

# RNN: recurrent neural networks (part 2)

More advance stuff: for the very  
brave (or the very patient!)

Filippo Biscarini  
Senior Scientist  
CNR, Milan (Italy)

Nelson Nazzicari  
Research fellow  
CREA, Lodi (Italy)



# A sip of NLP

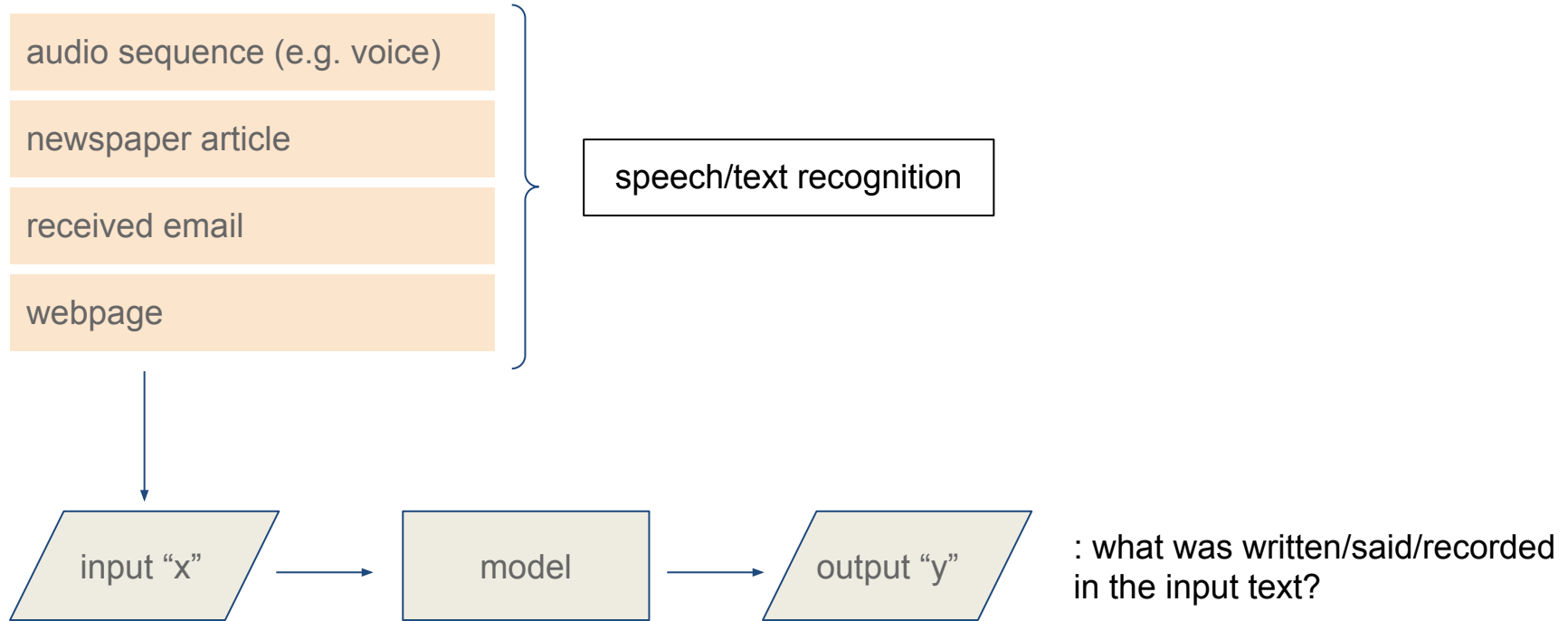
## RNNs for natural language processing

Filippo Biscarini  
*Senior Scientist*  
*CNR, Milan (Italy)*

Nelson Nazzicari  
*Research fellow*  
*CREA, Lodi (Italy)*



# NLP: language models



# NLP: language models

## Speech recognition:

- The red dear is a ruminant
- The red deer is a ruminant

$P(\text{The red dear is a ruminant}) =$

$P(\text{The red deer is a ruminant}) =$



# RNN models for speech/text recognition

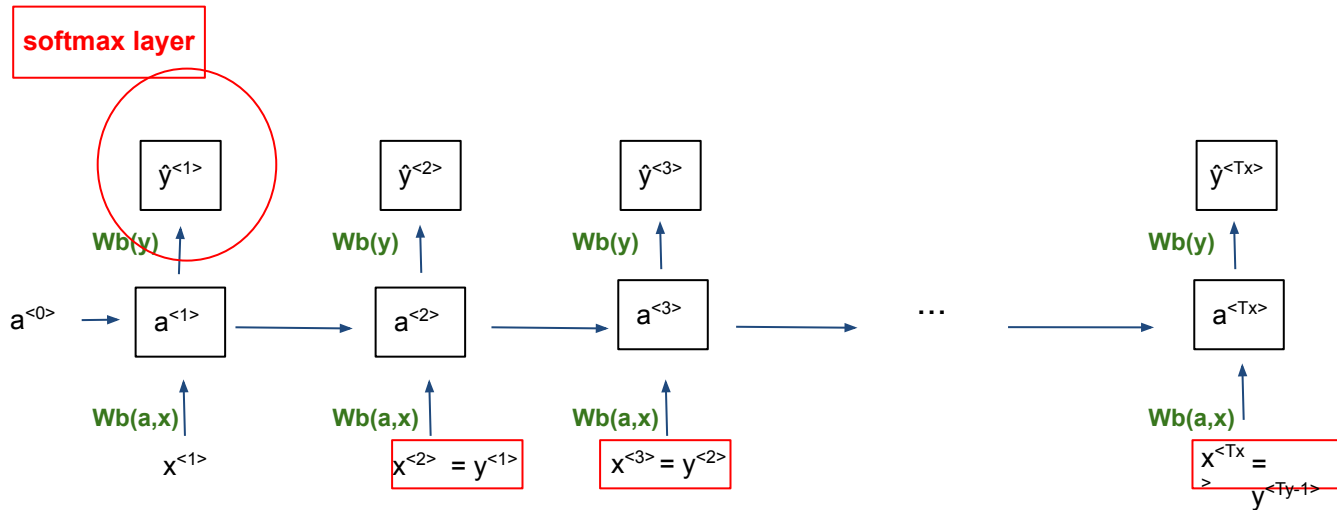
1. **Training set:** large corpus of English texts
2. **Input text/speech:** e.g. “The red deer is a ruminant”
3. **Tokenization:** split the text in tokens (words)

