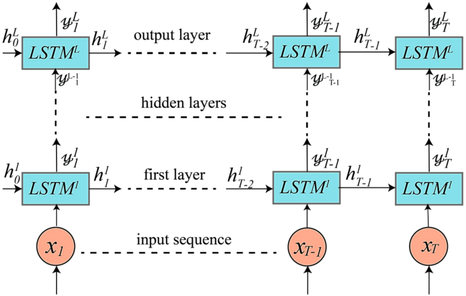
EE3-25: Deep Learning

Krystian Mikolajczyk and Carlo Caliberto

Sara Iodice, Axel Barroso Laguna and Adrian Lopez Rodriguez

Department of Electrical and Electronic Engineering

Imperial College London

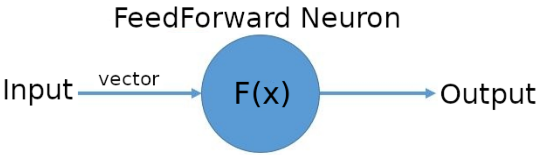


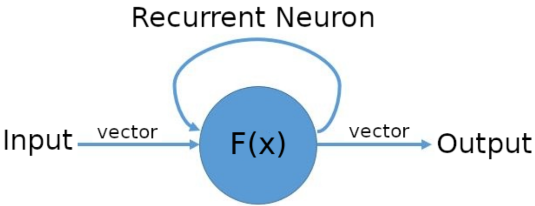
Recurrent Neural Networks

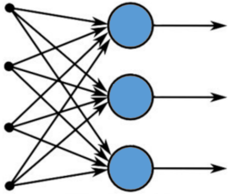
- Recurrent Neural Networks
  - Word embedding
- RNN Unit
- LSTM Unit
- GRU Unit
- Architectures & applications

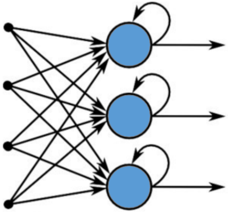
Recurrent Neural Networks

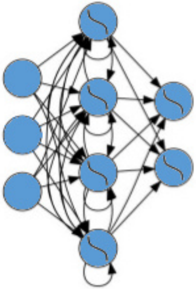
- Feed-Forward Networks vs. Recurrent Neural Networks






Feed-Forward Neural Network

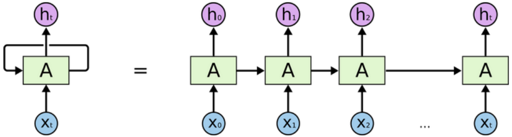

Recurrent Neural Network



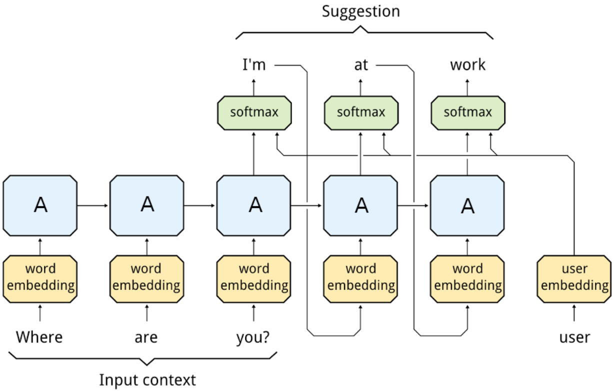
Recurrent Neural Networks

- Chain like structure of RNN suitable for sequences and lists

Speech, language, text, temporal data (audio, video)

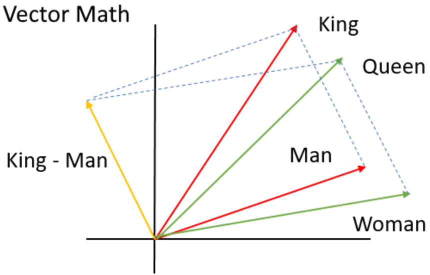


- Dialog generation example
  - Input question, output answer
  - Semantic meaning needs numerical embedding



