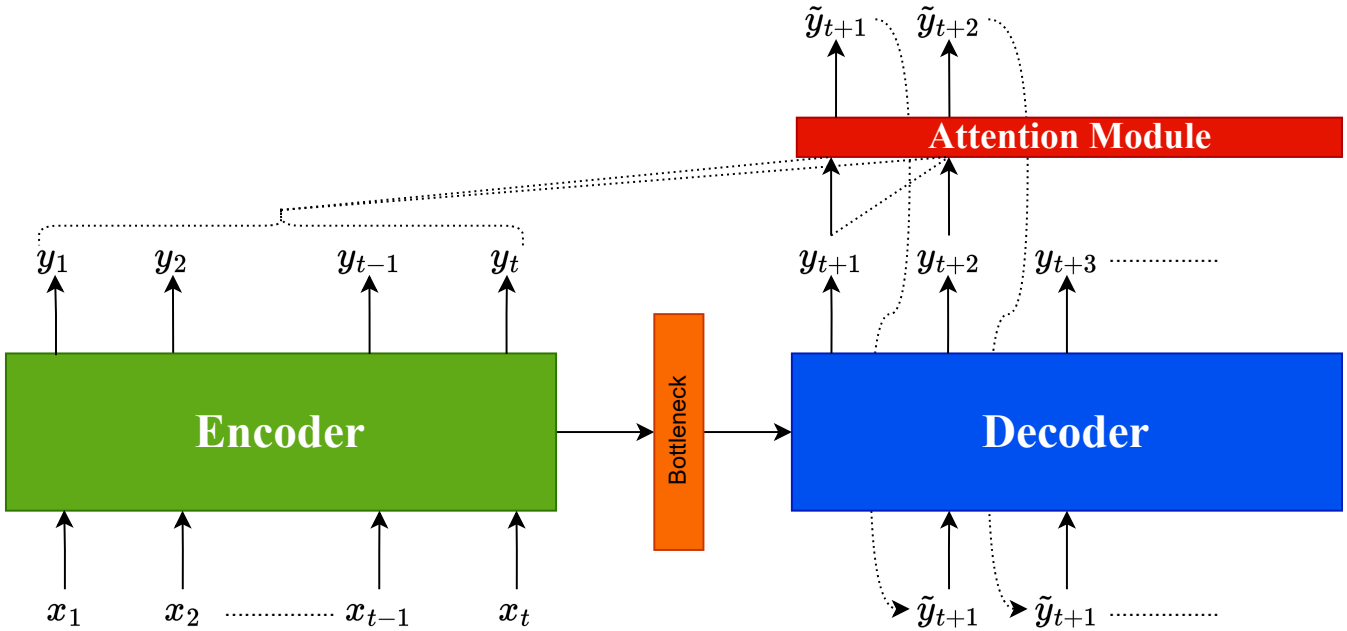
Attention in Neural Networks

Why Attention?

What if we are dealing with very long sequences?

- The RNN encoder would not be capable to compress all important information into the final hidden state
- Hence, the RNN decoder won't be able to produce meaningful outputs

Idea : look at all previous encoder and decoder outputs before predicting the next item



Self-Attention

Given a temporal sequence $\{x_i\}_{i=1}^{T}$, $T \geq 2$, $x_i \in \mathbb{R}^d$, represented in matrix form as $X \in \mathbb{R}^{T \times d}$, self-attention produces another sequence $\widetilde{X} \in \mathbb{R}^{T \times d}$ as follows:

1. Compute Keys, Queries, Values:

$$K = XW^K \qquad Q = XW^Q \qquad V = XW^V$$

$$W^K, W^Q \in \mathbb{R}^{d \times d'} \qquad W^V \in \mathbb{R}^{d \times d}$$

2. Compute similarity matrix $A \in [0,1]^{T \times T}$ with:

$$A_{i,:} = \text{Softmax}\left(\frac{(QK^\top)_{i,:}}{\sqrt{d'}}\right) \in \mathbb{R}^T \text{(row-wise)}, \quad \forall i \in \{1, \ldots, T\}$$