The red deer is a ruminant

Target sentence  
(training set) whose  
probability we want to  
model



# RNN models for speech/text recognition



$$\begin{cases} a^{<t>} = g(W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>} + b_a) \\ \hat{y}^{<t>} = g(W_{ya} \cdot a^{<t>} + b_y) \end{cases}$$



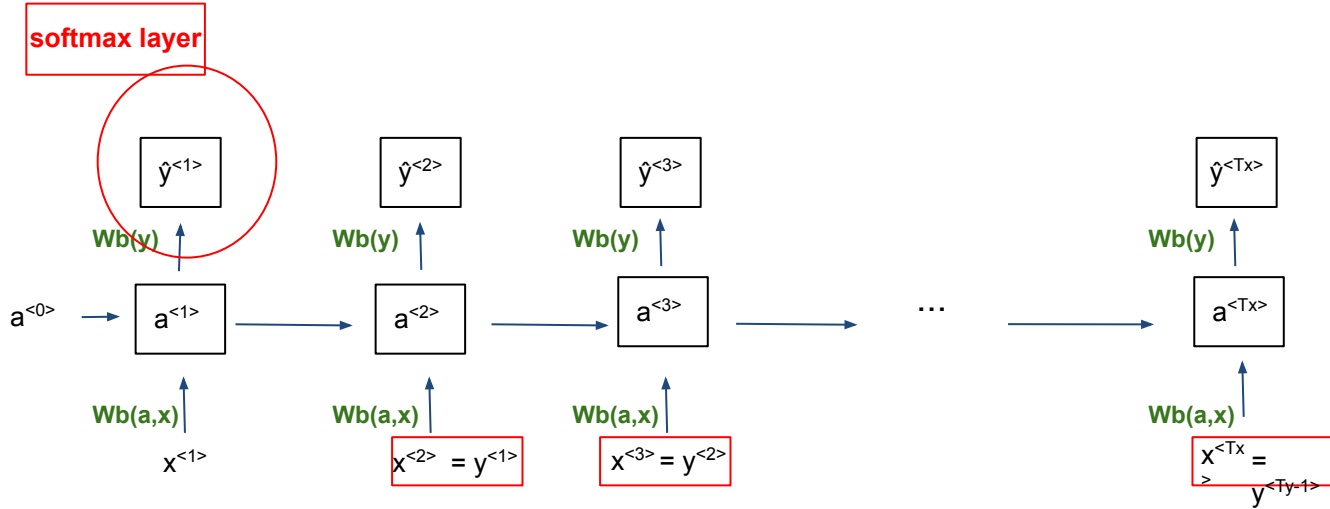
The red deer is a ruminant

# RNN models for speech/text recognition

- in each step, the layer will take some set of the preceding words (directly or through activation) and will calculate the conditional probability of the next word
- the RNN thus learns one word at a time, from left to right



# RNN models for speech rec. - loss function



$$\begin{cases} \mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = - \sum_i y_i^{<t>} \cdot \log(\hat{y}_i^{<t>}) \\ L = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) \end{cases}$$

**(softmax loss function)**





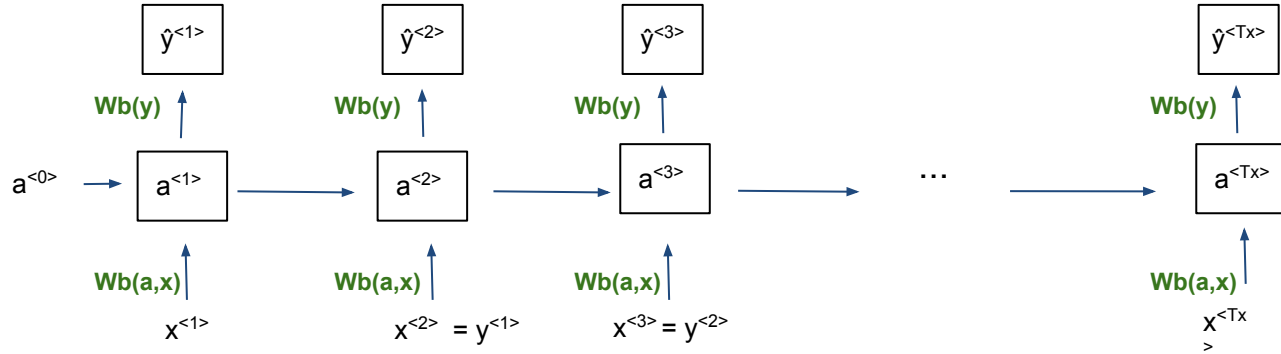
# Trained RNN for sequence models

- speech recognition, sequence generation → examples of **sequence models**
- we saw how to **train a RNN for a sequence model**: architecture, loss, trained output (conditional probabilities of sequences of words)
- with a **trained sequence model** you can then **sample new sequences** (e.g. sequence/text generation)

