RNN: recurrent neural networks

Time (order) matters: sequence (longitudinal) data

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)

Sequence data problems

INPUT DATA	→	OUTPUT DATA
sequence	→	sequence
number/category/ string	→	sequence
sequence	→	number/category/ string

Sequence data problems - examples

INPUT DATA		OUTPUT DATA	
audio sequence	→	text sequence	[speech recognition]
type of music (integer/string)	→	audio sequence	[music generation]
text sequence	→	category/score	[text classification]
DNA sequence	→	coding sequence	[pattern recognition]
text sequence in Italian	→	text sequence in English	[text translation]

Sequence data problems - examples

INPUT DATA		OUTPUT DATA	
audio sequence	→	text sequence	[speech recognition]
type of music (integer/string)	→	audio sequence	[music generation]
text se			ification]
DNA sequence	→	coding sequence	[pattern recognition]
text sequence in Italian	→	text sequence in English	[text translation]

Supervised learning!

Sequence models - data representation

<u>Pattern (entity) recognition problem</u>

x: in the impenetrable forest there are populations of *Panthera leo* and *Loxodonta africana*

y:

output representations:

- vector of 1's and 0's (wild scientific animal names or not)
- start and end position of animal names
- ...

Sequence models - data representation

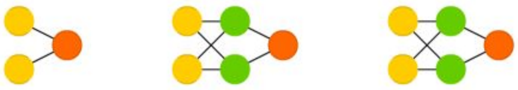
Pattern (entity) recognition problem

x: in the impenetrable forest there are populations of *Panthera leo* and *Loxodonta africana*

Vocabulary / dictionary

$$\begin{bmatrix} \text{a} \\ \text{Aarhus} \\ \ldots \\ \text{africana} \\ \text{are} \\ \ldots \\ \text{Leo} \\ \ldots \\ \ldots \\ \text{Panthera} \\ \ldots \\ \text{Wagyu} \\ \ldots \\ \text{zebra} \\ \ldots \\ \text{Zürich} \end{bmatrix}$$

- ex. 10,000 words
- OHE: one-hot encoding (“T” *1-hot* 10,000-long vectors)
- supervised learning of a function f(x) that maps **x**→**y**



Building a NN model for sequence data

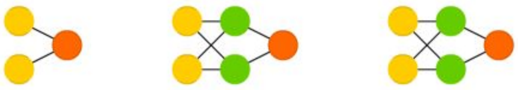
From dense NNs to RNNs

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

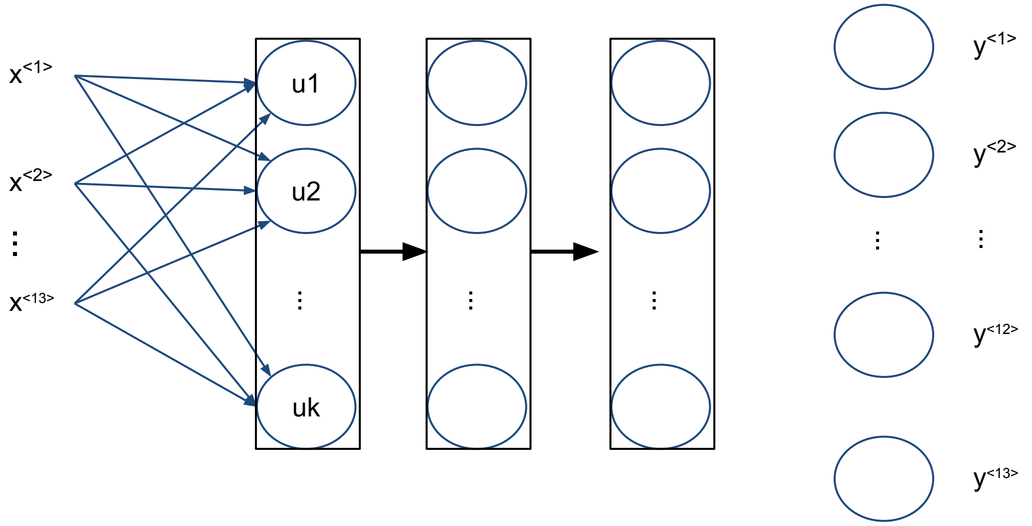
Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)

A neural network model for word recognition

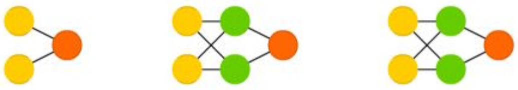
- text data + vocabulary → **data representation** (sequence of 1-hot-enc. vectors and labels)
- **objective**: find (approximate) function that maps 1-hot vectors to labels (x → y)
- y = f(x)
- which neural network **architecture**? Shall we try a dense neural network?



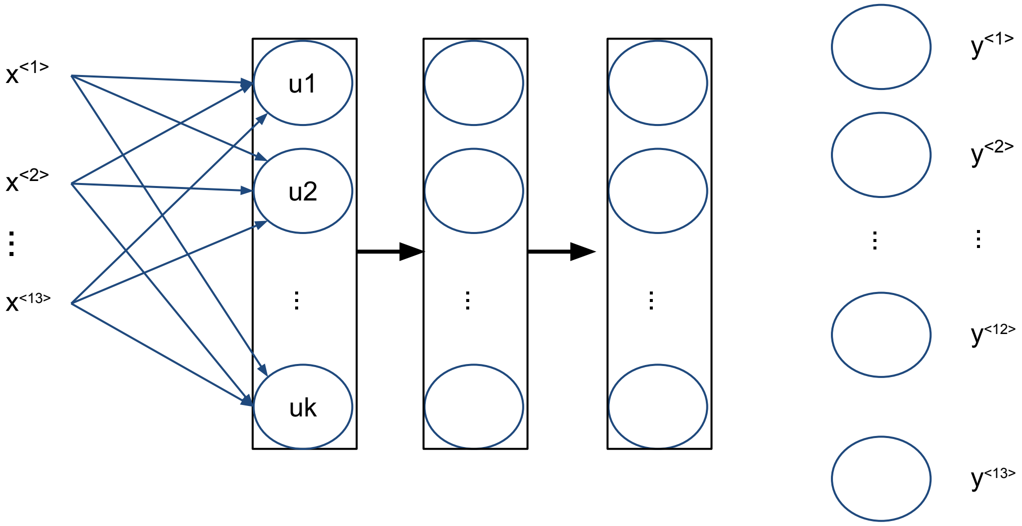
Let’s try a standard (dense) neural network



T = 13 words, **3** hidden layers (**u**: units), **one** output layer (13 labels)



Let’s try a standard (dense) neural network

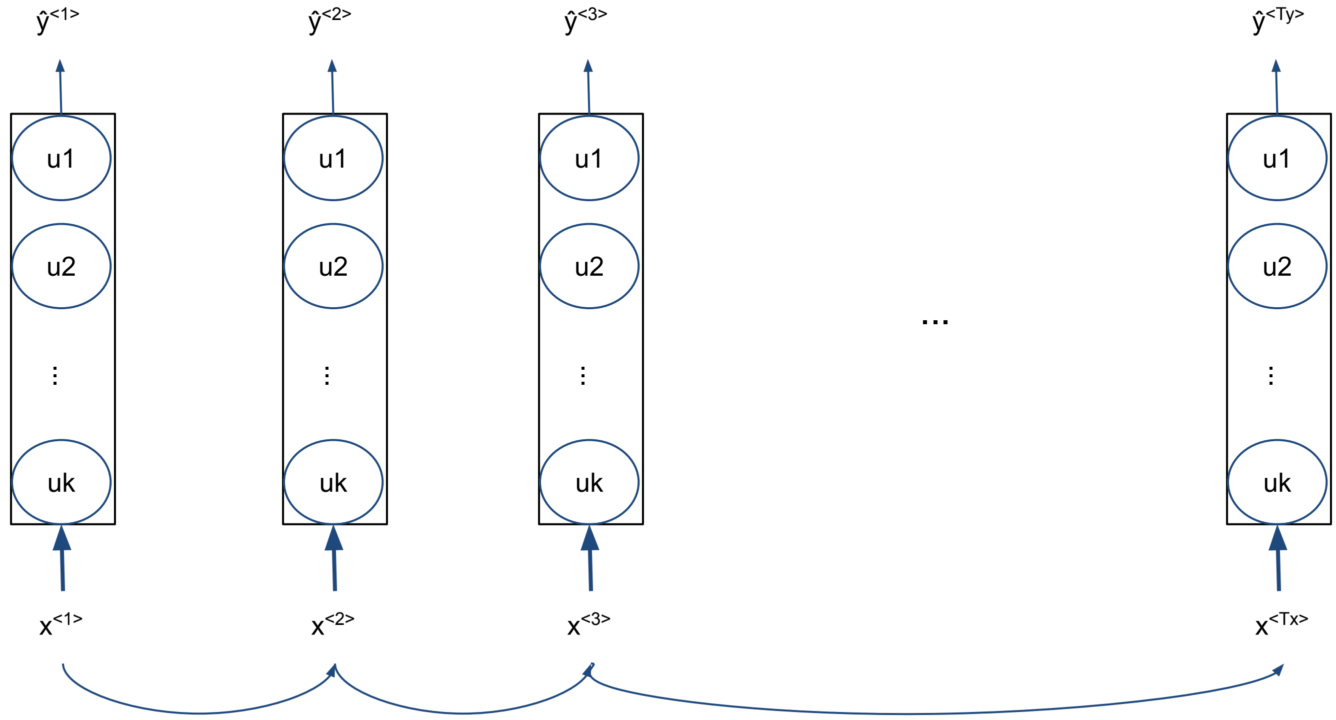


T = 13 words, **3** hidden layers (**u**: units), **one** output layer (13 labels)

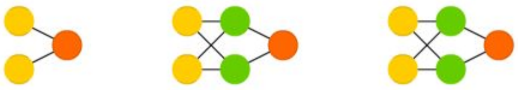
Won’t work!

