# RNN: recurrent neural networks (part 1)

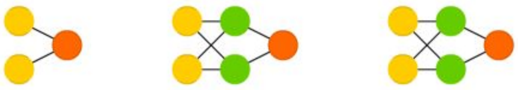
## Time (order) matters: sequence (longitudinal) data


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

<u>Nelson Nazzicari</u>
*Senior scientist*
*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

Pattern (entity) recognition problem

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

**input representation**

x: statisticians are beautiful human beings

$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ $x^{<4>}$ $x^{<5>}$

<u>Vocabulary</u>

$$\begin{bmatrix} \text{are} \\ \text{beautiful} \\ \text{beings} \\ \text{human} \\ \text{statisticians} \end{bmatrix}$$

OHE: [one-hot encoding]

# Sequence models - data representation

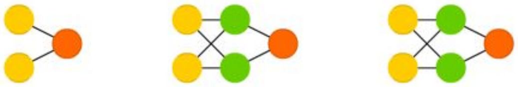
Pattern (entity) recognition problem

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**



# Sequence models - data representation

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

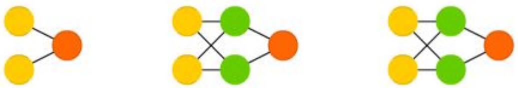
x: in the impenetrable forest there are populations of *Panthera leo* and *Loxodonta africana*

<u>Vocabulary / dictionary</u>

a
Aarhus
...
africana
are
...
Leo
…
…
Panthera
…
Wagyu
…
zebra
…
Zürich

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

Can you see a potential problem with this OHE representation of the data?



# Building a NN model for sequence data

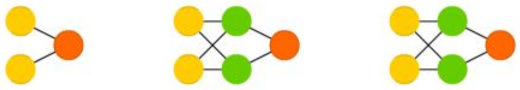
## From dense NNs to RNNs

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

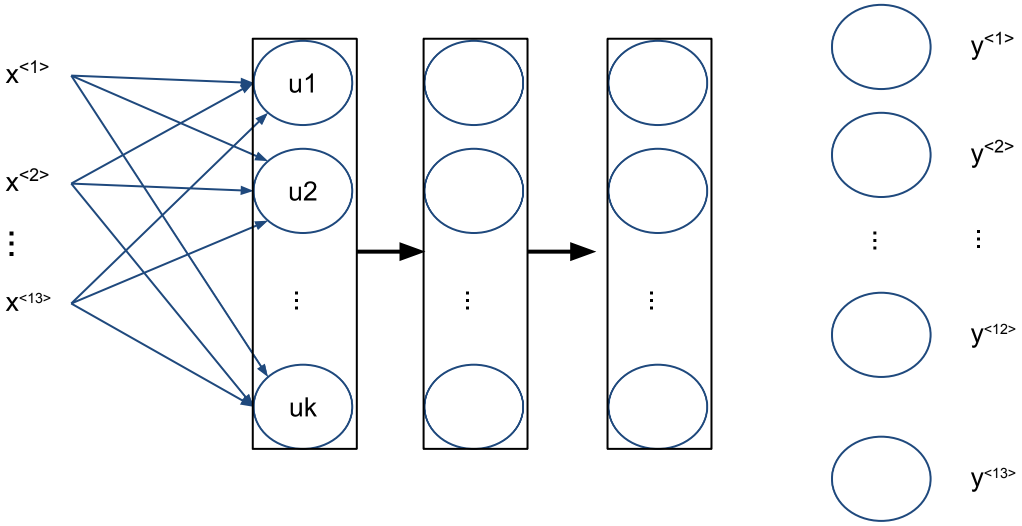
Nelson Nazzicari
*Senior scientist*
*CREA, Lodi (Italy)*

# A neural network model for word/sentence recognition

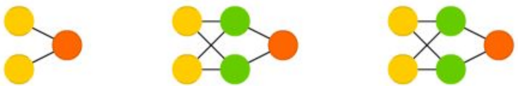
- text data + vocabulary → **data representation** (sequence of 1-hot-encoded vectors and labels)
- **objective**: find (approximate) function that maps OHE vectors to labels (x → y: labels can be “scientific name” or not, “name of persons” or not, singular/plural verbs etc.)
- y = f(x)
- which neural network **architecture**? Shall we try a dense neural network?



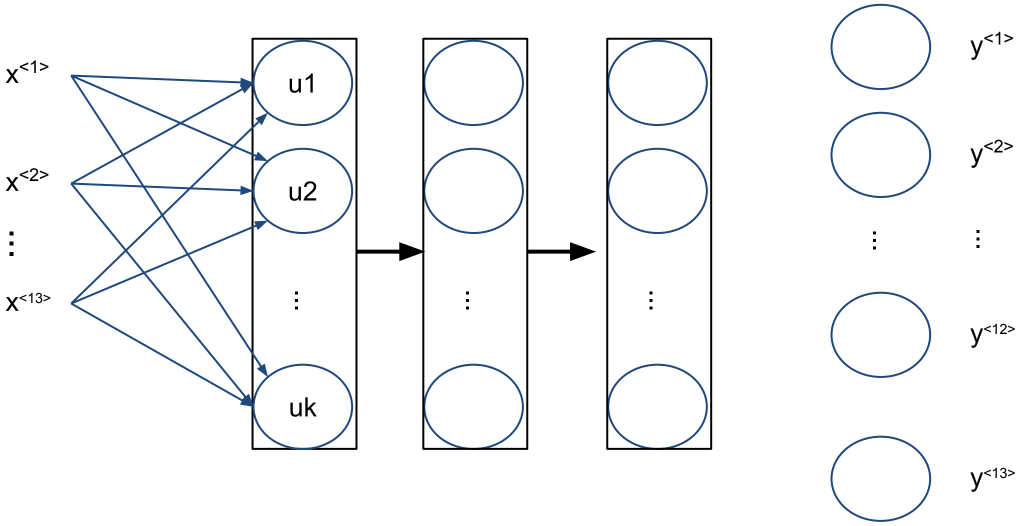
# Let’s try a standard (dense) neural network