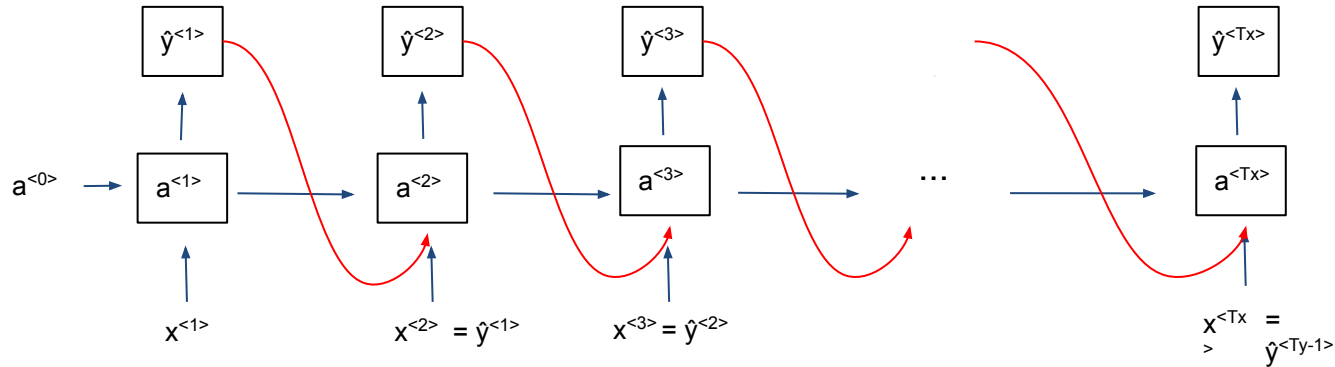


# Sampling new sequences from a trained RNN

TRAINING



SAMPLING



# Sampling new sequences from a trained RNN

- sampling words
- sampling characters from the alphabet (sequence of characters/letters instead of words):
  - no worries about unknown word tokens ( → sequences of characters with zero prob.)
  - much longer sequences → not good at capturing relationships between early parts and later parts of the sequence
  - more computationally expensive
- music notes



# Generating text

## News

President enrique peña nieto, announced  
sench's sulk former coming football langston  
paring.

"I was not at all surprised," said hich langston.

"Concussion epidemic", to be examined. ←

The gray football the told some and this has on  
the uefa icon, should money as.

## Shakespeare

The mortal moon hath her eclipse in love.

And subject of this thou art another this fold.

When besser be my love to me see sabl's.

For whose are ruse of mine eyes heaves.

From Andrew Ng



# Generating text

- there are much more sophisticated RNN models for text generation:
  - [newsyoucantuse.com](http://newsyoucantuse.com)
- also applications to detect fake news
  - [“Fake news detection: A hybrid CNN-RNN based deep learning approach”](#),  
*Nasir, Khan and Varlamis, 2021*

IMMM 2019 : The Ninth International Conference on Advances in Information Mining and Management

## Fake News Detection Method Based on Text-Features

Ahlem Drif

Networks and Distributed Systems Laboratory  
Faculty of Sciences  
University of Sétif 1  
Sétif, Algeria  
Email: adrif@univ.setif.dz

Zineb Ferhat Hamida

Computer Science Department  
University of Sétif 1  
Sétif, Algeria  
Email: zineb.ferhat@yahoo.com

Silvia Giordano

Networking Lab, SUPSI  
University of Applied Sciences of Southern Switzerland  
Lugano, Switzerland  
Email: silvia.giordano@supsi.ch



# Vanishing gradients

A problem of memory loss

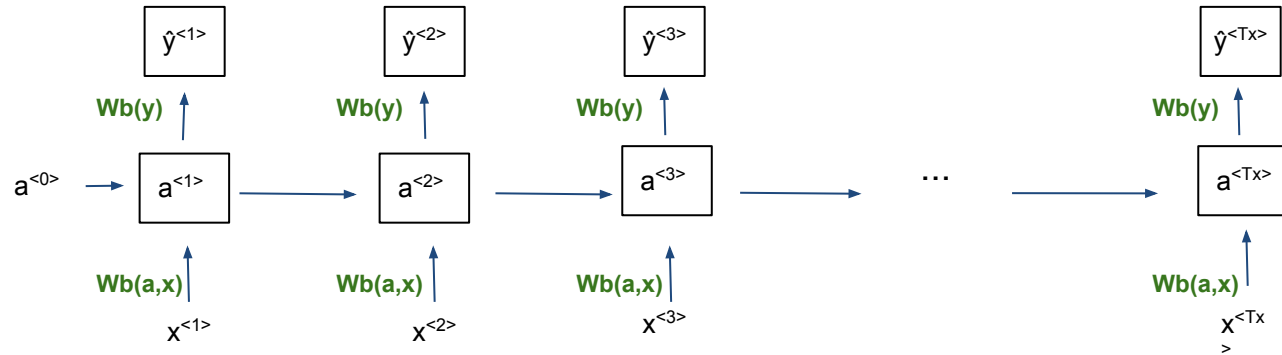
Filippo Biscarini  
Senior Scientist  
CNR, Milan (Italy)

Nelson Nazzicari  
Research fellow  
CREA, Lodi (Italy)



# Vanishing gradients with basic RNN

The red deer, that live in the woods, graze on ... and mate during ... at temperate latitudes ... is a ruminant  
 The red **deers**, that live in the woods, graze on ... and mate during ... at temperate latitudes ... **are** ruminants

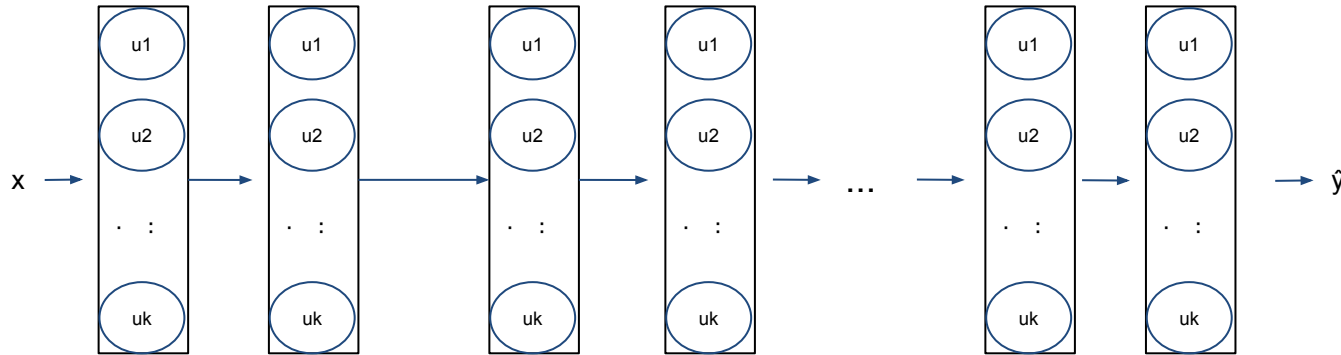


The basic RNN model is not good at capturing **distant relationships** in the sequence