- inputs, outputs can have **different lengths** in different sentences (examples) [zero-padding may circumvent, but suboptimal representation]
- **doesn’t transfer learning** along the sequence!
- the **number of parameters** to learn quickly explodes!
  → [(vocabulary size x T (max sentence length) x n. of nodes x n. of layers]

Recurrent Neural Network (RNN)



sequentially (word by word)



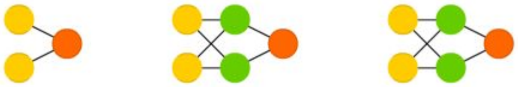
Recurrent Neural Network (RNN)

- words (sequences) are **analysed sequentially**, from left to right (or from top to bottom etc.)
- each time, the transformed information from the previous word/sequence (**activation values**) is passed on to the next word/sequence → transferring learning along the sequence!
- weakness: only previous information is used!

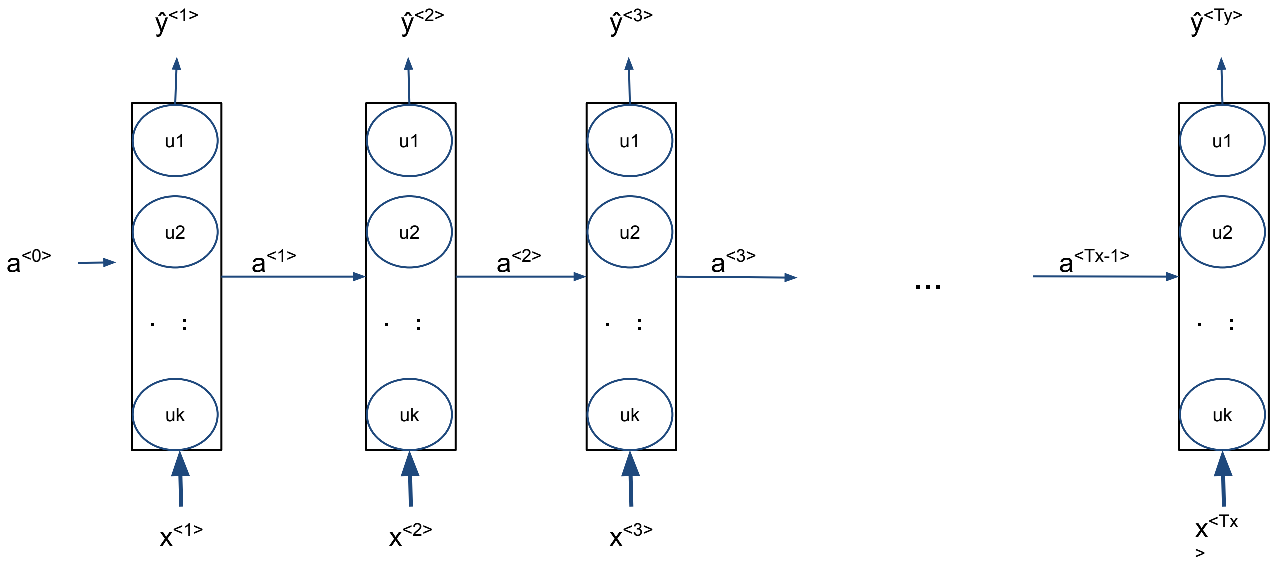
(e.g. “In this project, Panthera leo samples are used”

“In this project, Panthera Corporation is the leading partner”)

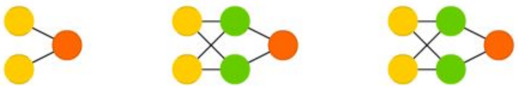
[bidirectional RNNs (BRNNs) offer a solution to this problem]



Simple unidirectional RNN



- dense NN: data are fed to the first layer, than activation values are passed from one layer to the other
- RNN: **data + activation values** from previous layer are fed to each layer sequentially → **memory!**



Simple RNN: forward propagation

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

- $a^{<0>}$ and $x^{<0>}$ → $a^{<1>}$ and $\hat{y}^{<1>}$
- $a^{<1>}$ and $x^{<1>}$ → $a^{<2>}$ and $\hat{y}^{<2>}$
- $a^{<2>}$ and $x^{<2>}$ → $a^{<3>}$ and $\hat{y}^{<3>}$
- and so on ...

- **Tanh** or **Relu** for the activation layer
- **Sigmoid** or **softmax** for the output layer

- **W_{aa} W_{ax} W_{ya}:** model coefficients
- **b_a b_y**: bias terms

Simple RNN: let’s work out the dimensions

$$\begin{cases} a^{<t>}_{(u,1)} = g\left(W_{aa(u,u)} \cdot a^{<t-1>}_{(u,1)} + W_{ax(u,m)} \cdot x^{<t>}_{(m,1)} + b_{a(u,1)}\right) \\ \\ \hat{y}^{<t>}_{(1,1)} = \sigma\left(W_{ya(1,u)} \cdot a^{<t>}_{(u,1)} + b_{y(1,1)}\right) \end{cases}$$

- **u**: **n. of units** (nodes) in the layer
- **m**: n. of features (**vocabulary size**)
- **sigmoid activation**: binary classification (name / no-name)
- for multiclass, softmax would be used, and y_hat would have dimensions (c,1), with c = n. of classes

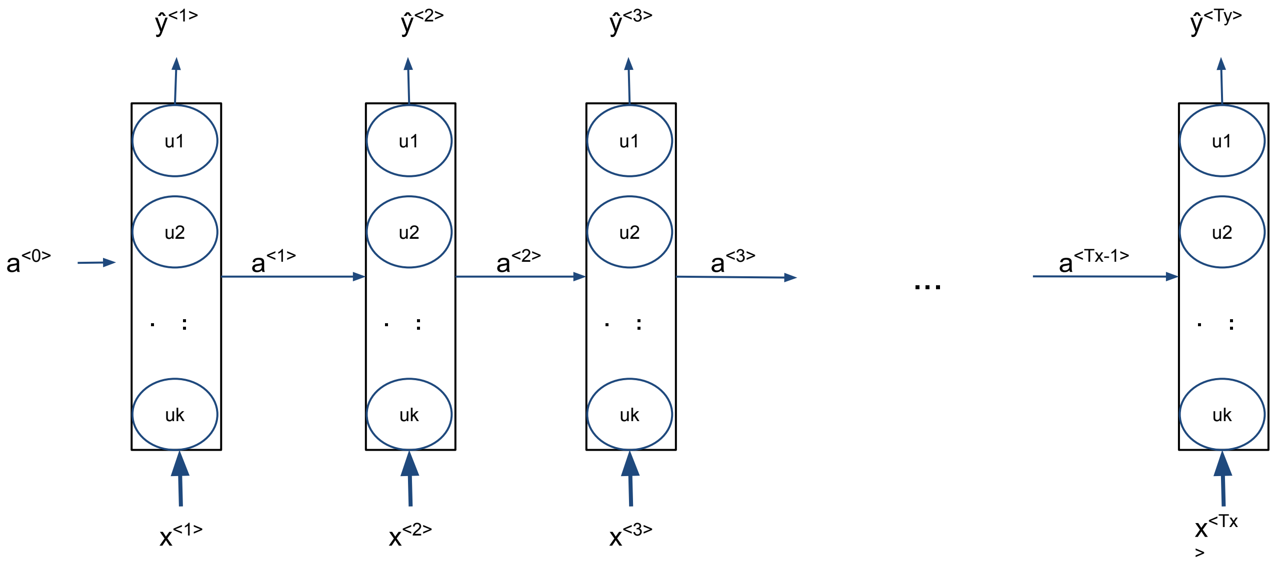
Back to the future!