RNN - word embeddings

- **Word embedding** is a real number, vector (numerical) representation of a word or text to allowing to apply mathematical tools
  - words with similar meaning will have vector representations that are close together in the embedding space
  - the goal is to capture some sort of relationship in that space e.g. meaning, morphology, context, etc



RNN - word embeddings

- One-Hot Encoding (Count Vectorizing) with a vocabulary
  - Binary indicator of word persence/absence from the text
- TF-IDF vectors, Bag of Words, instead of binary presence count, words are represented by their term frequency multiplied by their inverse document frequency
  - words that occur everywhere are given little weighting (i.e. 'the', 'and')
- A co-occurrence matrix is a large matrix $V \times V$. If words occur together, they are marked with a positive count, otherwise 0.

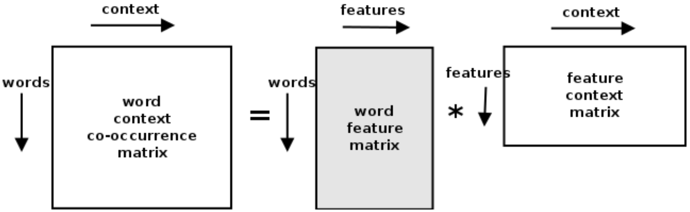
$$
\begin{aligned}
\texttt{Rome} &= [\overset{\text{Rome}}{1},\ \overset{\text{Paris}}{0},\ 0,\ 0,\ 0,\ 0,\ \ldots,\ \overset{\text{word V}}{0}] \\
\texttt{Paris} &= [0,\ 1,\ 0,\ 0,\ 0,\ 0,\ \ldots,\ 0] \\
\texttt{Italy} &= [0,\ 0,\ 1,\ 0,\ 0,\ 0,\ \ldots,\ 0] \\
\texttt{France} &= [0,\ 0,\ 0,\ 1,\ 0,\ 0,\ \ldots,\ 0]
\end{aligned}
$$

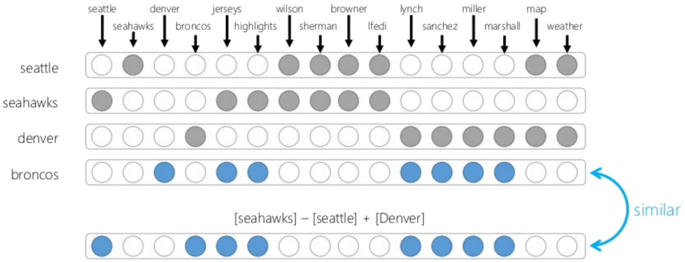
$$
X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array}
\begin{array}{c}
\begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\
\begin{bmatrix}
0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\
2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 0
\end{bmatrix}
\end{array}
$$

RNN - word embeddings

- Word2Vec, Doc2Vec is a neural probabilistic model
  - Shallow network trained on large text data
  - Captures semantic relations
  - One-word contex, multi-word context

- GloVe is an extension of Word2Vec





RNN Unit

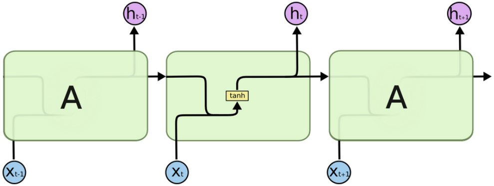
- inputs $x_t \in \mathbb{R}^{N_{in}}$ and outputs $y_t \in \mathbb{R}^{N_{out}}$

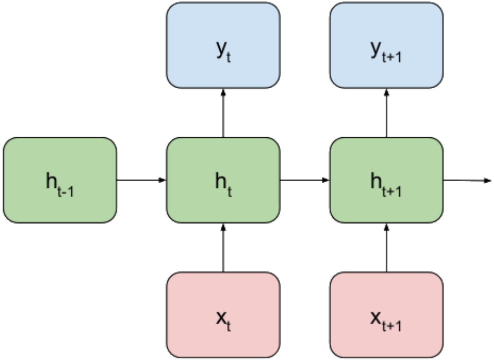
  $h_t = \Theta(W^{(hh)} h_{t-1} + W^{(xh)} x_t + b_h)$