# Vanishing gradients with basic RNN

- problem with very deep NNs (and **RNNs are very very deep NNs!**)
- in a deep NN backprop has difficulties to have an effect on the early layers of the network (won't affect the weights) → **loss of memory!**





# Why do gradients vanish?

Some basic intuition:

- a deep NN can be thought of as a series of matrix products;
- simplifying further:

$$\mathbf{W}^L \text{ (with } L = \text{n. of layers)}$$

- when you combine many features ( $x_i$ ), the coefficients (weights) will tend (have) to be small:

$$\mathbf{w}_1 * \mathbf{x}_1 + \mathbf{w}_2 * \mathbf{x}_2 \dots \mathbf{w}_m * \mathbf{x}_m \text{ (with } m = \text{n. of features)}$$



# Why do gradients vanish?

Some basic intuition:

- a deep NN can be thought of as a series of matrix products;
- simplifying further:

$$\mathbf{W}^L \text{ (with } L = \text{n. of layers)}$$

- when you combine many features ( $x_i$ ), the coefficients (weights) will tend (have) to be small:

$$\mathbf{w}_1 * \mathbf{x}_1 + \mathbf{w}_2 * \mathbf{x}_2 \dots \mathbf{w}_m * \mathbf{x}_m \text{ (with } m = \text{n. of features)}$$

**many small weights** ( $-1 < w < 1$ ), **many layers** (exponent) → **vanishing gradients**



# Gated Recurrent Unit (GRU)

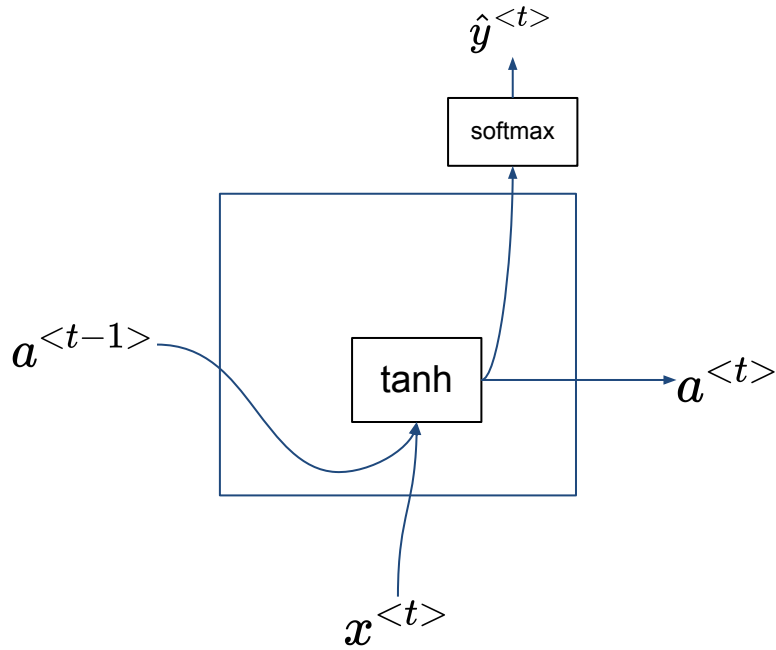
Memory pills!

Filippo Biscarini  
Senior Scientist  
CNR, Milan (Italy)

Nelson Nazzicari  
Research fellow  
CREA, Lodi (Italy)



# RNN unit



$$\begin{cases} a^{<t>} = g(W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>} + b_a) \\ \hat{y}^{<t>} = g(W_{ya} \cdot a^{<t>} + b_y) \end{cases}$$



# RNN unit - GRU (gated recurrent unit)

$c$  = memory cell (e.g. remember deer/deers)

$$c^{<t>} = a^{<t>}$$

$$\tilde{c}^{<t>} = \tanh(W_{cc} \cdot c^{<t-1>} + W_{cx} \cdot x^{<t>} + b_c)$$

Introducing the gate

$$\Gamma_u = \sigma(W_{uu} \cdot c^{<t-1>} + W_{ux} \cdot x^{<t>} + b_u)$$



# RNN unit - GRU (gated recurrent unit)

- in a GRU unit we calculate a value  $\tilde{\mathbf{c}}^{<t>}$
- we may then potentially **update** the value in the memory cell by replacing  $\mathbf{c}^{<t>}$  with  $\tilde{\mathbf{c}}^{<t>}$
- the value of the **gate**  $\Gamma_u$  will decide whether or not  $\mathbf{c}^{<t>}$  will be updated



**This is the key part of GRU!**

The red deer, that live in the woods, graze on ... and mate during ... at temperate latitudes ... is a ruminant



# RNN unit - GRU (gated recurrent unit)

$c$  = memory cell (e.g. remember deer/deers)