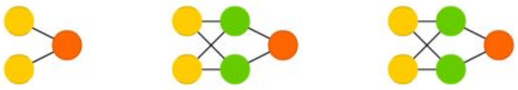
Back propagation for RNNs

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

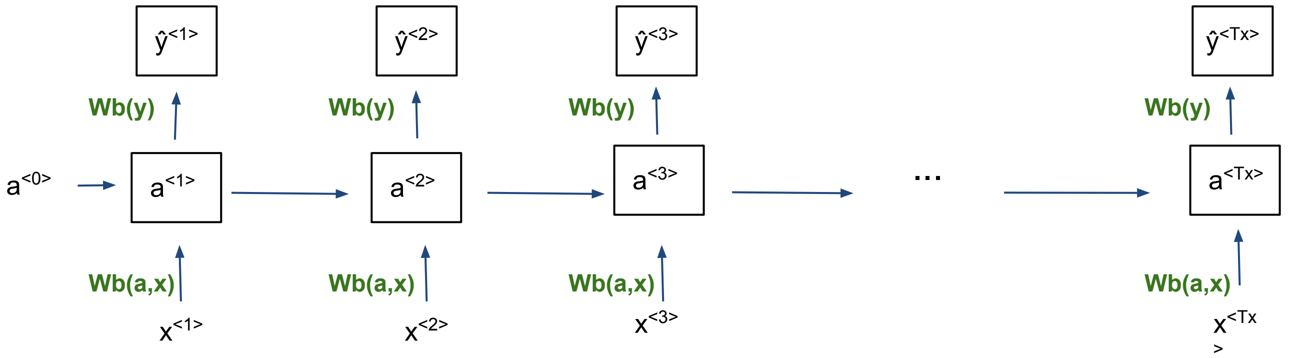
Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)

RNN: forward and back propagation





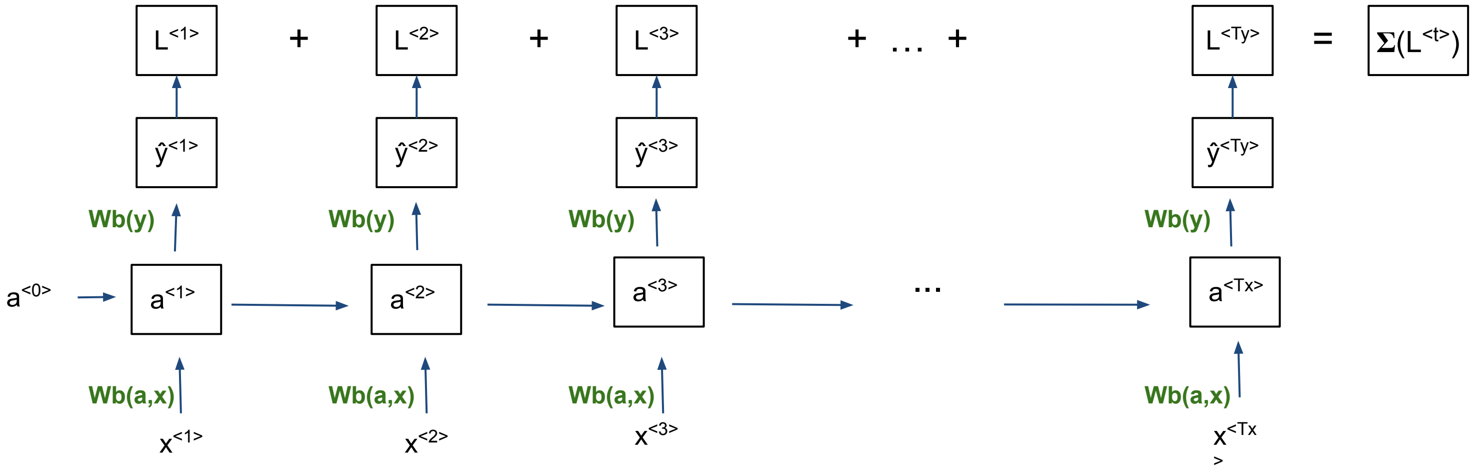
RNN: forward and back propagation



This is **forward propagation**: with **data** and **parameters (weights/coefficients)** we go through the network and obtain **predictions**

- **How do we calculate the weights of the model?**

RNN: loss function



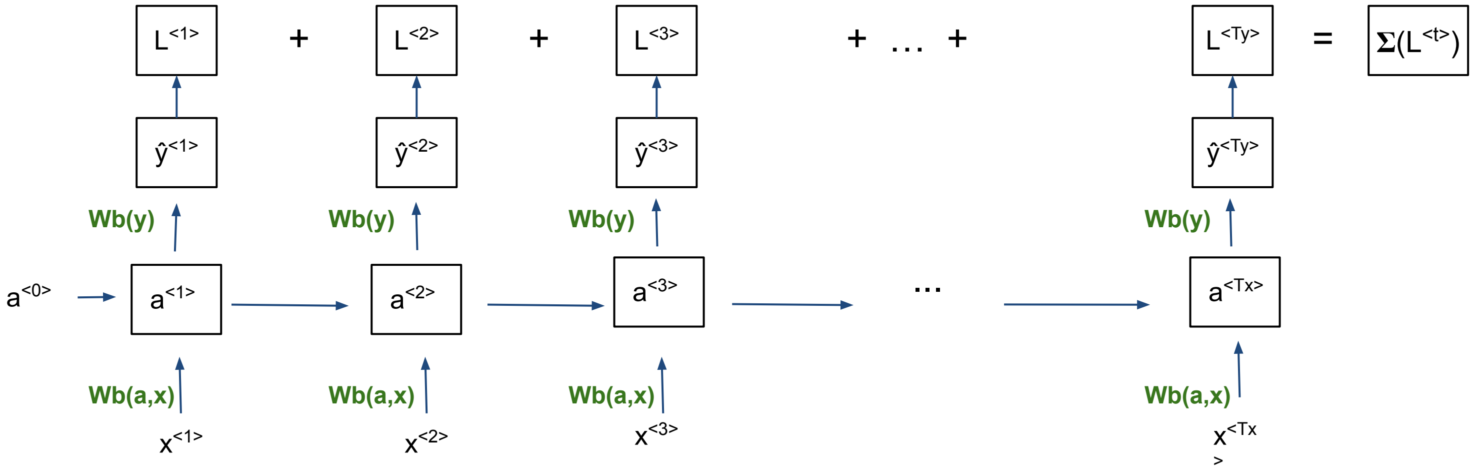
$$\mathcal{L}^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right) =$$

← loss for single word (position)

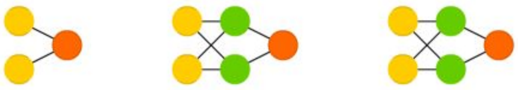
$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{Ty} \mathcal{L}^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right)$$

← loss for the whole sequence (sentence)

RNN: backpropagation through time



- with **loss functions** and their **partial derivatives** (with respect to model coefficients), we can move through the network and **update the coefficients** (~ gradient descent)
- **“Backpropagation through time”**: algorithm to solve RNNs (from right to left, over decreasing time indices “t”, kind of backwards in time)



Architects at work

Different RNN architectures

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

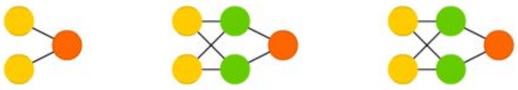
Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)

RNNs: input / output

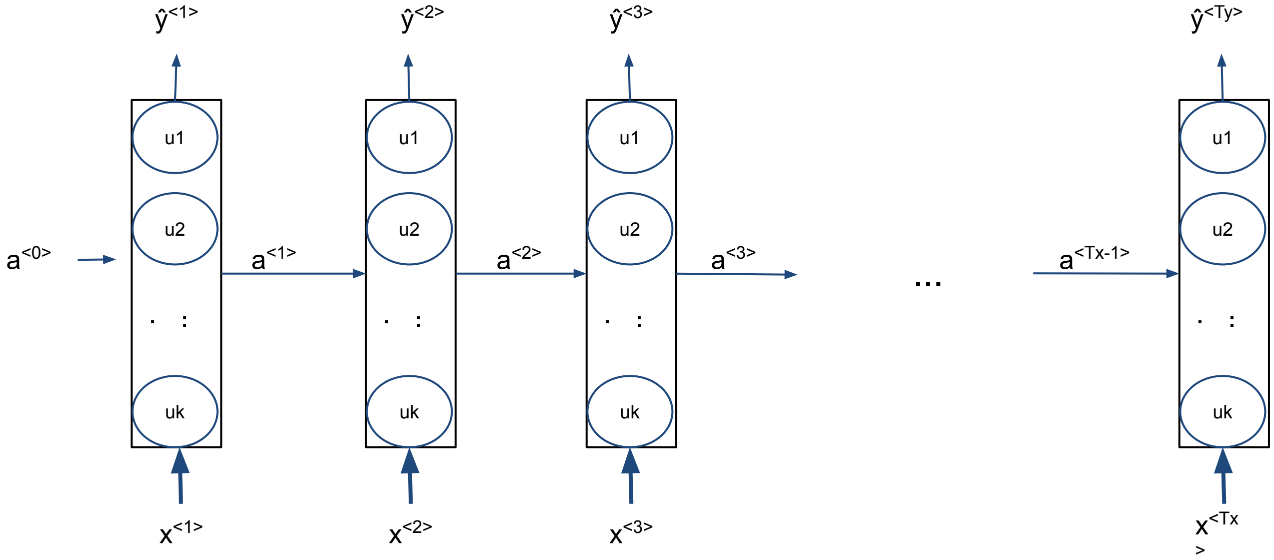
So far:

- simple unidirectional RNN
- input and output: same type, same dimension (Tx = Ty)

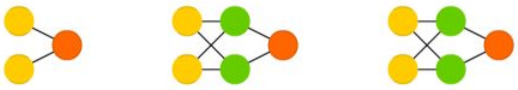
audio sequence →	text sequence	**[speech recognition]**
type of music (integer/string) →	audio sequence	**[music generation]**
text sequence →	category/score	**[text classification]**
DNA sequence →	coding sequence	**[pattern recognition]**
text sequence in Italian →	text sequence in English	**[text translation]**