3. Compute output sequence as:

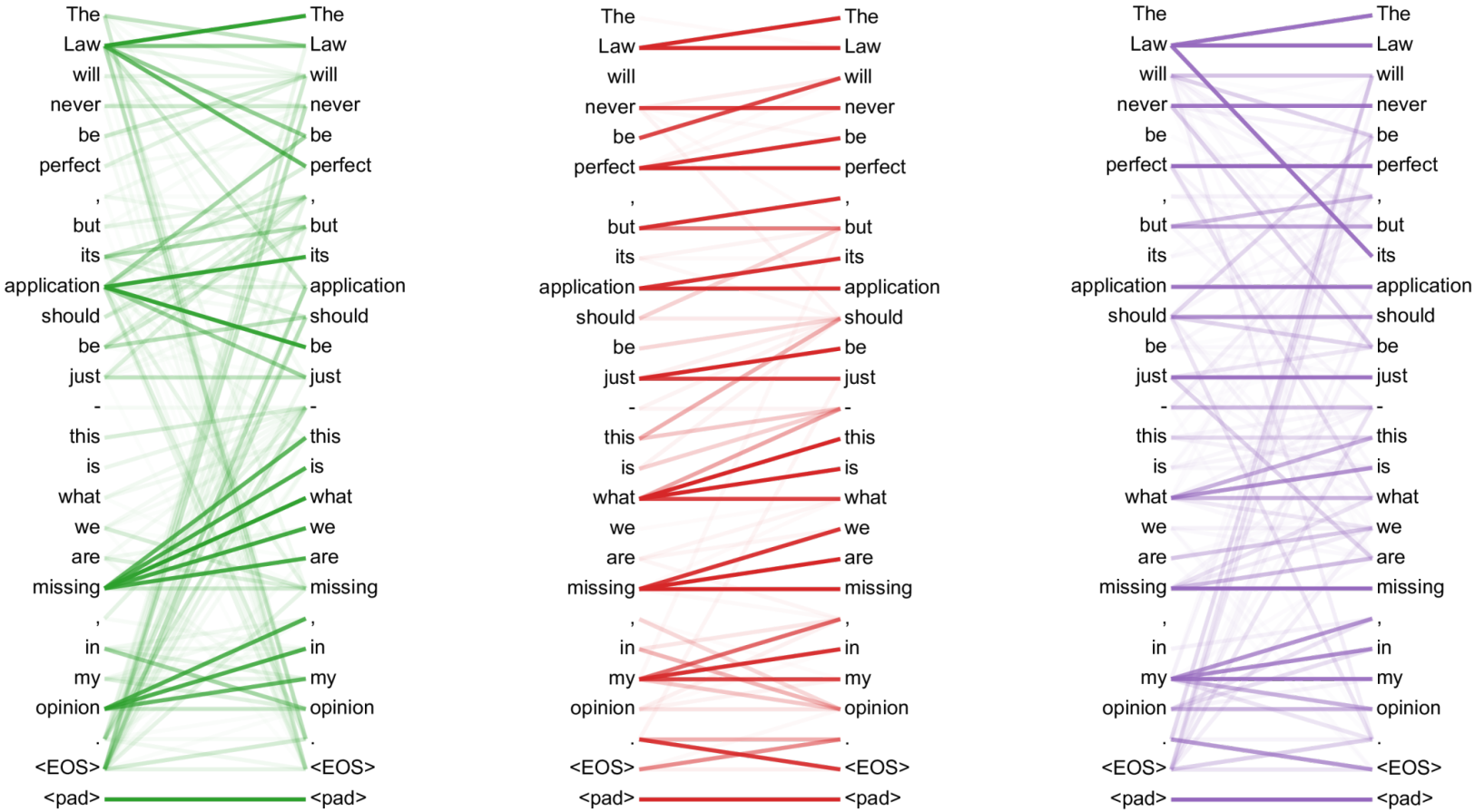
$$\widetilde{X} = AV$$

In this case, the temporal items of $\widetilde{X}$ roughly correspond to convex combinations of the initial x_i.