$$c^{<t>} = a^{<t>}$$

$$\tilde{c}^{<t>} = \tanh(W_{cc} \cdot c^{<t-1>} + W_{cx} \cdot x^{<t>} + b_c)$$

## Introducing the gate

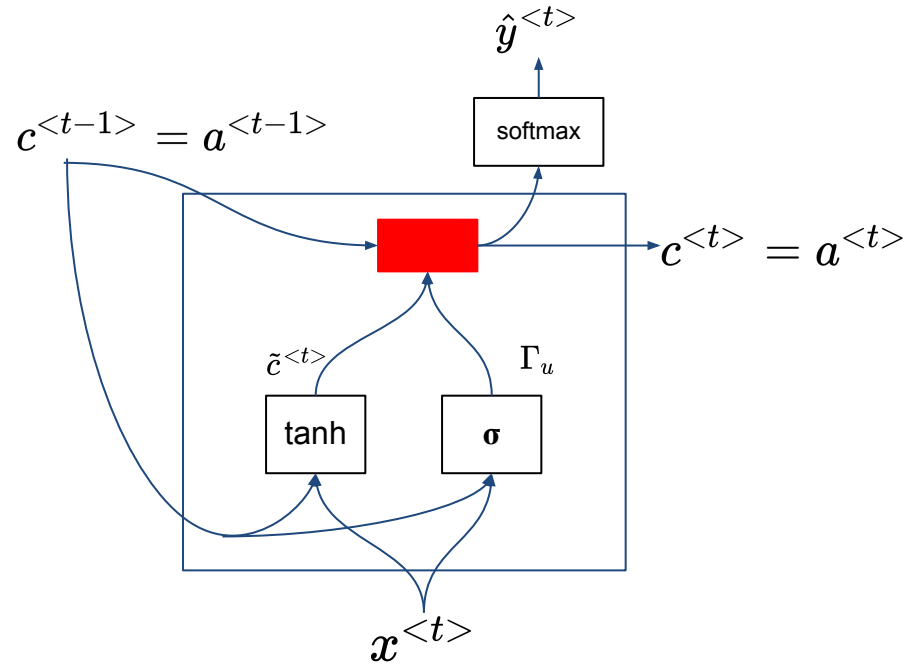
$$\Gamma_u = \sigma(W_{uu} \cdot c^{<t-1>} + W_{ux} \cdot x^{<t>} + b_u)$$

$$c^{<t>} = \Gamma_u \cdot \tilde{c}^{<t>} + (1 - \Gamma_u) \cdot c^{<t-1>}$$

The red deer, that live in the woods, graze on ... and mate during ... at temperate latitudes ... is a ruminant



# RNN unit - GRU (schematic representation)



$c$  = memory cell (e.g. remember deer/deers)

$$c^{<t>} = a^{<t>}$$

$$\tilde{c}^{<t>} = \tanh(W_{cc} \cdot c^{<t-1>} + W_{cx} \cdot x^{<t>} + b_c)$$

Introducing the gate

$$\Gamma_u = \sigma(W_{uu} \cdot c^{<t-1>} + W_{ux} \cdot x^{<t>} + b_u)$$

$$c^{<t>} = \Gamma_u \cdot \tilde{c}^{<t>} + (1 - \Gamma_u) \cdot c^{<t-1>}$$





# GRU units

- good at learning when to update  $c^{<t>}$  and then keep it constant until used
- $\Gamma_u$  very close to zero  $\rightarrow c^{<t>} = c^{<t-1>}$  across many layers  $\rightarrow$  reduced problems with vanishing gradients
- GRU units / layers can learn long-range dependencies in sequences
- $c^{<t>}$  can be a vector  $\rightarrow$  multiple memory cells to “remember” multiple things (e.g. singular/plural, past/present, context etc.)
- full GRU units are more complex and include one additional gate, i.e.  $\Gamma_r$  for the relevance (weight) of elements in the sequence ( $\Gamma_r$  would be used in the calculation of  $\tilde{c}^{<t>}$ , the candidate replacement for  $c^{<t>}$ )



# DNN units

GRU unit: were we doing linear regression inside this unit?



# DNN units

GRU unit: were we doing linear regression inside this unit? → **NO!**

- Dense layers: “linear regression”
- CNN layers: convolutions
- Simple RNN: linear combination of the features + activations “from the past”
- RNN+GRU layers: gate/relevance/update calculations
- RNN+LSTM: (we’ll see in a while)
- etc.

With DNN you need:

- “invent” some calculations
- repeat it thousands of times
- let the NN learn the parameters
- and ... “bam”, the job is done! ;-)



# Long-Short Term Memory (LSTM) Unit

More memory pills!

Filippo Biscarini  
*Senior Scientist*  
*CNR, Milan (Italy)*

Nelson Nazzicari  
*Research fellow*  
*CREA, Lodi (Italy)*



# GRU unit

vs

# LSTM unit

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r \cdot c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_{uu} \cdot c^{<t-1>} + W_{ux} \cdot x^{<t>} + b_u)$$

$$\Gamma_r = \sigma(W_{rr} \cdot c^{<t-1>} + W_{rx} \cdot x^{<t>} + b_r)$$

$$c^{<t>} = \Gamma_u \cdot \tilde{c}^{<t>} + (1 - \Gamma_u) \cdot c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_{uu} \cdot a^{<t-1>} + W_{ux} \cdot x^{<t>} + b_u)$$