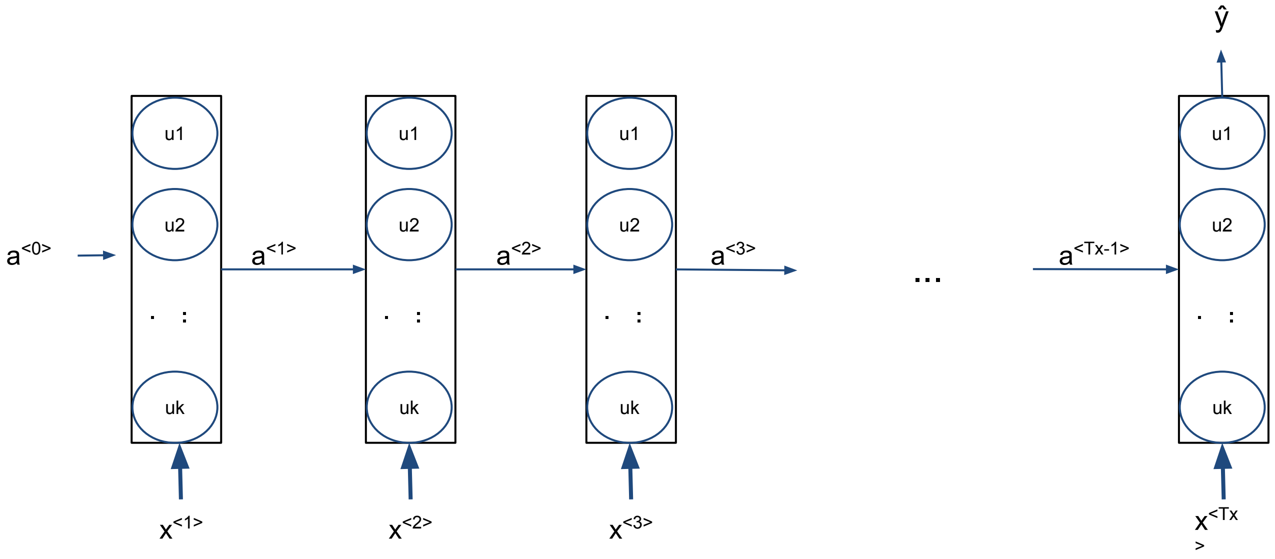
RNN: many-to-many architecture



- e.g. entity recognition
- (as) **many inputs** (words/sequences) are mapped to (as) **many outputs** (e.g. labels)

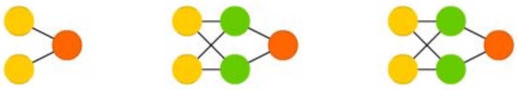


RNN: many-to-one architecture

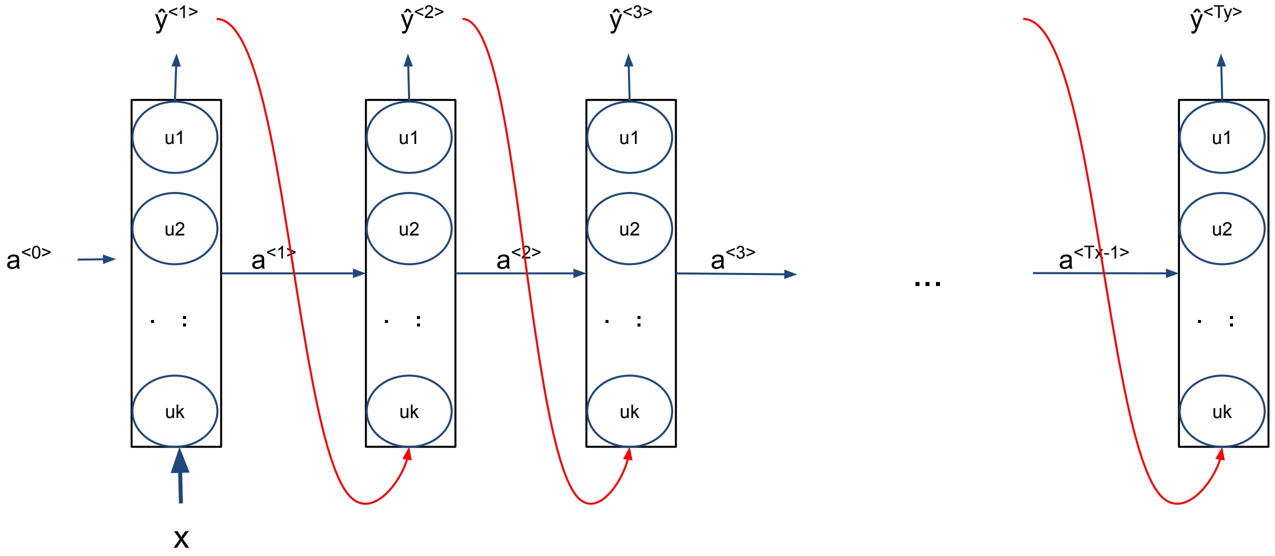


- e.g. **text classification**: reviewer report (input text) classified as accept, minor revisions, major revisions, reject (categories)
- **many inputs** (words in the reviewer report) are mapped to **one output** (category)

e.g. "The research problem is very important and was treated fine by the researches. The objective is clear and conclusions are supported by the results and methods used. Indeed no one has dealt with this matter before. So, it's a novelty. The figures are great also the tables."

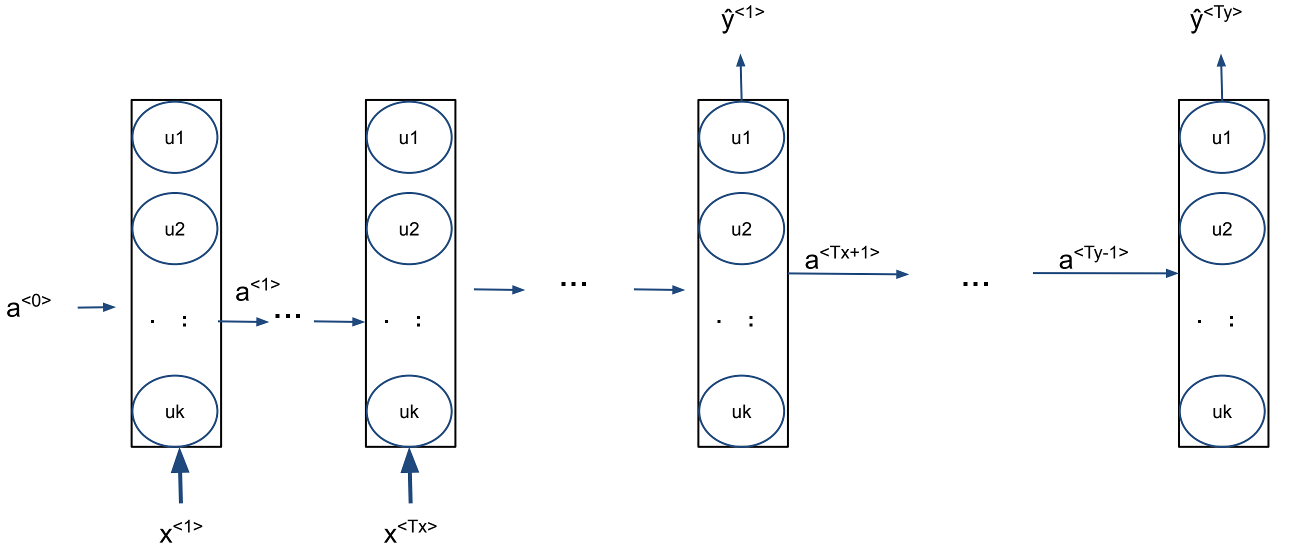


RNN: one-to-many architecture



- e.g. sequence generation: input the musical genre (one integer) to generate a song (sequence of notes)
- **one input** (integer/string) are mapped to **many outputs** (the sequence of notes in the song)
- generated notes at "t-1", together with $a^{<t-1>}$ are input of layer "t"

RNN: many-to-many* architecture



- e.g. machine translation: input sequence Tx ≠ output sequence Ty (Italian text → English text)
- **encoder**: RNN that processes the input text
- **decoder**: RNN that processes the translation

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

audio sequence (e.g. voice)

newspaper article

received email

webpage

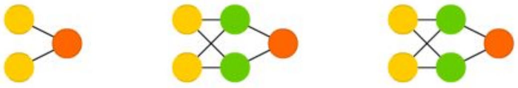
speech/text recognition

input “x”

model

output “y”

: what was written/said/recorded in the input text?