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



# Let’s try a standard (dense) neural network



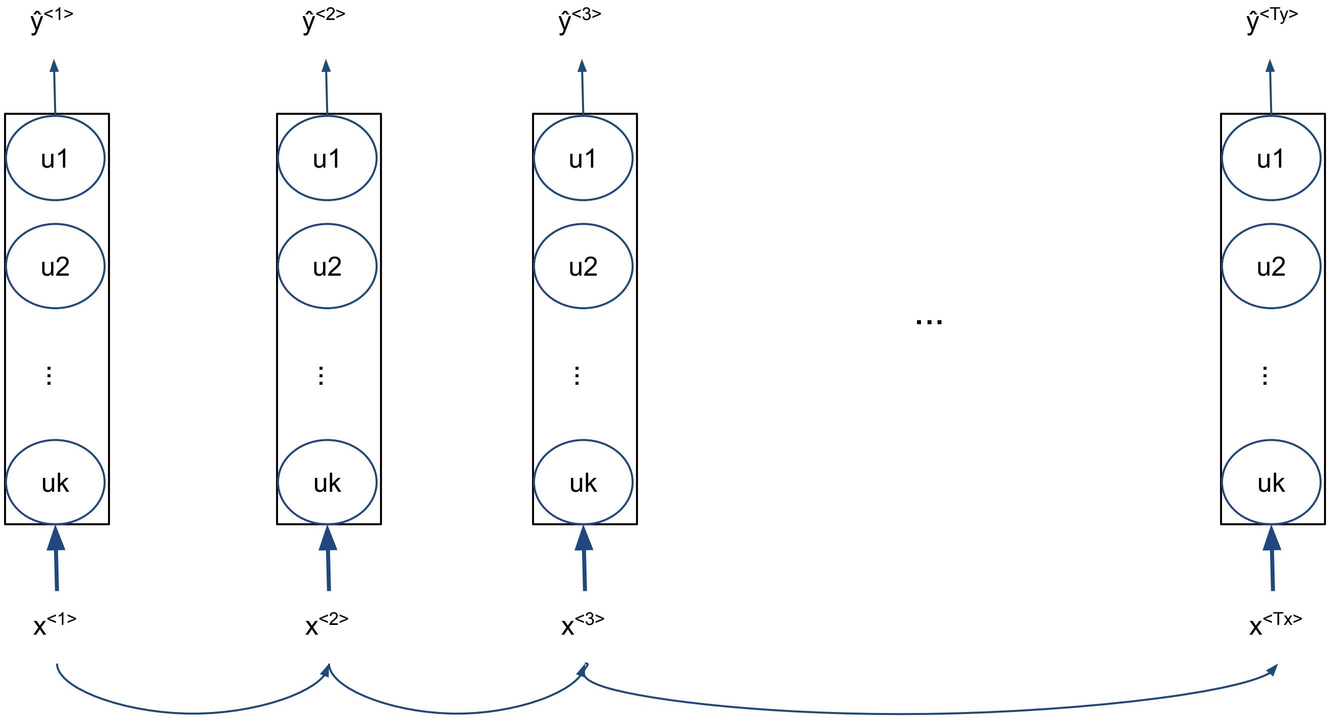

**T = 13** words, **3** hidden layers (**u**: units), **one** output layer (13 labels)

**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]

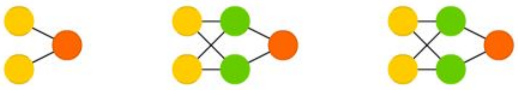
similar to CNN, we would like learning to generalize from one part of the image (sequence) to other parts of the image (sequence)