$$\Gamma_f = \sigma(W_{ff} \cdot a^{<t-1>} + W_{fx} \cdot x^{<t>} + b_f)$$

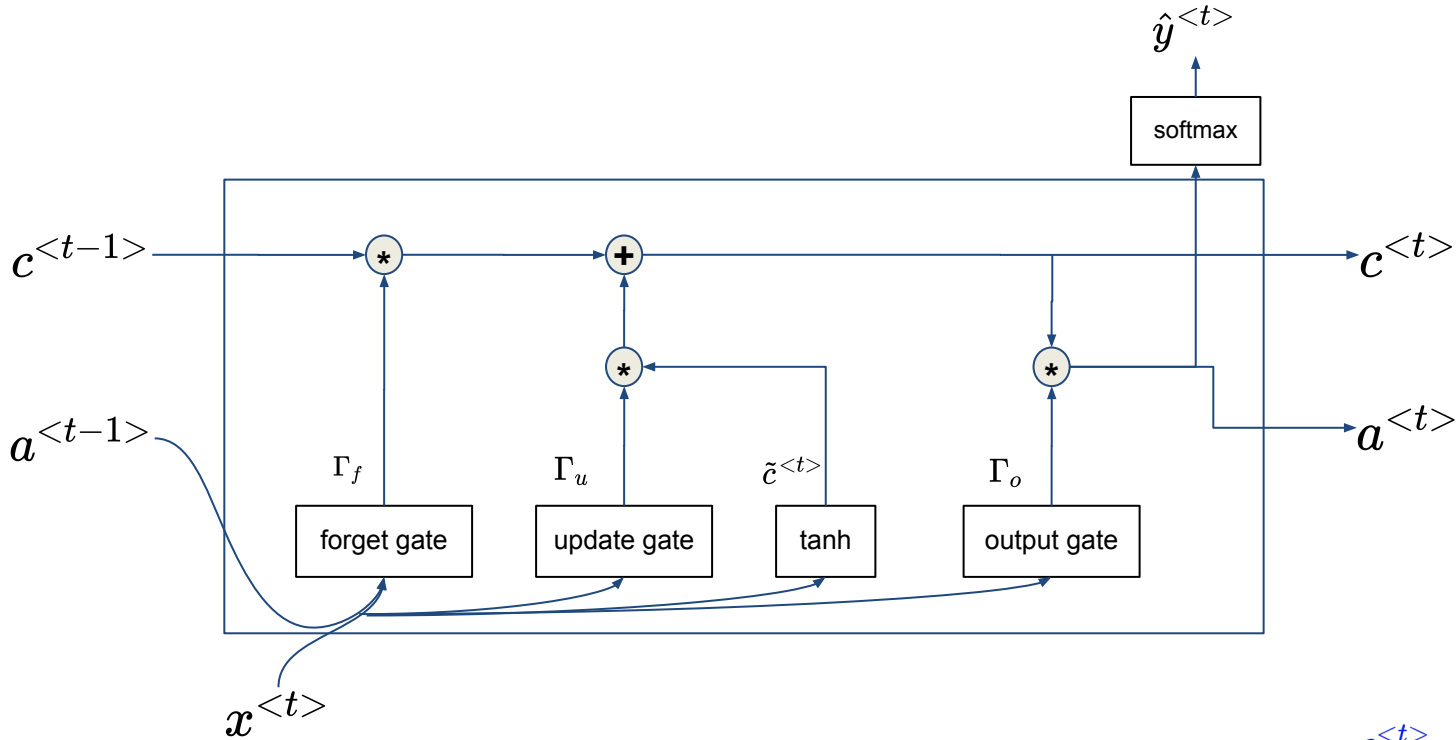
$$\Gamma_o = \sigma(W_{oo} \cdot a^{<t-1>} + W_{ox} \cdot x^{<t>} + b_o)$$

$$c^{<t>} = \Gamma_u \cdot \tilde{c}^{<t>} + \Gamma_f \cdot c^{<t-1>}$$

$$a^{<t>} = \Gamma_o \cdot \tanh(c^{<t>})$$



# LSTM unit - schematic representation

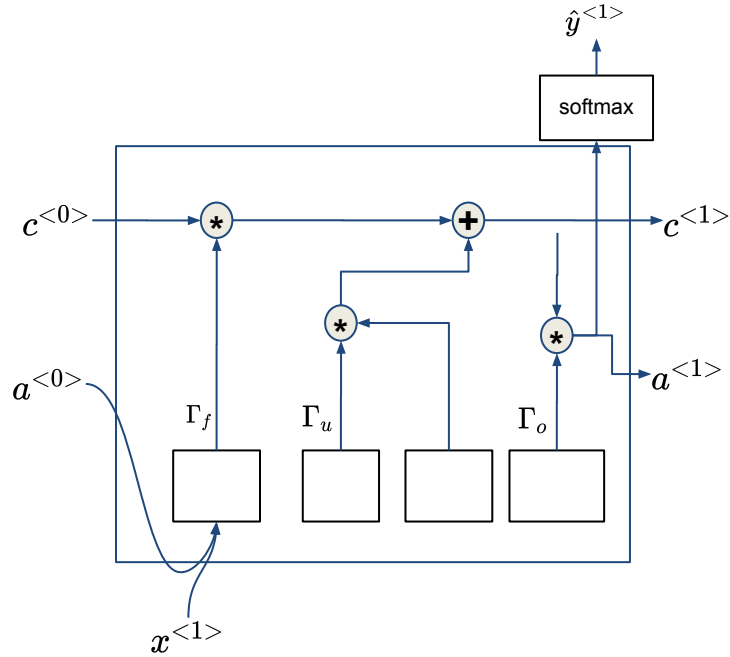


$$c^{<t>} = \Gamma_u \cdot \tilde{c}^{<t>} + \Gamma_f \cdot c^{<t-1>}$$

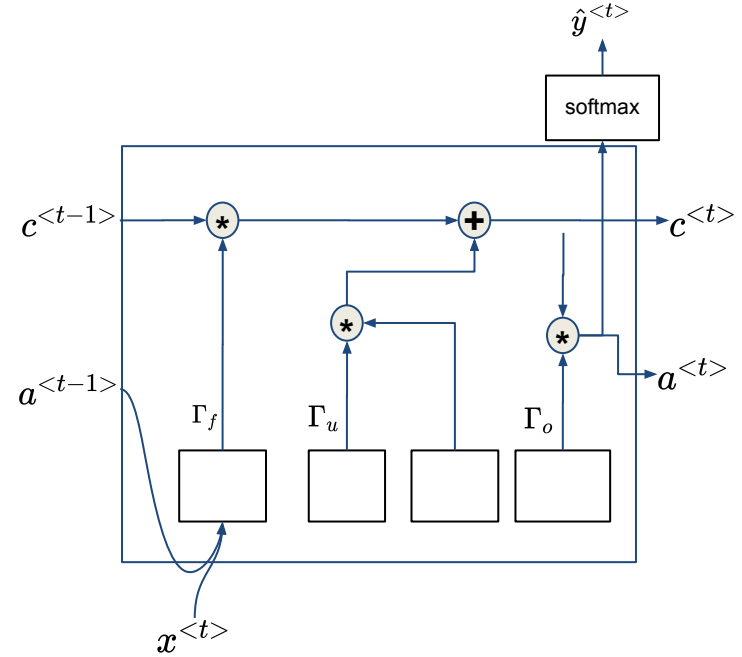
$$a^{<t>} = \Gamma_o \cdot \tanh(c^{<t>})$$



# LSTM units: with the memory flow



...