NLP: language models

<u>Speech recognition</u>:

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

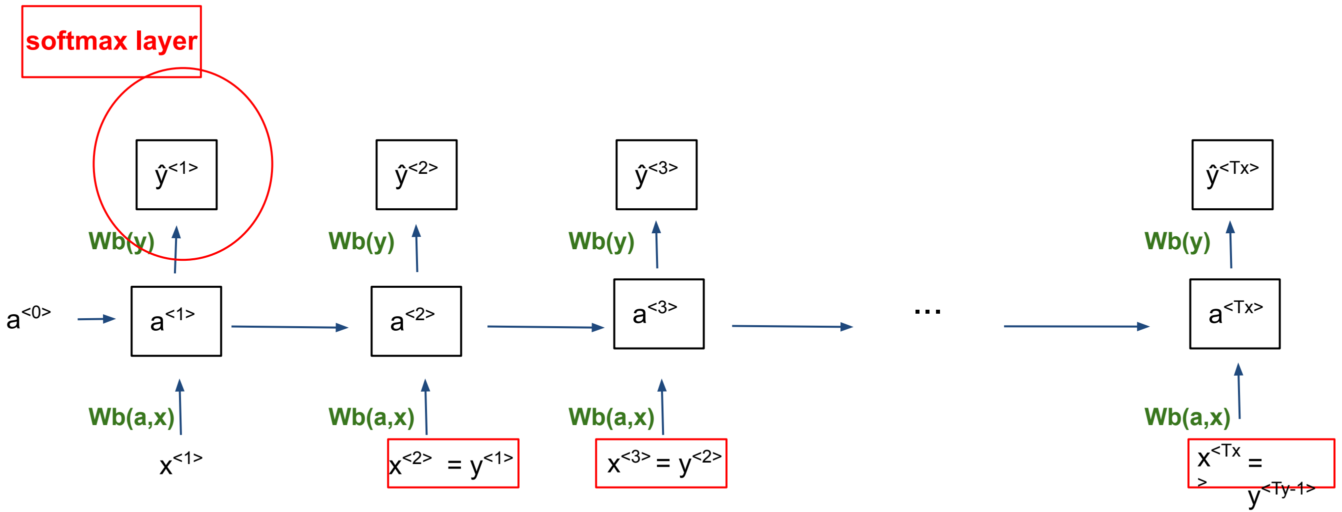
P(The red dear is a ruminant) =

P(The red deer is a ruminant) =

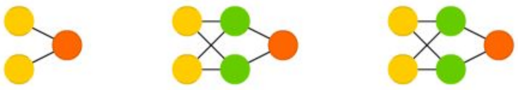
RNN models for speech/text recognition

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)

RNN models for speech/text recognition



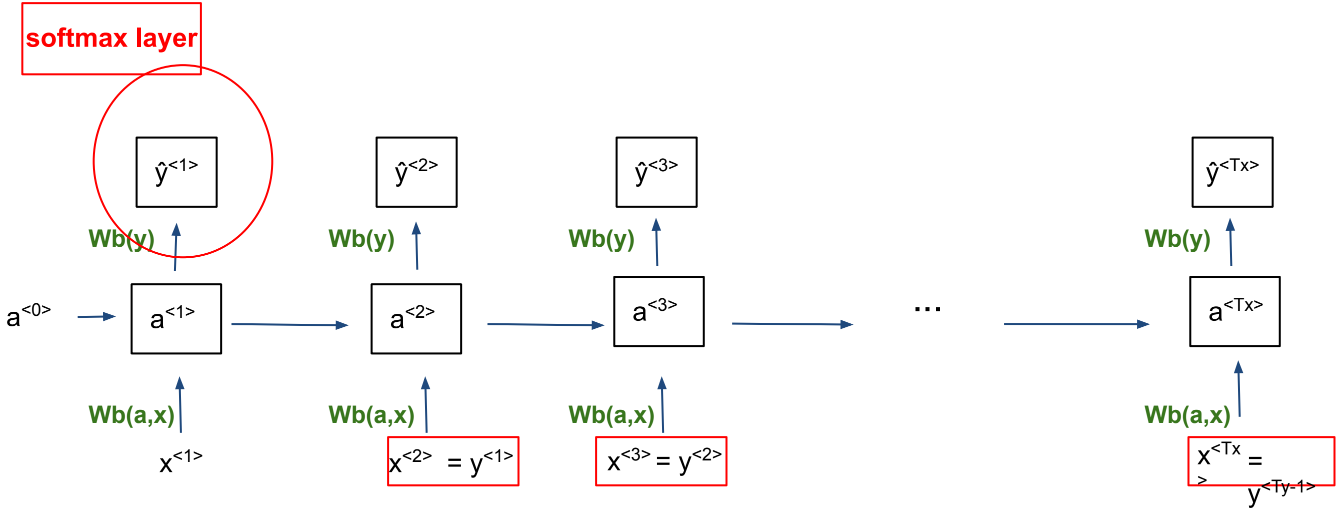
$$\begin{cases} a^{<t>} = g\left(W_{aa} \cdot a^{<t-1>} + W_{ax} \cdot x^{<t>} + b_a\right) \\ \hat{y}^{<t>} = g\left(W_{ya} \cdot a^{<t>} + b_y\right) \end{cases}$$



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



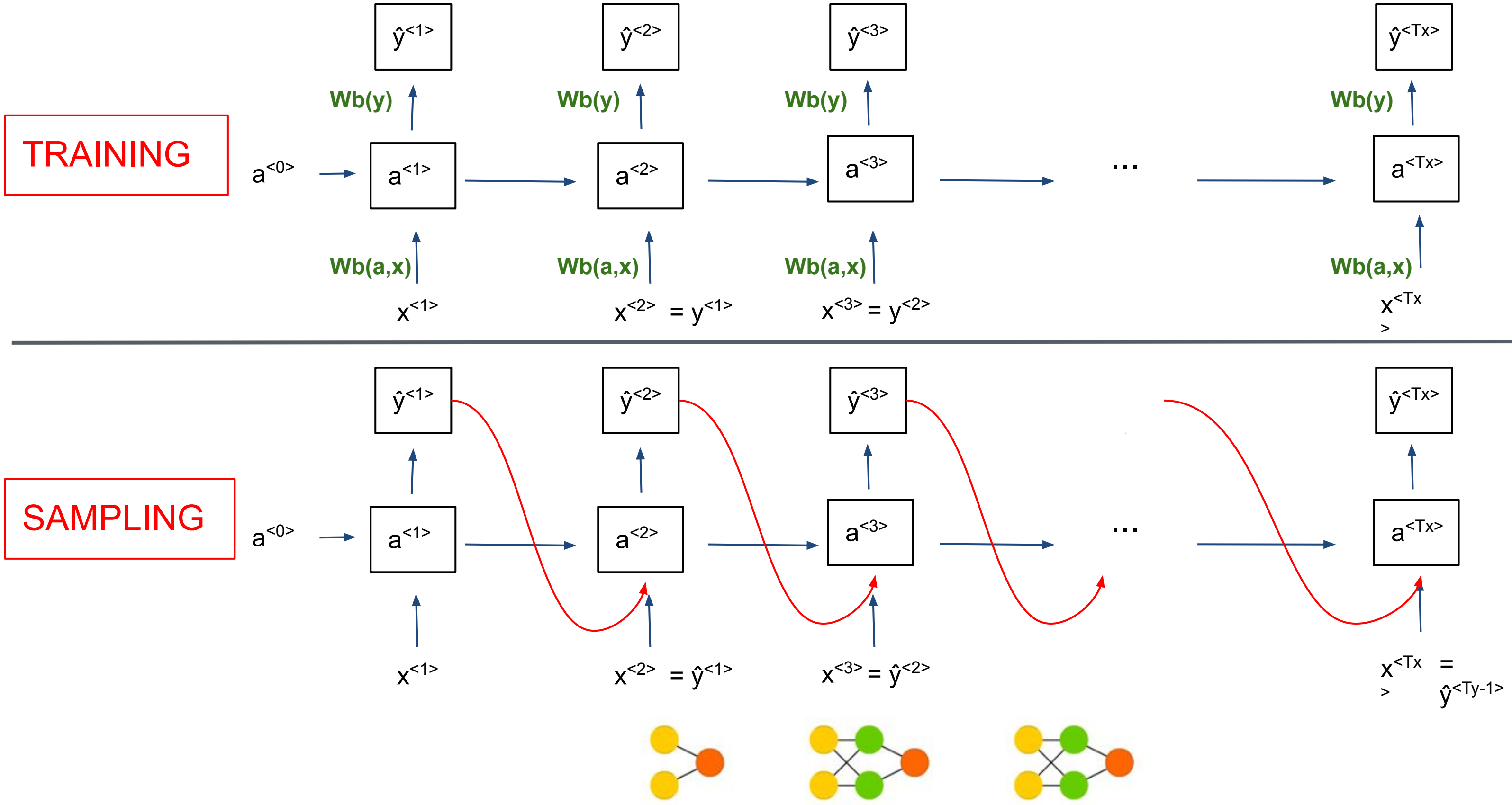
$$\begin{cases} \mathcal{L}(\hat{y}^{<t>}, y^{<t>}) = -\sum_i y_i^{<t>} \cdot \log(\hat{y}_i^{<t>}) \\ L = \sum_t \mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) \end{cases}$$

(softamx 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 the **sample new sequences** (e.g. sequence/text generation)

Sampling new sequences from a trained RNN



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
- 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