  $y_t = W^{(hy)} h_t + b_y$

  - Θ is an activation function,
  - $h_t \in \mathbb{R}^{N_h}$ is the hidden state,
  - $W^{(hh)} \in \mathbb{R}^{N_h \times N_h}$,
  - $W^{(xh)} \in \mathbb{R}^{N_h \times N_{in}}$,
  - $W^{(hy)} \in \mathbb{R}^{N_{out} \times N_h}$,
  - $b_h \in \mathbb{R}^{N_h}$ and $b_y \in \mathbb{R}^{N_{out}}$

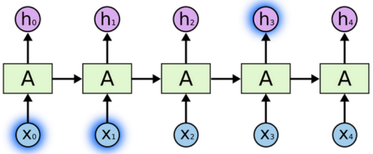
- The output could also be passed through as input to the next unit e.g. $x_{t+1} = h_t$,
  - to train this type of RNN, we shift the sequence by one to obtain the target sequence



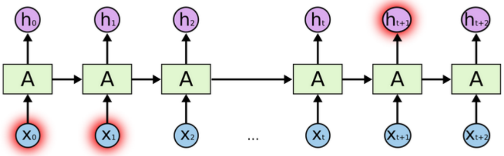


RNN Unit

- RNNs use a distributed hidden state that allows them to store sequence information
- RNN as a layered, feedforward network with shared weights
- Training with backpropagation algorithm with the forward and backward passes stepping through time
- At the backward pass we add the derivatives at all the different times for each weight (weight sharing)



- Models well short term dependencies
  - A single tanh layer

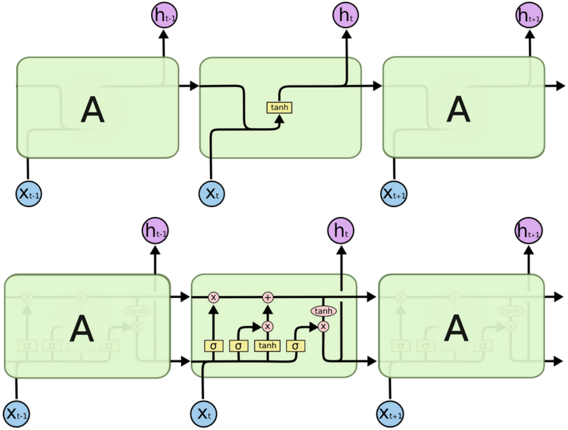


- Vanishing gradients with long term dependencies (long sequences)

$$\delta^{(l)} = \left(\prod_{k=l}^{L-1} \Theta'(s^{(k)})(W^{(k)})^T\right) \Theta'(s^{(L)}) \nabla_{x^{(L)}} \mathcal{L}$$

Long Short Term Memory

- LSTM was esigned to avoid long term dependency problems i.e. remembers information for long periods
  - Multiple layers, three sigmoid gates (forget, input/update, output) that protect and control the cell state by letting some information through with $[0, 1]$
  - tanh layer pushes information to range $[-1,] >$



LSTM Unit

- Input: concatenated h_{t-1}, x_t
- Forget gate f_t decides what and how much old to forget

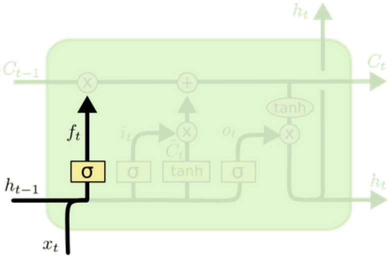
  $f_t = \Theta(W_f[h_{t-1}, x_t] + b_f)$

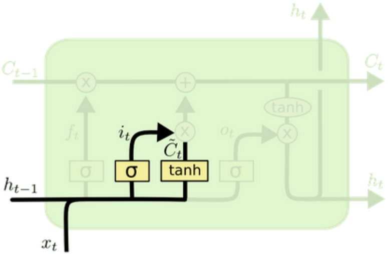
- Update gate i_t decides what and how much new to remember