# Recurrent Neural Network (RNN)




**Sequentiality**
**+**
**“Activation trick”**

**sequentially** (word by word)



# Recurrent Neural Network (RNN)

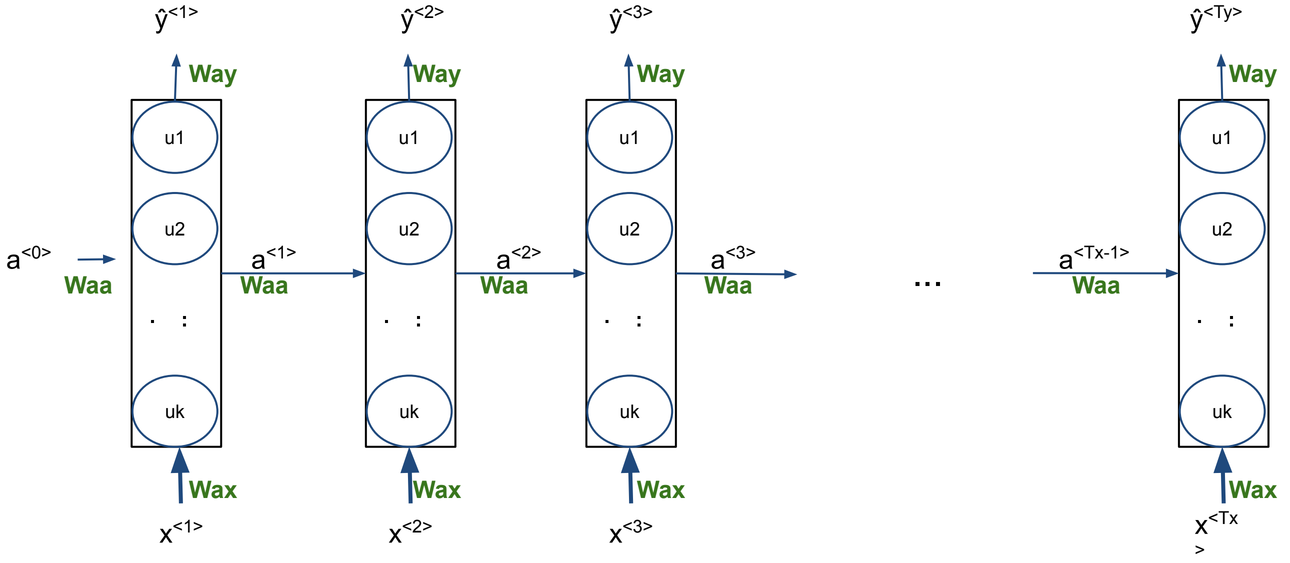
- words (sequences) are **analysed sequentially**, from left to right (or from top to bottom etc.)
- each time, the transformed information from the previous word/sequence (**activation values**) is passed on to the next word/sequence → transferring learning along the sequence!
- weakness: only previous 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!**
- **The different sets of parameters (Waa, Wax, Wya, ba, by) are shared along the sequence [IMPORTANT!]**

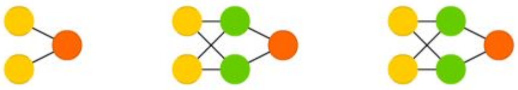
# Simple RNN: forward propagation

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

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

- **Tanh** or **Relu** for the RNN 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

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

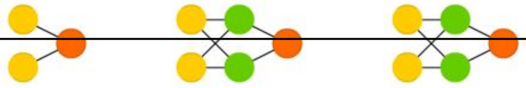
- **u**: **n. of units** (nodes) in the layer
- **m**: n. of features (**vocabulary size**)
- Wax(u,m): u nodes, m features
- Waa(u,u): input a(u,1) * u “new” nodes (from initialization onward)
- **sigmoid activation**: binary classification (name / no-name)
- for multiclass, softmax would be used, and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

# Simple RNN: let’s work out the dimensions

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

- **u**: **n. of units** (nodes) in the layer
- **m**: n. of features (**vocabulary size**)
- Wax(u,m): u nodes, m features
- Waa(u,u): input a(u,1) * u “new” nodes (from initialization onward)
- **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

**“DL: it all boils down to data, matrix multiplications, repeated and scaled with non-linear switches”** [A. R. Gosthipaty and R. Raha, 2022]



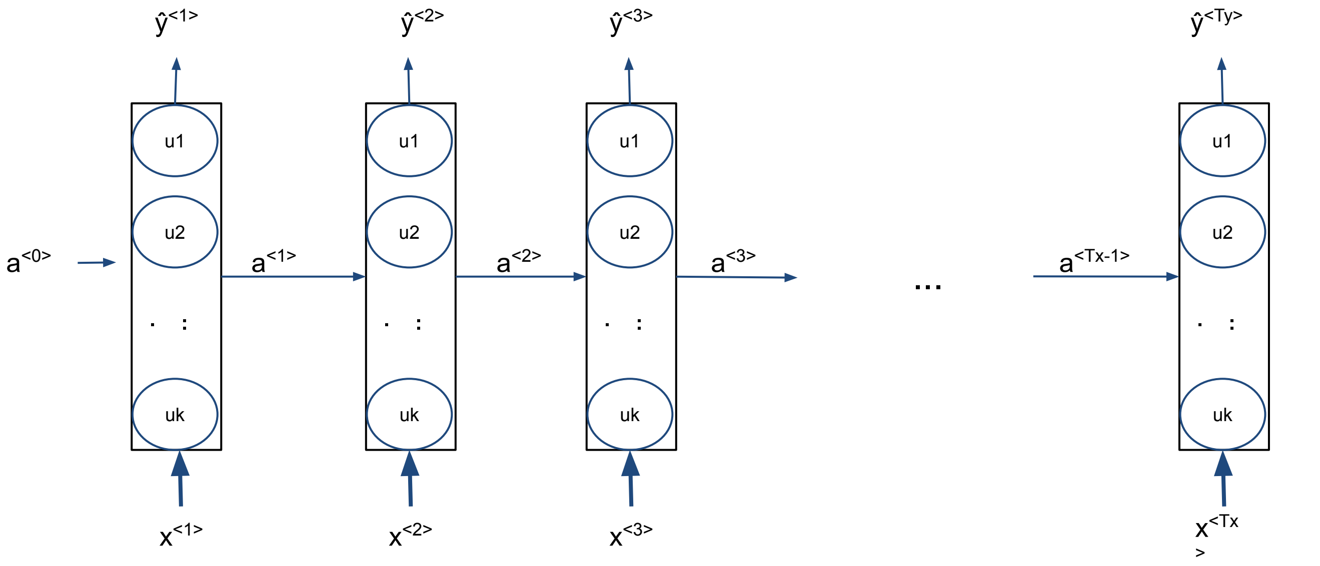
# Back to the future!