# LSTM vs GRU - take #2

- LSTM (introduced in 1997): pass memory info (cell state) alongside (but separately) the hidden state (activation)
- GRU (introduced in 2014): gets rid of cell state, and pass memory info intermingled with the activation (hidden state)
- GRU perform fewer tensor operations → fewer parameters, faster calculations
- We showed GRU before LSTM for illustration purposes (GRU simpler to explain)





# Bidirectional RNN

Back from the future!

Filippo Biscarini  
*Senior Scientist*  
*CNR, Milan (Italy)*

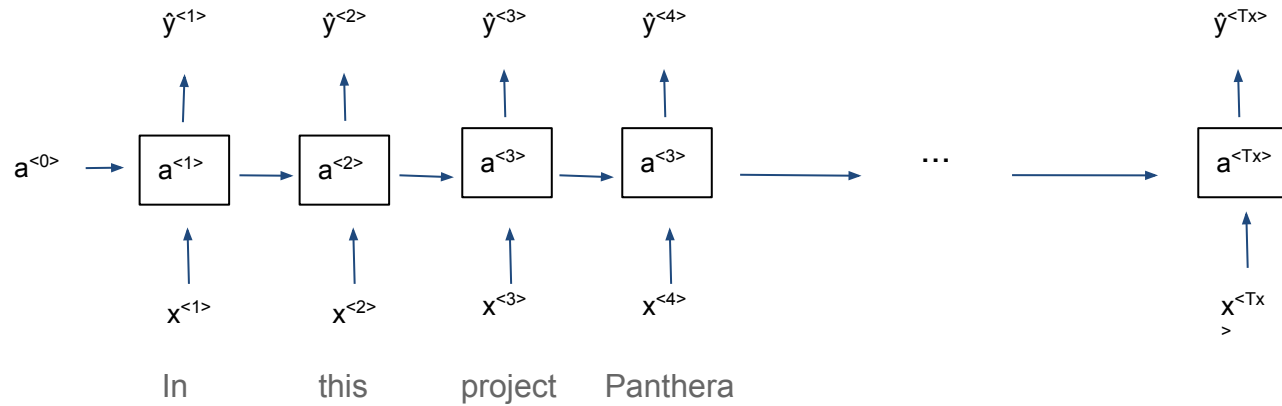
Nelson Nazzicari  
*Research fellow*  
*CREA, Lodi (Italy)*



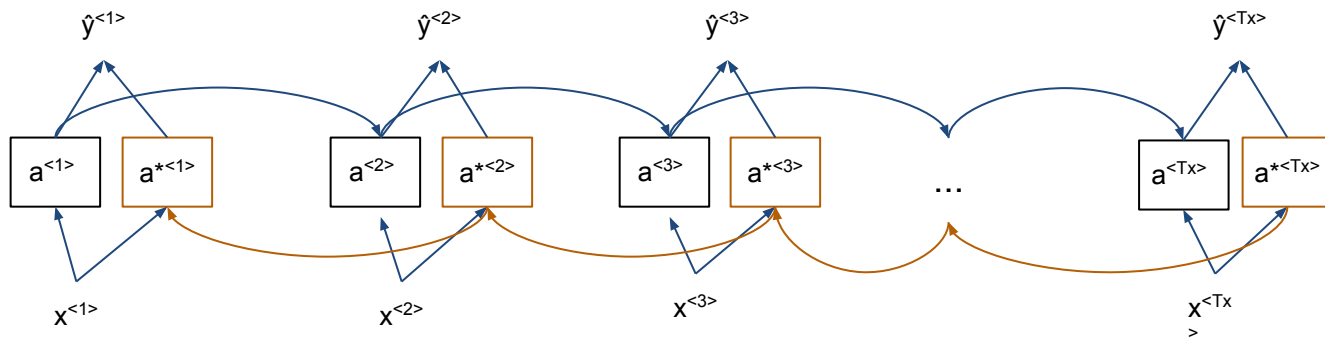
# Using information from the “future”

“In this project Panthera leo samples are used”

“In this project Panthera Corporation is the leading partner”



# Using information from the “future”



- acyclic graph
- the **forward sequence** starts from  $a^{<1>}$  ... to  $a^{<T_x>}$  (as usual)
- the **backward sequence** will begin from the end ( $a^{*<T_x>}$ ) and move from right to left



# Pros and cons of bidirectional RNN

- uses information from past, present and future (early and late parts of the sequence)
- making predictions anywhere in the middle of a sequence
- bidirectional RNN + LSTM units → common choice in NLP problems
- you need to process the entire sequence before making predictions (e.g. in speech recognition you need to wait for the person to finish talking) [ → more sophisticated models are needed]
- computationally expensive



# Deep RNN

Not complex enough? Let's stack  
RNN layers!

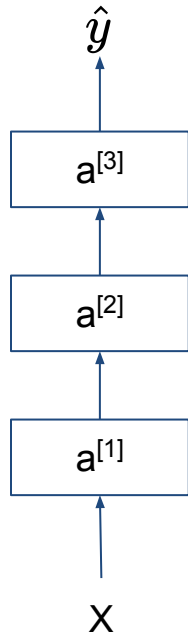
Filippo Biscarini  
*Senior Scientist*  
*CNR, Milan (Italy)*