  $i_t = \Theta(W_i[h_{t-1}, x_t] + b_i)$

- Internal cell state

  $\hat{C}_t = \tanh(W_C[h_{t-1}, x_t] + b_C)$





LSTM Unit

- Internal cell state C_t that allows the unit to store and retain information, which results from old C_{t-1} and new $\hat{C}_t$

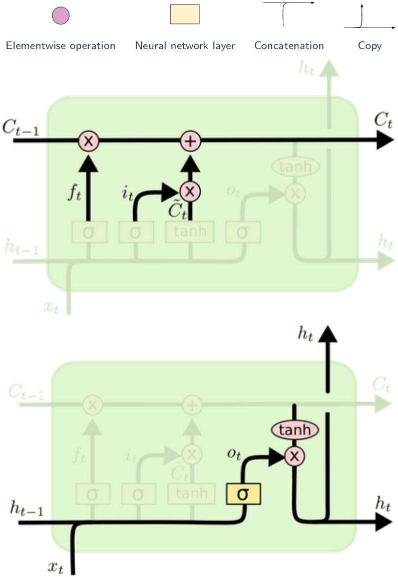
  $C_t = f_t \otimes C_{t-1} \oplus i_t \otimes \hat{C}_t$

  - $\otimes$, $\oplus$ - elementwise operations

- Output gate decides what and how much to output using o_t

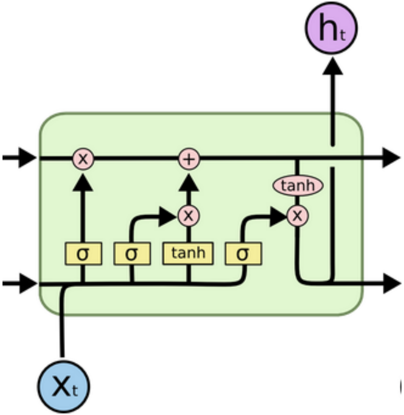
  $o_t = \Theta(W_o[h_{t-1}, x_t] + b_o)$

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



LSTM Unit

- Advantages of a typical RNN architecture
  - Possibility of processing input of any length
  - Model size not increasing with size of input
  - Computation takes into account historical information
  - Weights can be shared across time
- Drawbacks
  - Computation can be slow
  - Difficulty of accessing information from a long time ago
  - Cannot consider any future input for the current state
  - Many internal parameters

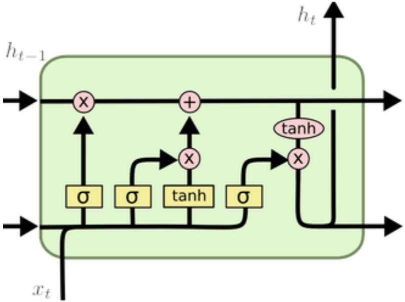


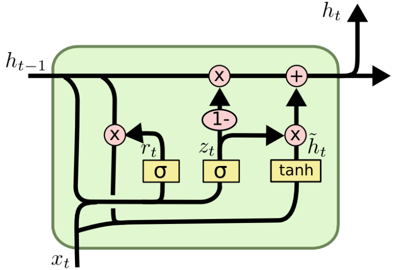
Gated Recurrent Unit

- GRU adaptively reset and update memory, r_t and z_t gates are similar to the forget and the input gate of the LSTM

$$z_t = \Theta(W_z[h_{t-1}, x_t] + b_z)$$
$$r_t = \Theta(W_r[h_{t-1}, x_t] + b_r)$$
$$\hat{h}_t = \tanh(W[r_t \otimes h_{t-1}, x_t] + b)$$
$$h_t = (1 - z_t) \otimes h_{t-1} + z_t \otimes \hat{h}_t$$
$$o_t = h_t$$