## Back propagation for RNNs

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

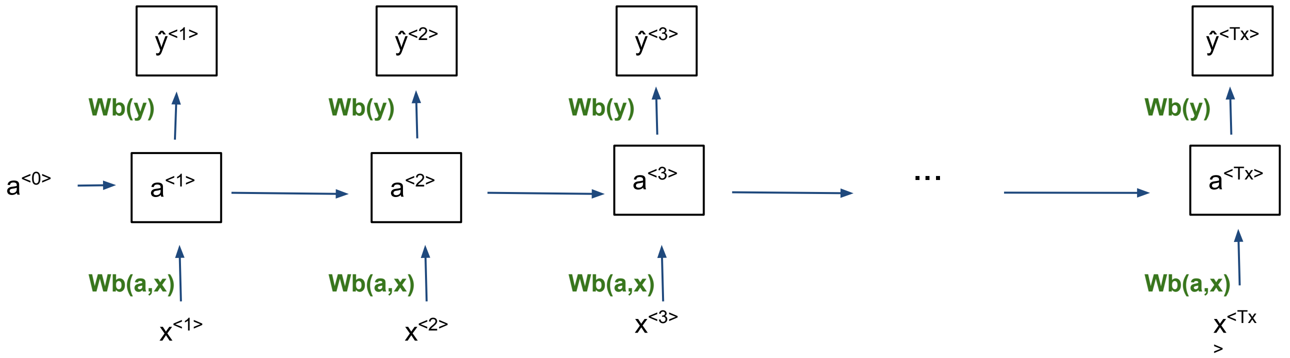
Nelson Nazzicari
*Senior scientist*
*CREA, Lodi (Italy)*

# RNN: forward and back propagation




We have our **RNN** and now we'll see what happens during **forward** and **back propagation**

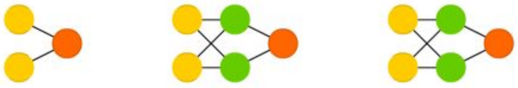
# RNN: forward and back propagation



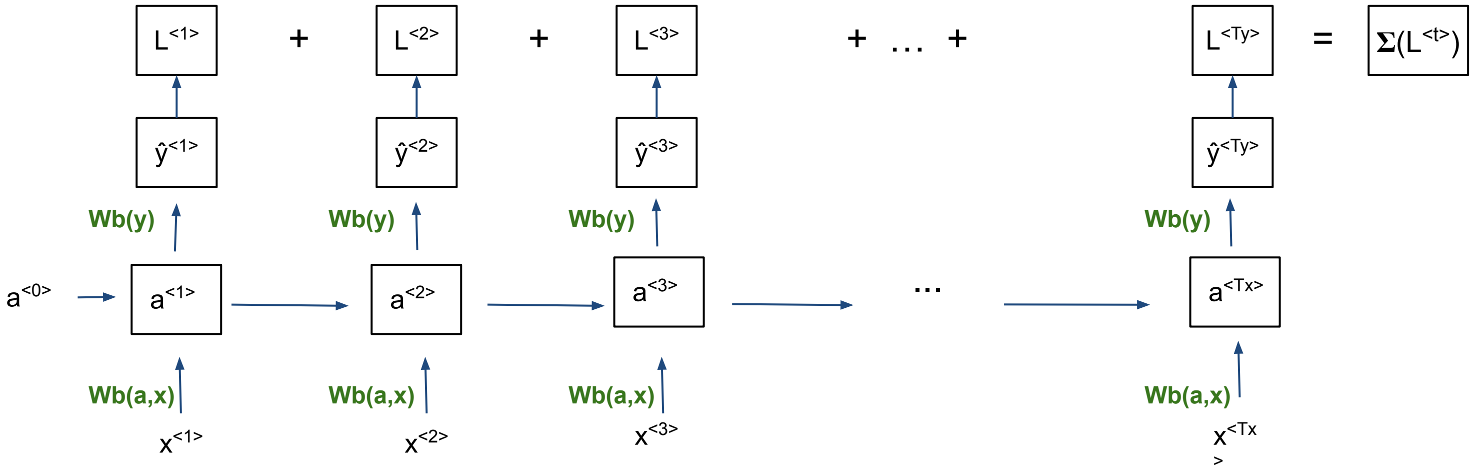

This is **forward propagation**: with **data** and **parameters (weights/coefficients)** we go through the network and obtain **predictions**