Nelson Nazzicari  
*Research fellow*  
*CREA, Lodi (Italy)*



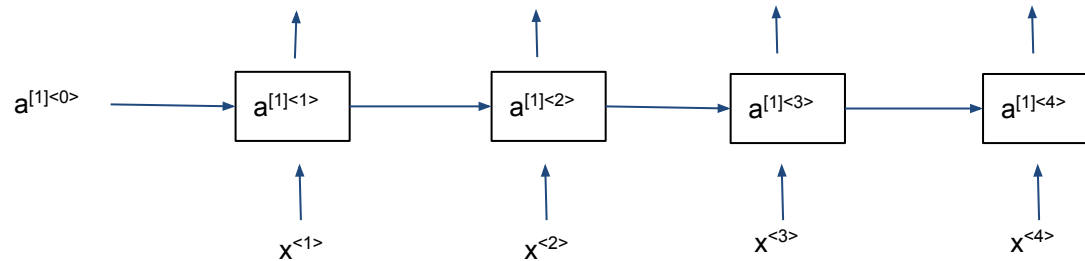
# Deep RNN

## Standard multilayer NN



## Basic RNN model

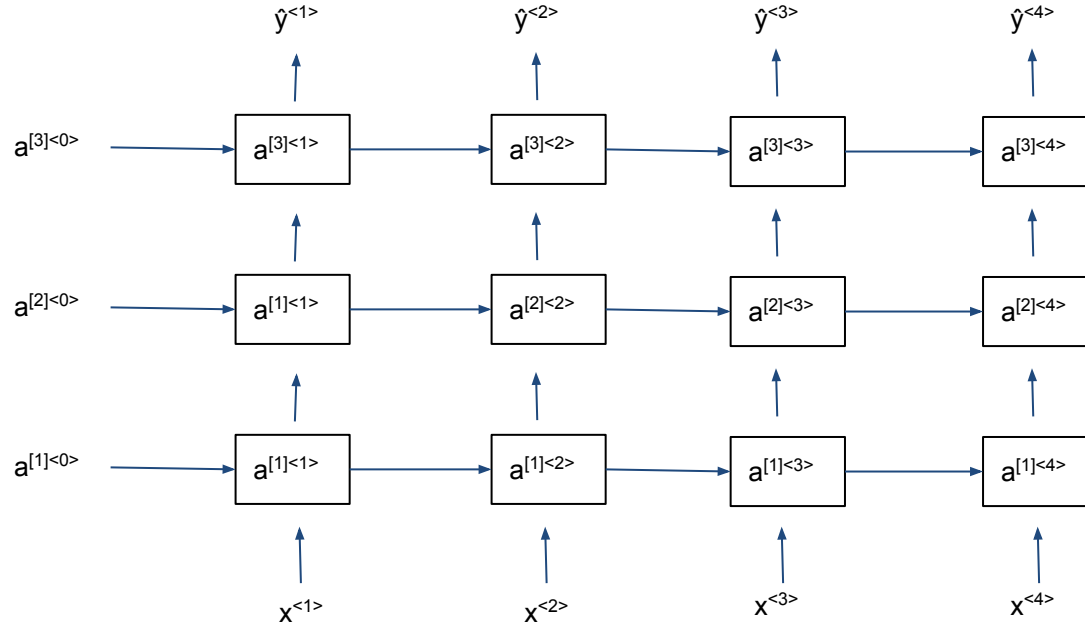
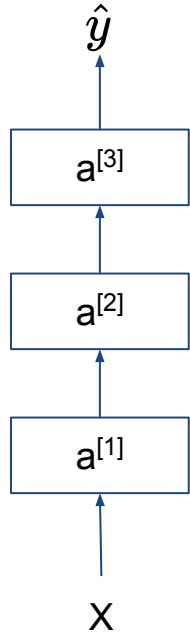
- $\langle t \rangle$ : time dimension (each element of the sequence)  $\rightarrow$  units inside (standard, GRU, LSTM: multiple units/nodes per  $\langle t \rangle$ )
- $[l]$ : layers (stack of RNN layers)



# Deep RNN

## Multilayer RNN model

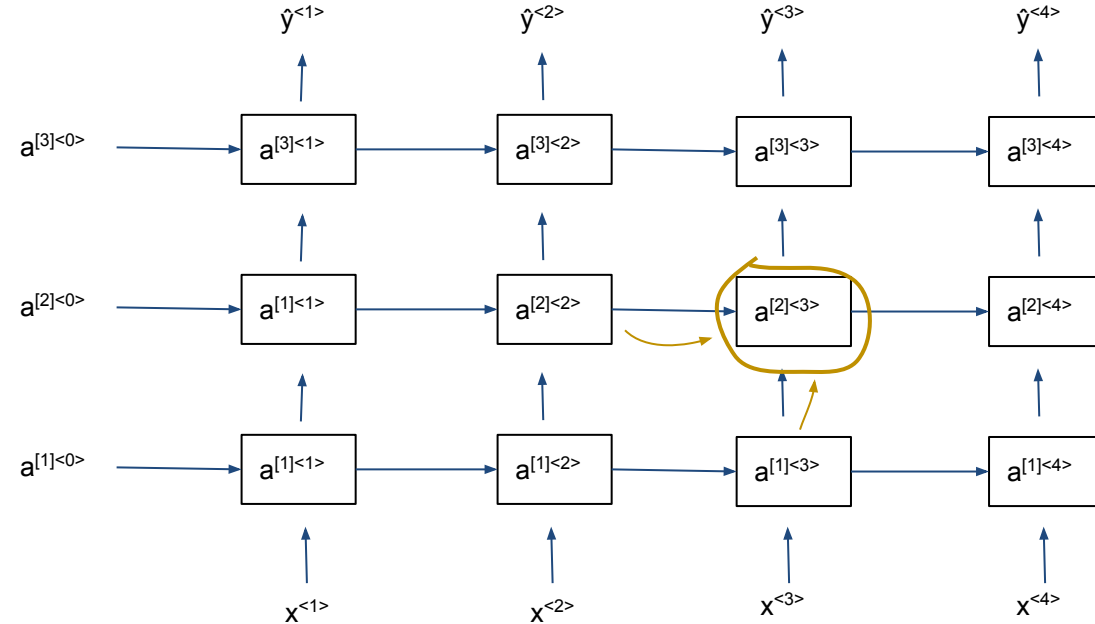
### Standard multilayer NN



# Deep RNN - example calculation

From basic RNN models

$$a^{<t>} = g(W_a[a^{<t-1>}, x^{<t>}] + b_a)$$





# RNN models

- demonstration

- day4\_code02 RNN-1 architectures.ipynb (GRU, LSTM layers)

- day5\_code02 RNN-2 architectures.ipynb (advanced forecasting)

- day5\_code03 RNN-3 text-sequence-data.ipynb