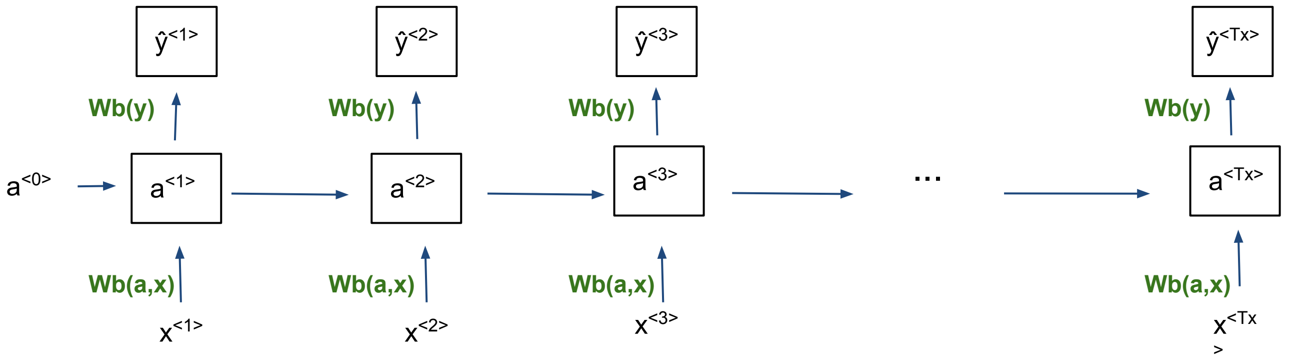
A problem of memory loss

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)

Vanishing gradients with basic RNN

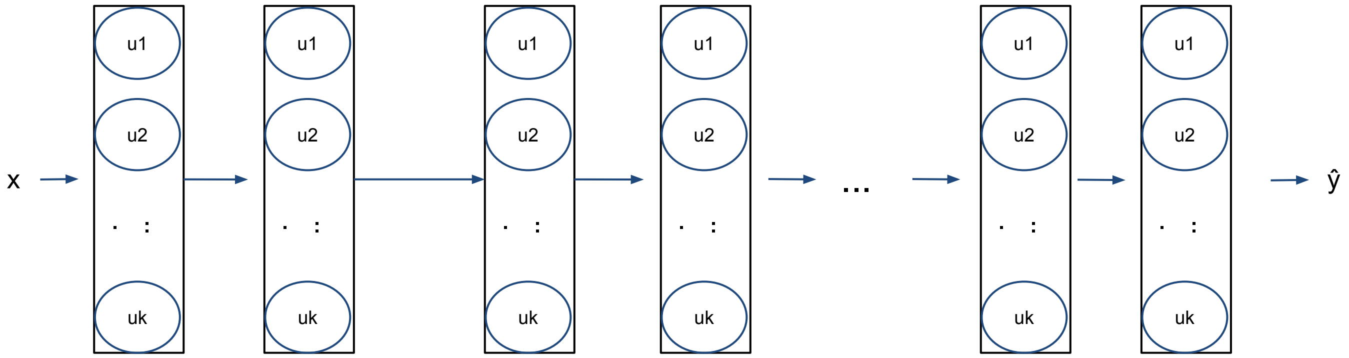
The red deer, that live in the woods, graze on ... and mate during … at temperate latitudes … is a ruminant
The red deers, that live in the woods, graze on ... and mate during … at temperate latitudes … are ruminants

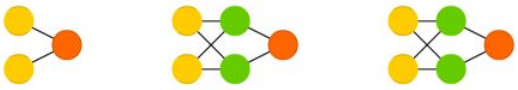


The basic RNN model is not good at capturing **distant relationships** in the sequence

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?

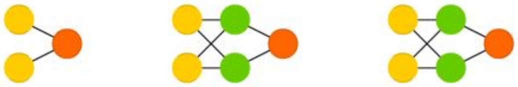
Some basic intuition:

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

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

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

$$\mathbf{w_1 * x_1 + w_2 * x_2 \dots w_m * x_m} \quad \text{(with m = n. of features)}$$



Why do gradients vanish?

Some basic intuition:

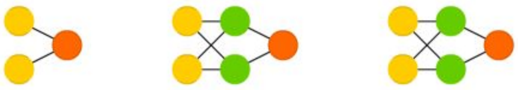
- a deep NN can be thought of as a series of matrix products;
- simplifying further:

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

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

$$\mathbf{w_1 {*} x_1 + w_2 {*} x_2 \ldots w_m {*} x_m} \quad \text{(with m = n. of features)}$$

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



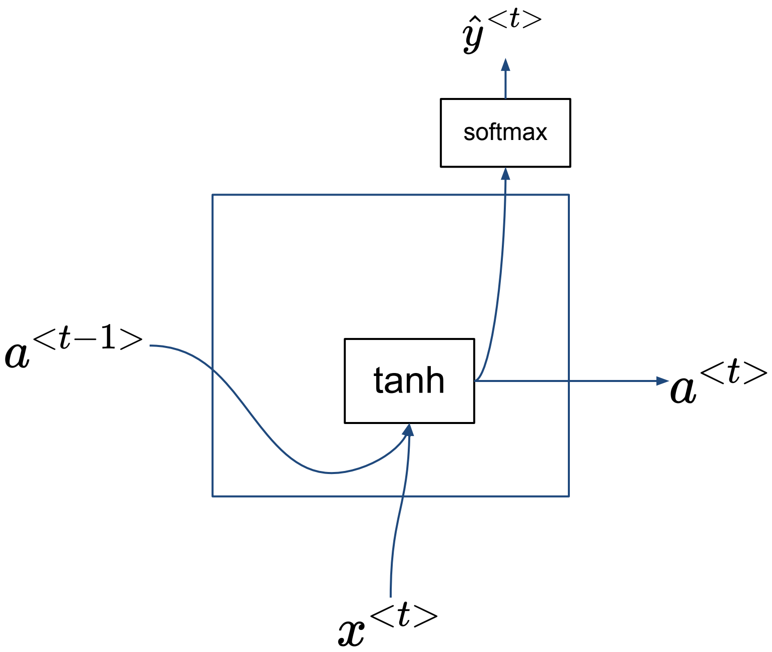
Gated Recurrent Unit (GRU)

Memory pills!

<u>Filippo Biscarini</u>
Senior Scientist
CNR, Milan (Italy)

<u>Nelson Nazzicari</u>
Research fellow
CREA, Lodi (Italy)

RNN unit



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

RNN unit - GRU (gated recurrent unit)

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

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

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

<u>Introducing the gate</u>

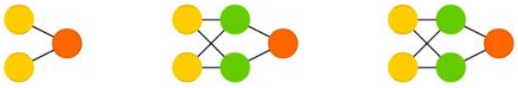
$$\Gamma_u = \sigma\left(W_{uu} \cdot c^{<t-1>} + W_{ux} \cdot x^{<t>} + b_u\right)$$

RNN unit - GRU (gated recurrent unit)

- in a GRU unit we calculate a value $\mathbf{c}^{\sim<t>}$
- we may then potentially **update** the value in the memory cell by replacing $\mathbf{c}^{<t>}$ **with** $\mathbf{c}^{\sim<t>}$
- the value of the **gate** Γ_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)

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

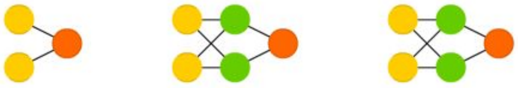
$$\tilde{c}^{<t>} = \tanh\left(W_{cc} \cdot c^{<t-1>} + W_{cx} \cdot x^{<t>} + b_c\right)$$

<u>Introducing the gate</u>

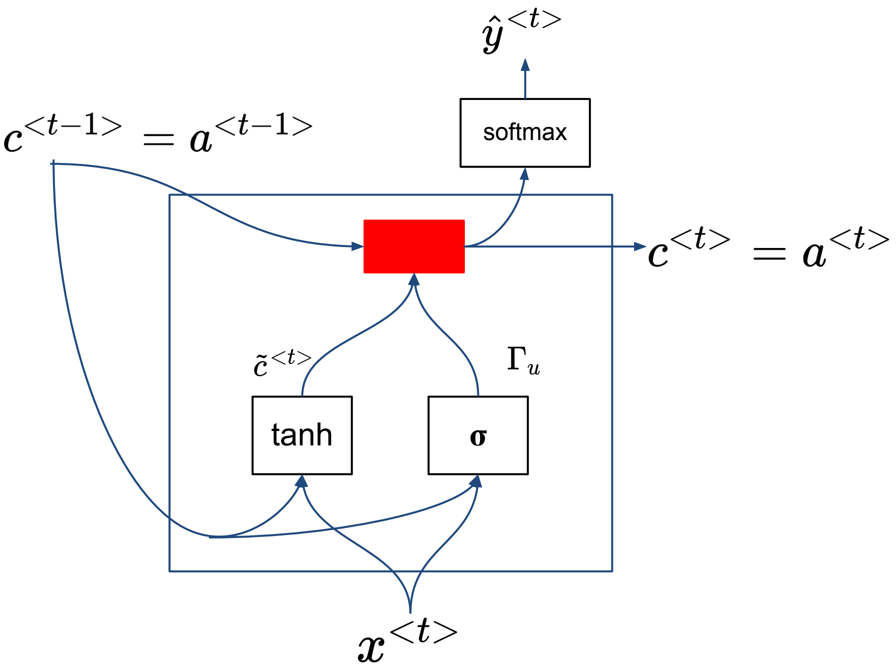
$$\Gamma_u = \sigma\left(W_{uu} \cdot c^{<t-1>} + W_{ux} \cdot x^{<t>} + b_u\right)$$