- **How do we now update the weights of the model?**

*we use here a more compact notation for the weights



# RNN: loss function



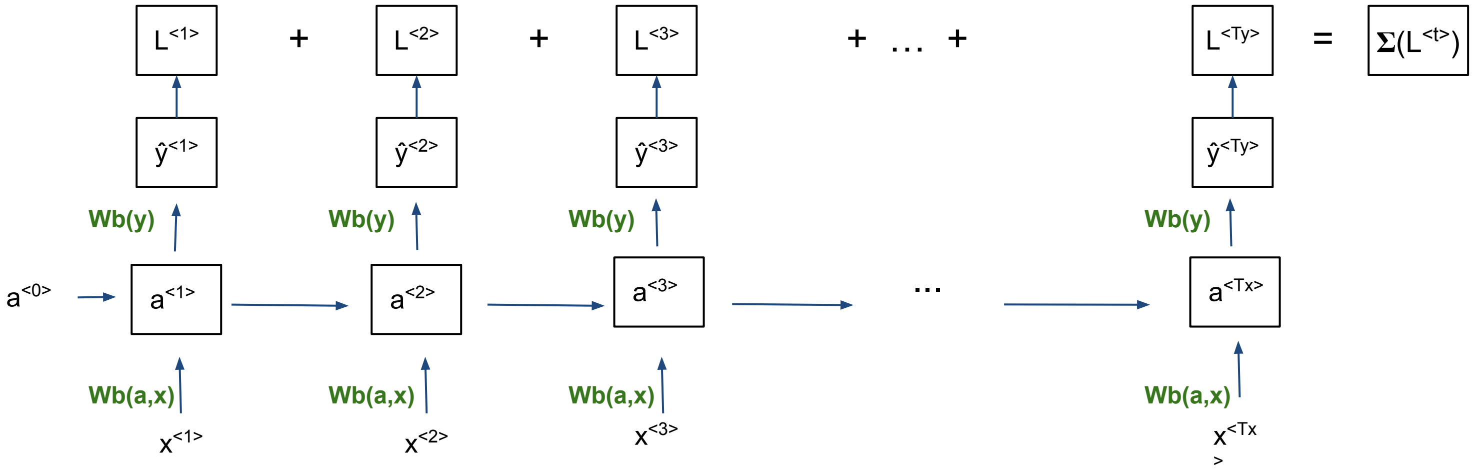

$$\mathcal{L}^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right) =$$

← loss for single word (position)

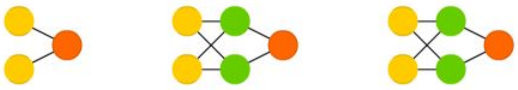
$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{Ty} \mathcal{L}^{<t>}\left(\hat{y}^{<t>}, y^{<t>}\right)$$

← loss for the whole sequence (sentence)