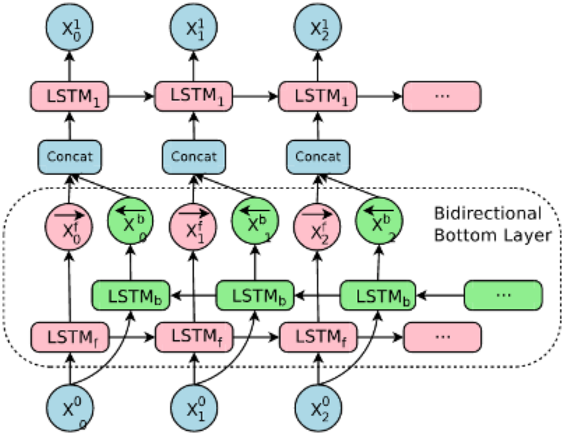
- Fully exposes its memory at each time step and has no separate memory cells, which is differently to the LSTM
- The output combines its last state and the new state.
- LSTM and GRU outperform the traditional tanh-unit but there is no significant performance difference between the LSTM and GRU.



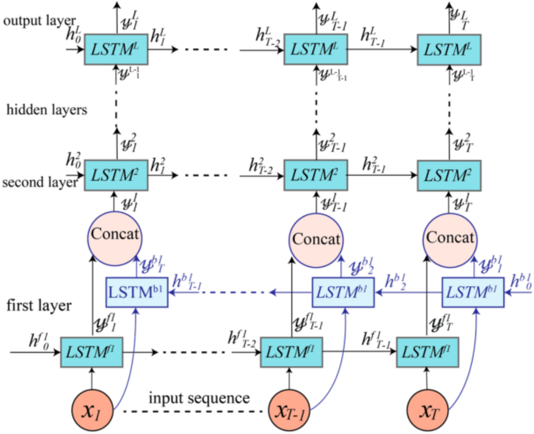


LSTM architectures

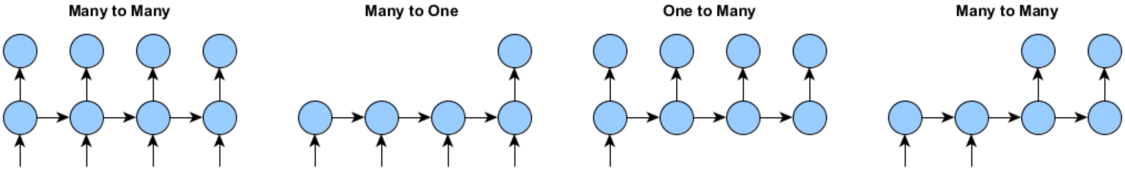
- Bidirectional (BRNN) consider past and future data
  - Sequence to sequence learning.



- Deep (DRNN) for large data e.g. video
  - Many hidden layers

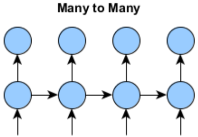


LSTM architectures

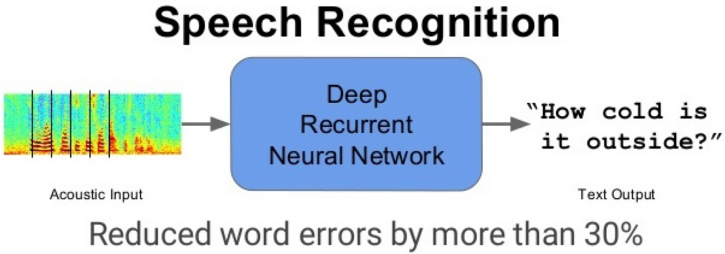


- Many to many I - name entity recognition, audio/video labelling
- Many to one - Sentiment classification, audio classification
- One to many - audio, text generation (music, captioning)
- Many to many II- machine translation

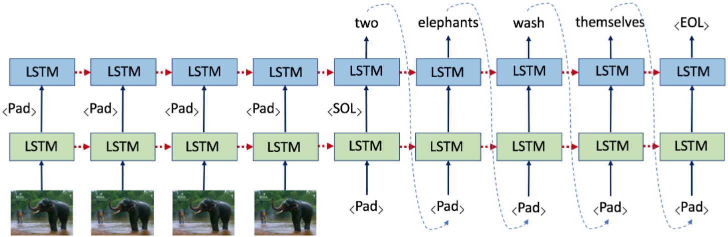
LSTM architectures - many to many



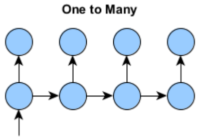
- Speech recognition
  - Audio sequence input and text sequence output