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

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)

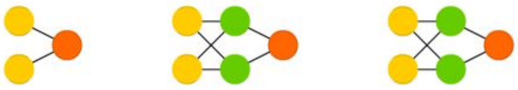
$$c^{<t>} = a^{<t>}$$

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

Introducing the gate

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

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



GRU units

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

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

GRU unit

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

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

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

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

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

LSTM unit

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

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

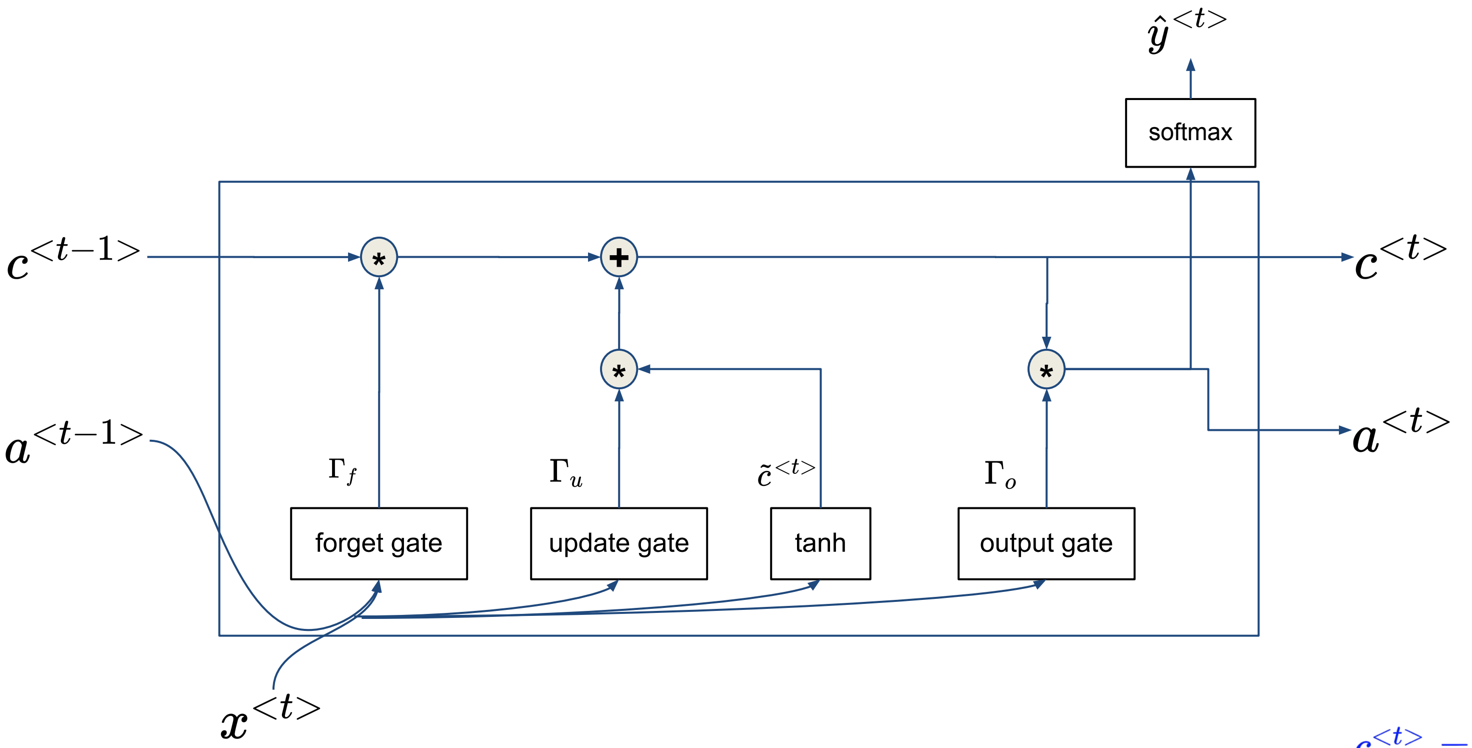
$$\Gamma_f = \sigma\left(W_{ff} \cdot a^{<t-1>} + W_{fx} \cdot x^{<t>} + b_f\right)$$

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

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

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

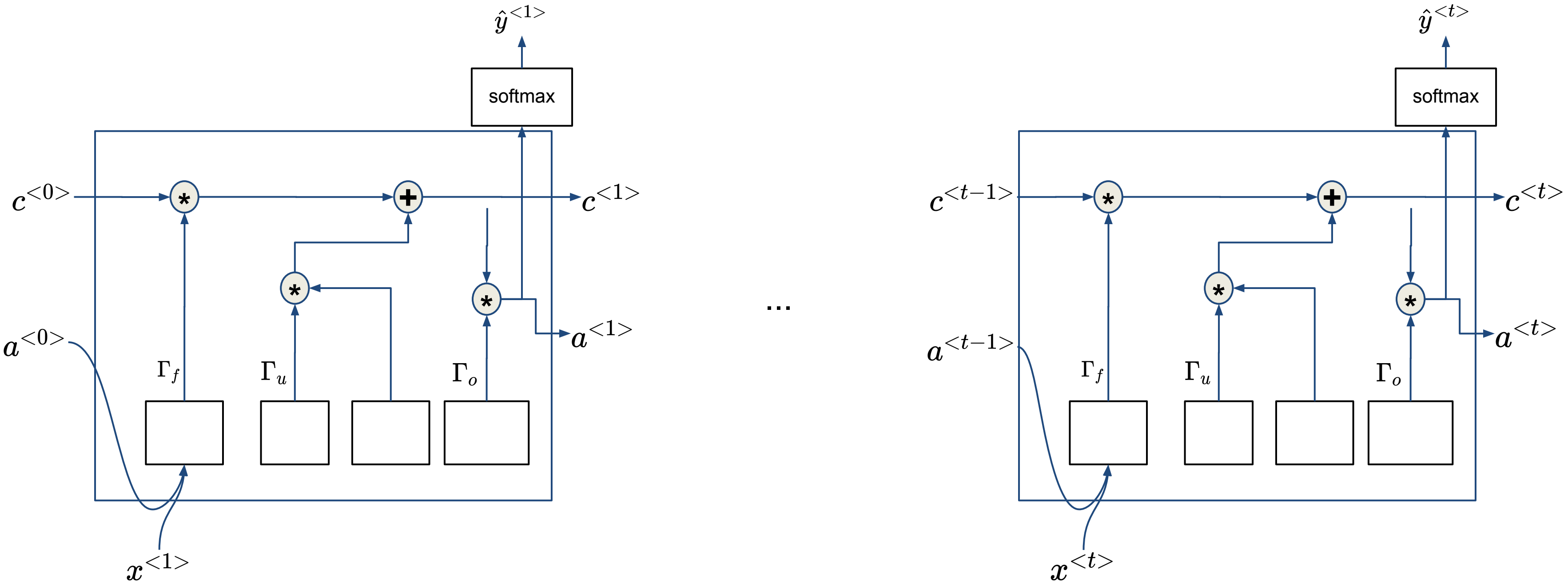
LSTM unit - schematic representation



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

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

LSTM units: with the memory flow



Bidirectional RNN

Back from the future!

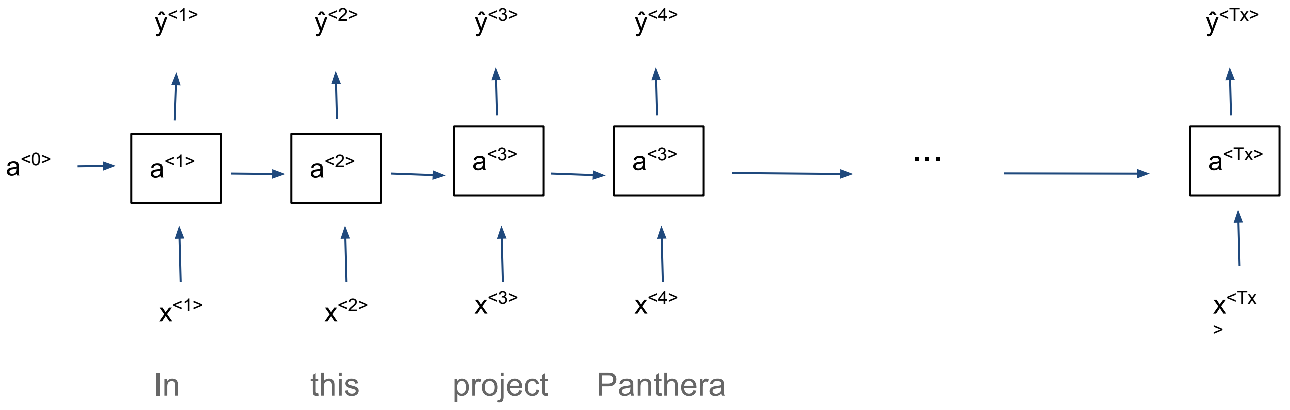
Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)