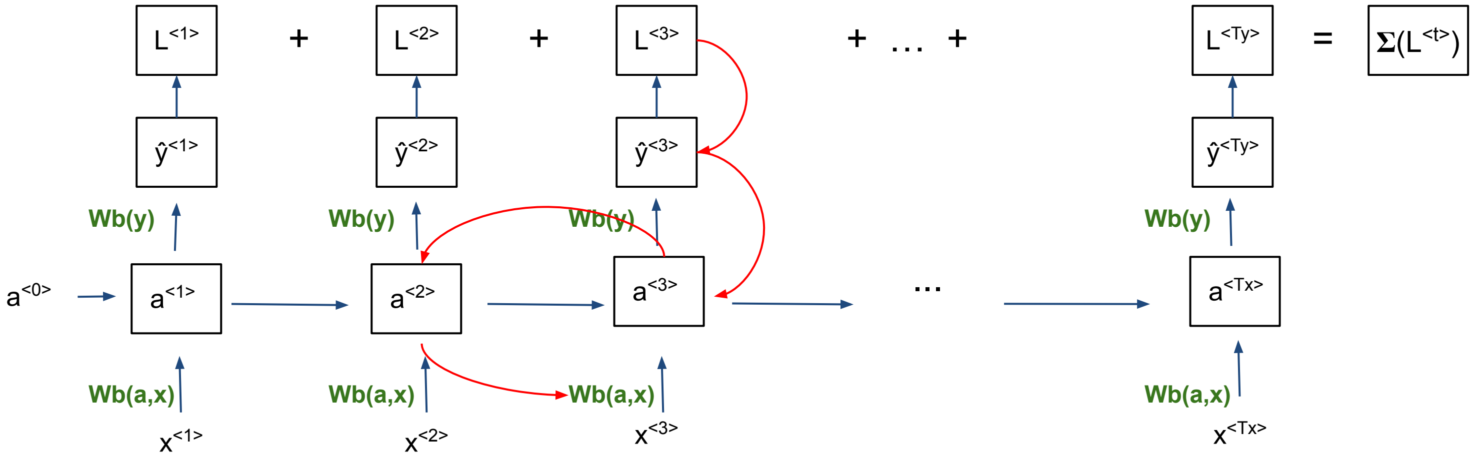
# RNN: backpropagation through time



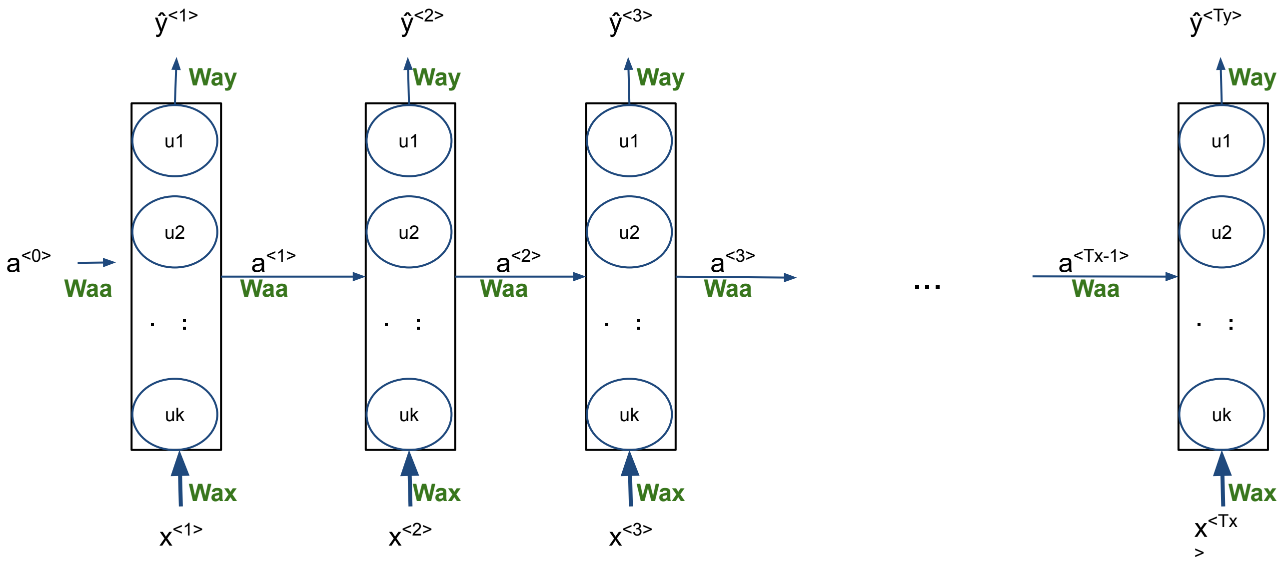

- with the **loss function** and the **partial derivatives** (with respect to model coefficients), we can move through the network and **update the coefficients** (~ gradient descent)
- **“Backpropagation through time”**: algorithm to solve RNNs (from right to left, over decreasing time indices “t”, kind of backwards in time)
- compared to backpropagation for dense NN, with RNN we need to consider also the flow of gradients over time (discretized into the $t$ elements of the sequences): one additional dimension to consider

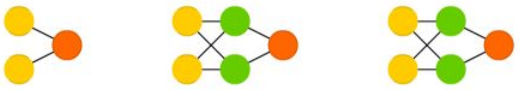


# RNN: backpropagation through time

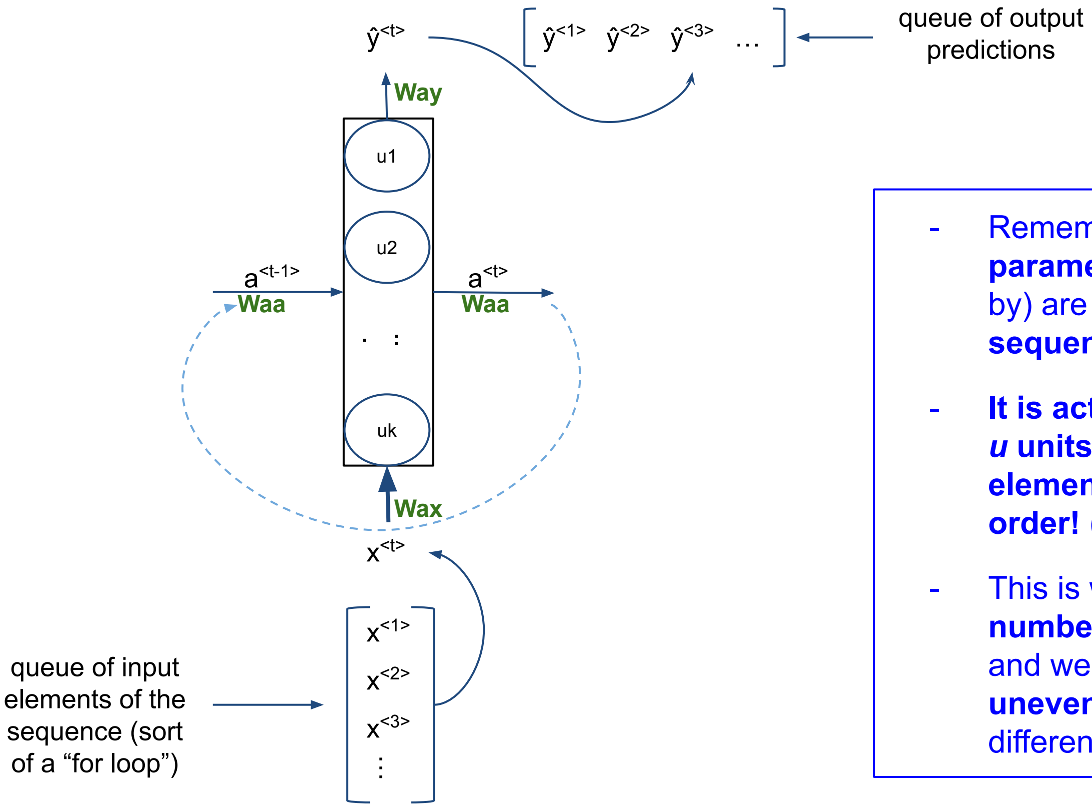


# RNN: I lied



# RNN: actual reality




- Remember that the **same sets of parameters** (Waa, Wax, Way, ba, by) are **shared across the sequence** of data
- **It is actually only one layer with *u* units that process all the elements of the sequence in order!** (irrespective of their size)
- This is why we can **reduce the number of trainable parameters** and we **don’t have problems with uneven sequences** (sequences of different length)