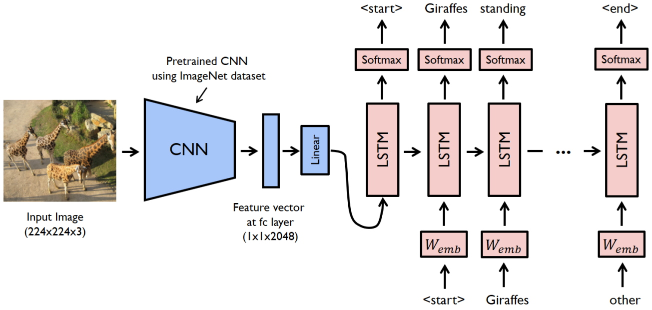
- Video annotation
  - Video input and text output



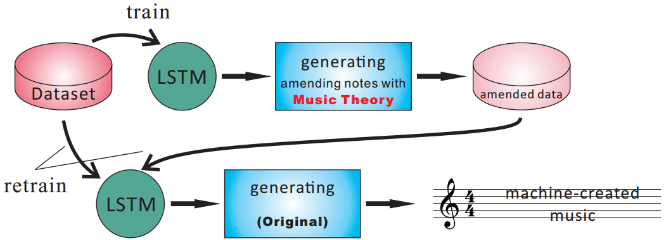
LSTM architectures - one to many



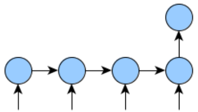
- Image captioning
  - Generating image description



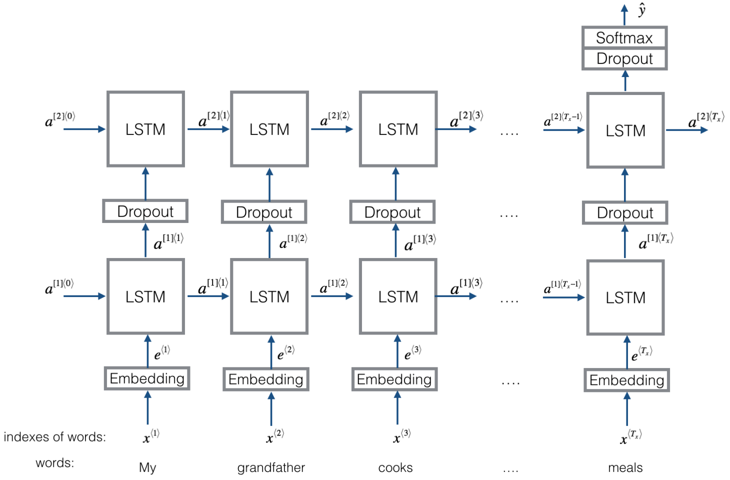
- Audio generation
  - Generating original music by genre



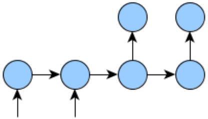
LSTM architectures - many to one



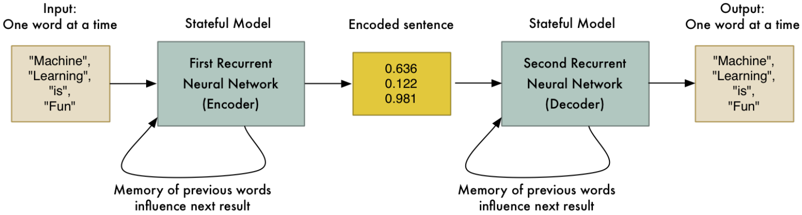
- Sentiment analysis
- A sentence (statement, comment, post) is input to multi layer LSTM that outputs a probability between positive and negative sentiment
  - note Dropout layer for regularisation