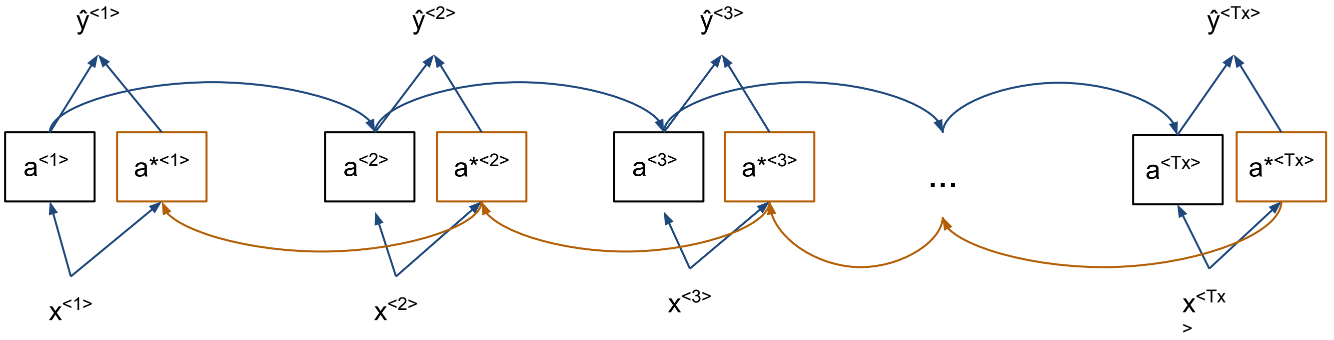
Using information from the “future”

“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 $a^{<Tx>}$ (as usual)
- the **backward sequence** will begin from the end ($a^{*<Tx>}$) 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

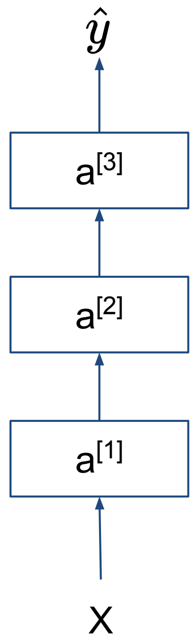
Not complex enough? Let’s stack RNN layers!

Filippo Biscarini
Senior Scientist
CNR, Milan (Italy)

Nelson Nazzicari
Research fellow
CREA, Lodi (Italy)

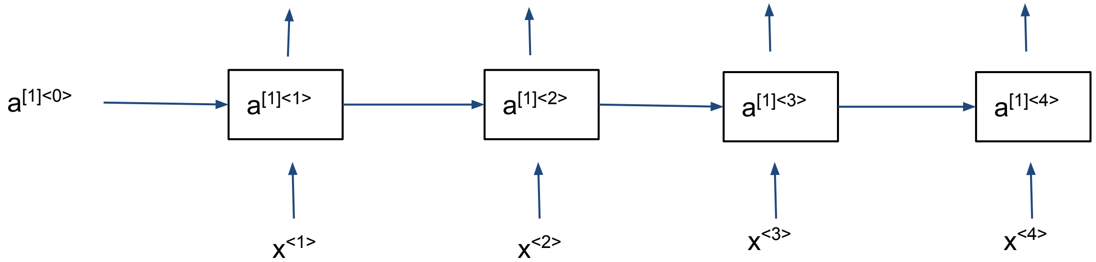
Deep RNN

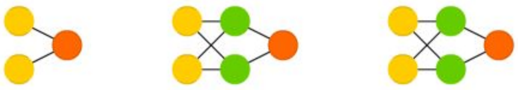
Standard multilayer NN



Basic RNN model

- <t>: time dimension (each element of the sequence) → units inside (standard, GRU, LSTM: multiple units/nodes per <t>)
- [l]: layers (stack of RNN layers)

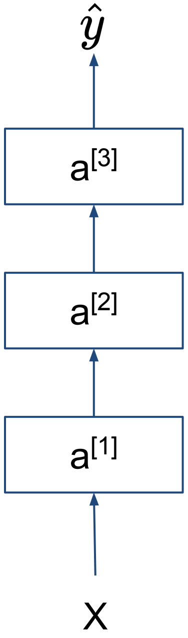


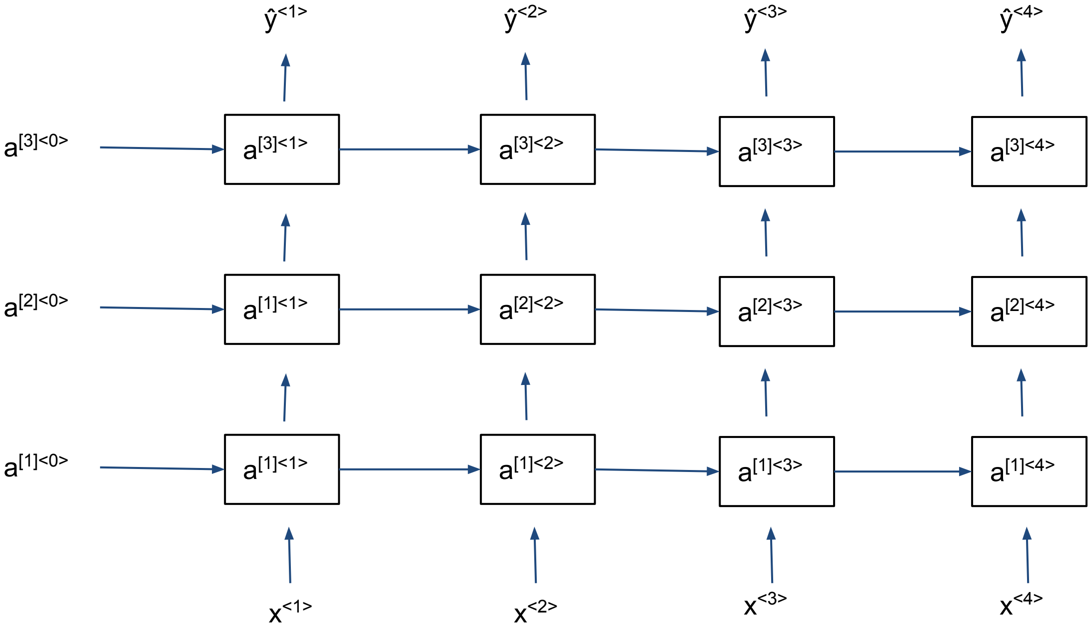


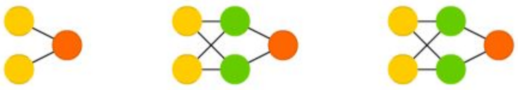
Deep RNN

Multilayer RNN model

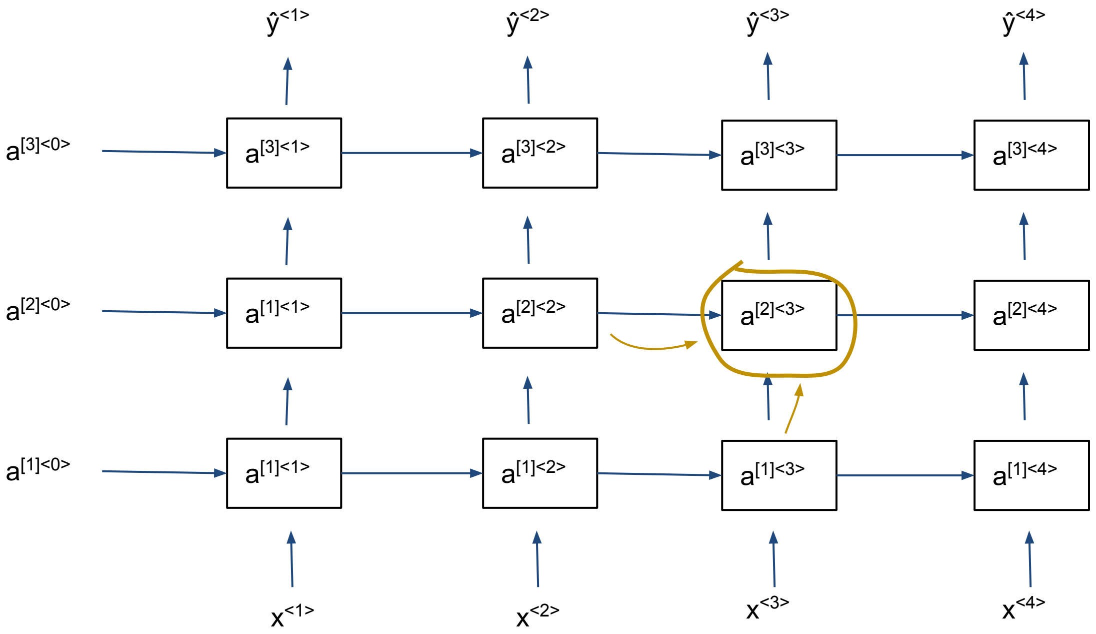
Standard multilayer NN







Deep RNN - example calculation



From basic RNN models

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

RNN models

- demonstration

  → day5_code01 RNN-1 architectures.ipynb
  → day5_code02 RNN-2 architectures.ipynb
  → day5_code03 RNN-3 architectures.ipynb