# Architects at work

## Different RNN architectures


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

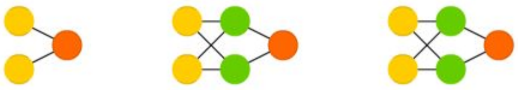
Nelson Nazzicari
*Senior scientist*
*CREA, Lodi (Italy)*

# RNNs: input / output

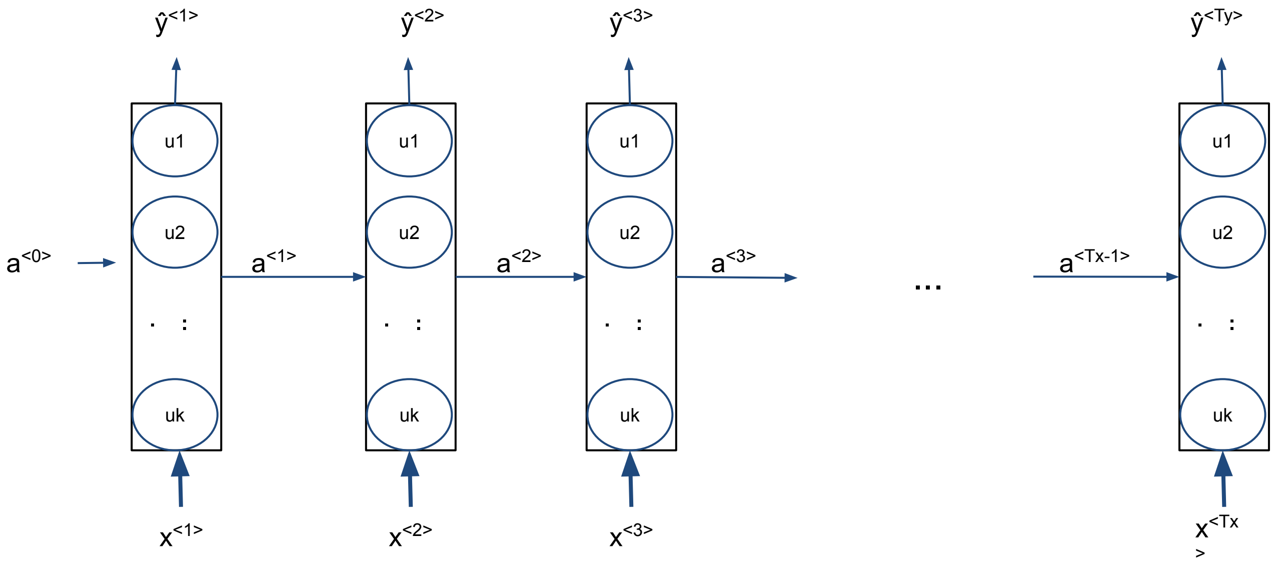
<u>So far:</u>

- 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] |

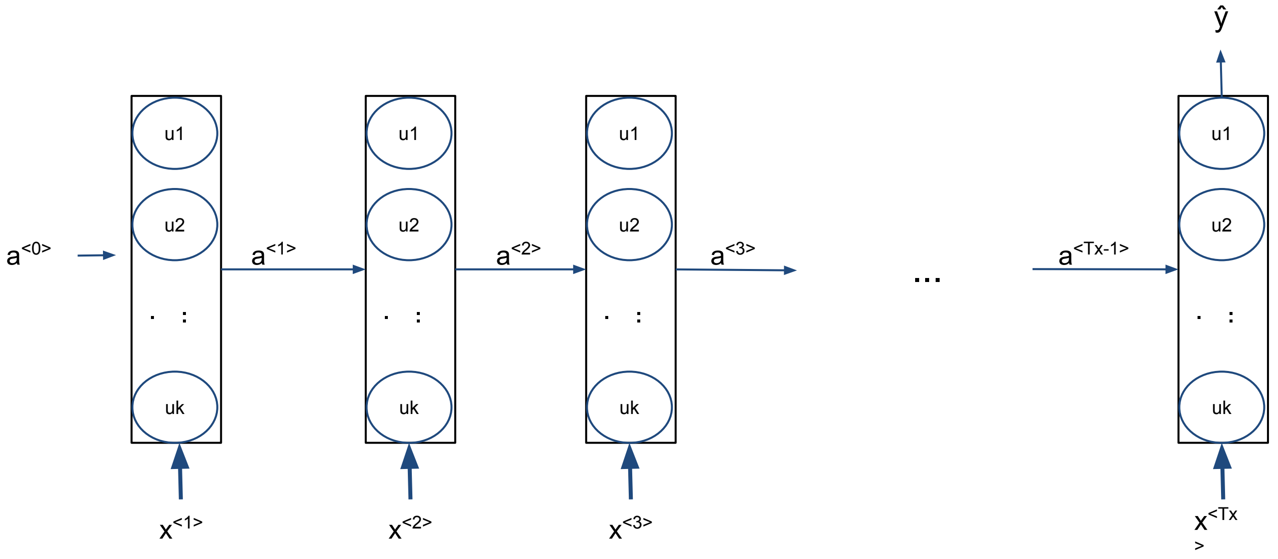


# RNN: many-to-many architecture



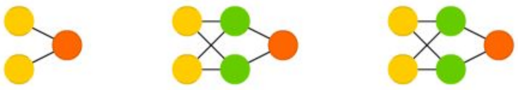

- e.g. entity recognition
- (as) **many inputs** (words/sequences) are mapped to (as) **many outputs** (e.g. labels)