Self-Attention

- Dividing by $\sqrt{d'}$ helps with stabilizing training, reducing the magnitude of attention scores
- The previous mechanism describes the inners of a *single-headed* self-attention
- For *multi-head* self-attention, with H heads:
  1. For each head $i \in \{1, \ldots, H\}$, compute triplets:
  $$(K_i, Q_i, V_i), \quad \text{using} \quad (W_i^K, W_i^Q, W_i^V)$$
  2. For each head i compute similarity matrix $A^{(i)}$, and output $\widetilde{X}_i = A^{(i)} V_i$
  3. Concatenate all $\widetilde{X}_i \in \mathbb{R}^{T \times d}$ column-wise into
  $$\widetilde{X}_c = [\widetilde{X}_1, \ldots \widetilde{X}_H] \in \mathbb{R}^{T \times (d \cdot H)}$$
  4. Compute final output:
  $$\widetilde{X} = \widetilde{X}_c W^O \in \mathbb{R}^{T \times d}, \quad \text{with} \quad W^O \in \mathbb{R}^{(d \cdot H) \times d}$$

Self-Attention



Example of similarities learned by different self-attention heads, in the same layer. Illustrations from Attention is All you Need

Cross-Attention

What if we want to represent one sequence in terms of another?

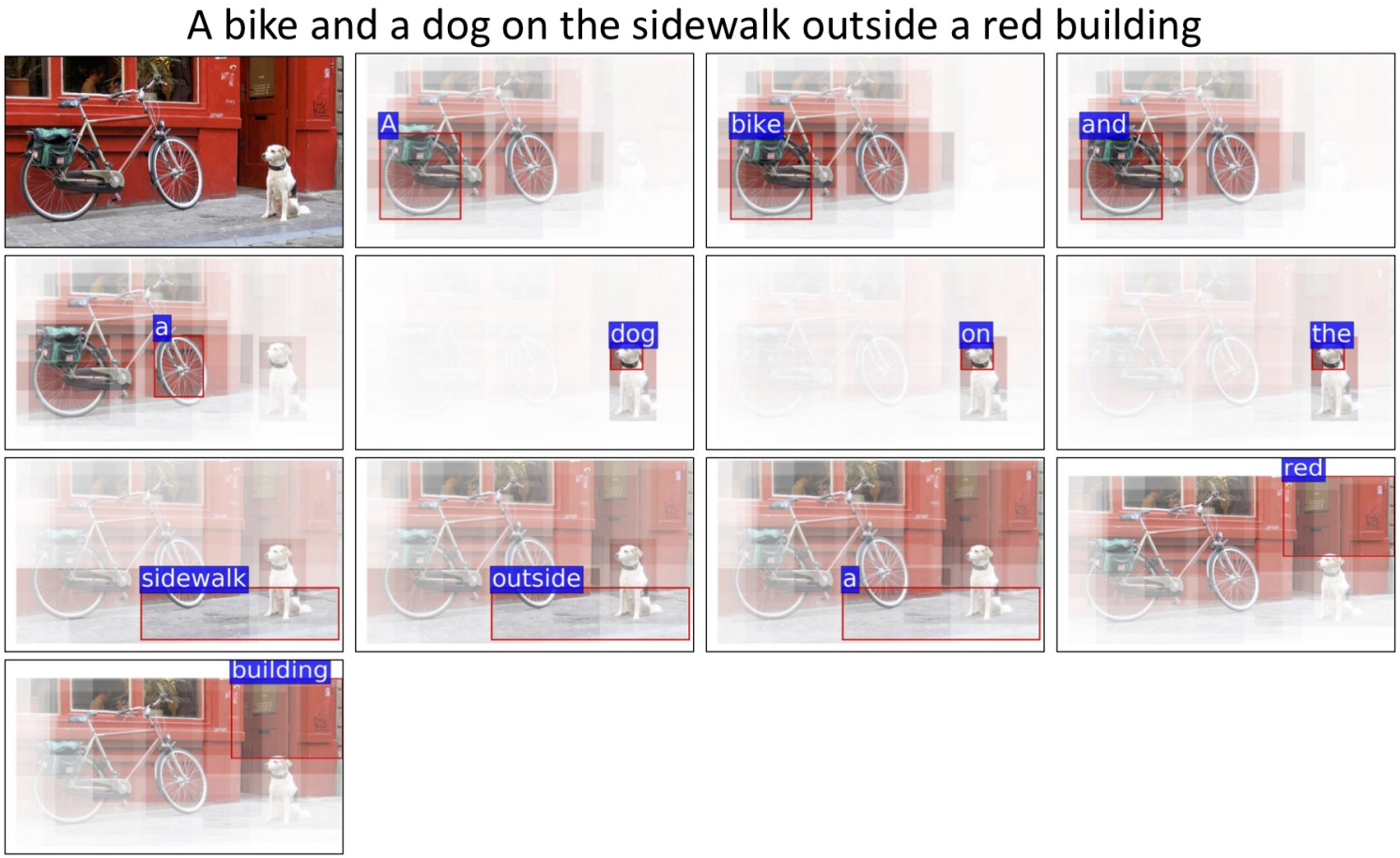
Given a sequence $Y \in \mathbb{R}^{T_Y \times d}$ which we want to represent in terms of sequence $X \in \mathbb{R}^{T_X \times d}$, $T_Y, T_X \geq 2$, cross-attention works as follows:

1. K_i, V_i are computed using X, Q_i is computed using Y
2. $A^{(i)}$ will now have size (T_Y, T_X)
3. Output is computed similarly as:

$$\widetilde{Y}_i = A^{(i)} V_i = A^{(i)} X W_i^V$$

Therefore, each element of $\widetilde{Y}_i$ will be represented as a convex combination of linearly projected elements of X.
For each element $\widetilde{y}_i$, one can use $A^{(i)}$ to retrive the most similar $\widetilde{x}_i$.

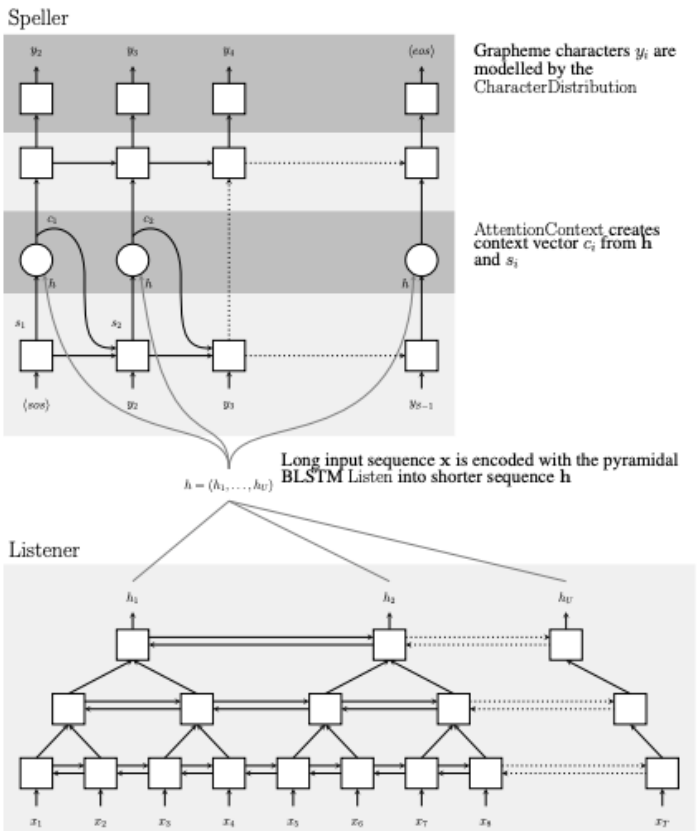
Cross-Attention



Alignment learned between a sequence of words and image regions (represented as a sequence of patches). Example from Stacked Cross Attention for Image-Text Matching

Applications of Attention

Listen, Attend and Spell → use a seq2seq encoder-decoder + attention for ASR. Introduced by Chan et. al.



- Input: MEL-spectrogram
- Output: The associated text
- 5 bi-LSTM encoder - 2 LSTM decoder
- scheduled sampling
- 4-head attention
- both queries and keys are computed using a simple MLP.