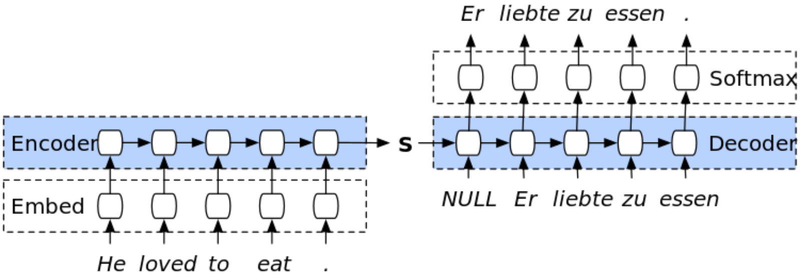


LSTM architectures - many to many



Many to Many



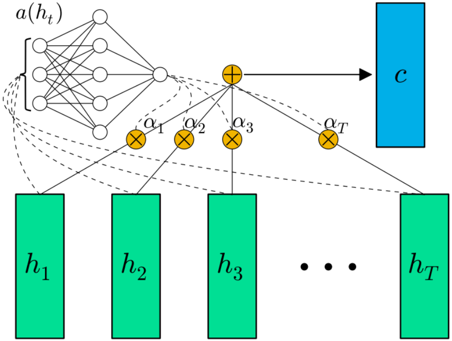


- The encoder RNN needs to encode the entire source sentence into the final hidden state
- For long sentences, it is difficult to store everything in the final hidden state

RNN, LSTM - attention mechanism

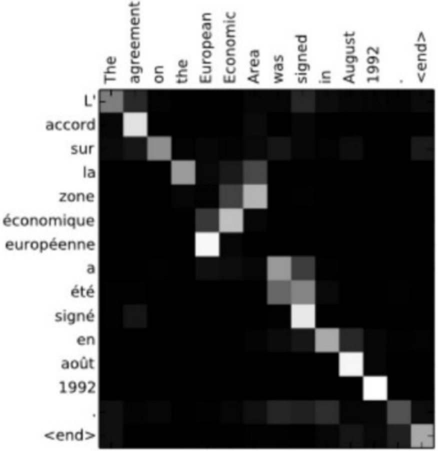
- Attention mechanisms allow neural networks to choose where they focus in the data in order to accomplish certain tasks
  - decoder can then attend to different parts of the source sentence at each step of the translation
  - At each step i of the decoder translation, an alignment model a computes scores between the most recent decoder hidden state and each encoder hidden state

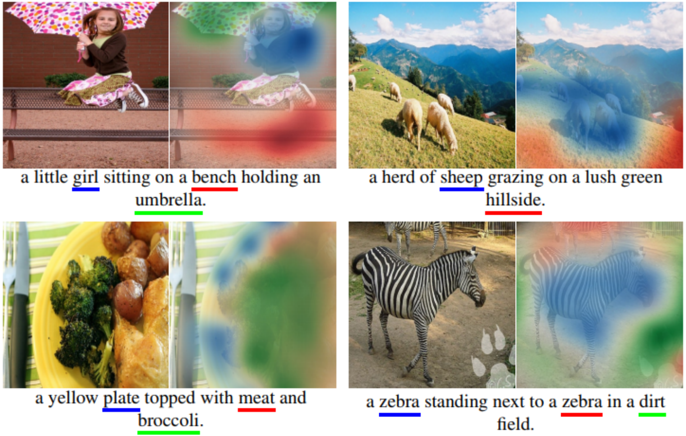
- Vector of the hidden state sequence h_t is fed into a learnable layers $a(h_t)$ (with softmax) to produce a probability vector α (attention) given the most recent decoder state
- The output vector c is a weighted average of $c = \alpha_t \otimes h_t$



RNN, LSTM - attention mechanism

- Attention mechanism also allows to inspect and interpret the model behaviour
  - note English-French related words, and image-text attention





Recurrent Neural Networks

- Recurrent Neural Networks
  - Word embedding
- RNN Unit
- LSTM Unit
- GRU Unit
- Architectures & applications
- Other sequence modelling networks: Autoregressive, WaveNet

Midterm summary

- Machine Learning fundamentals
- Optimization and development process
- Convolutional Neural Networks (Layers)
- Convolutional Neural Networks (Architectures)
- Recurrent Neural Networks

EE3-25 Deep Learning

Part II

- Reinforcement Learning
- Representation learning and Autoencoders
- Adversarial examples
- GANs