# RNN: many-to-one architecture

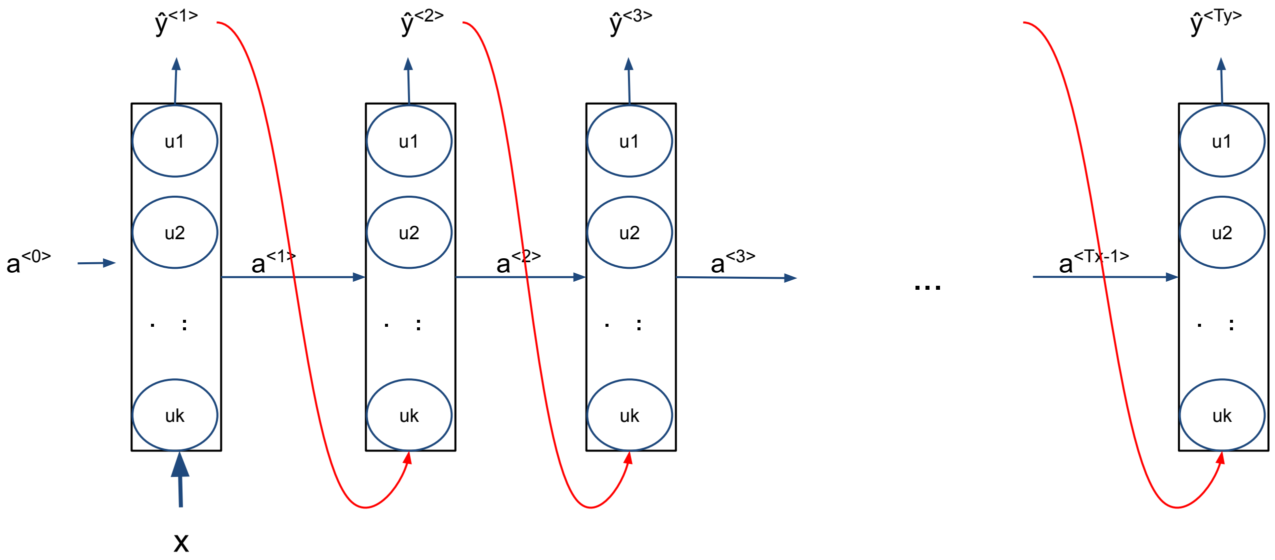



- e.g. **text classification**: reviewer report (input text) classified as accept, minor revisions, major revisions, reject (categories)
- **many inputs** (words in the reviewer report) are mapped to **one output** (category)

*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."*

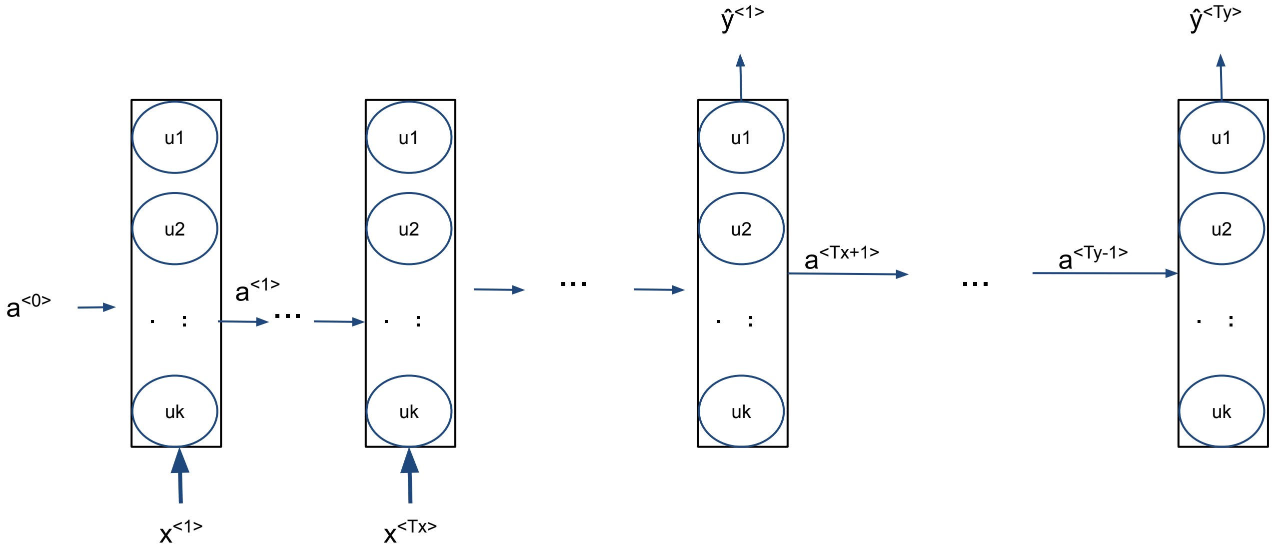


# RNN: one-to-many architecture




- e.g. sequence generation: input the musical genre (one integer) to generate a song (sequence of notes)
- **one input** (integer/string) is 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

# RNN - lab 1

- simple RNN
- RNN for forecasting

→ day4_code02 RNN-1 architectures.ipynb (up to embeddings (included))

→ day5_code02 RNN-2 time series data.ipynb (first part - up to END OF LIGHT DEMO PART (excluded))