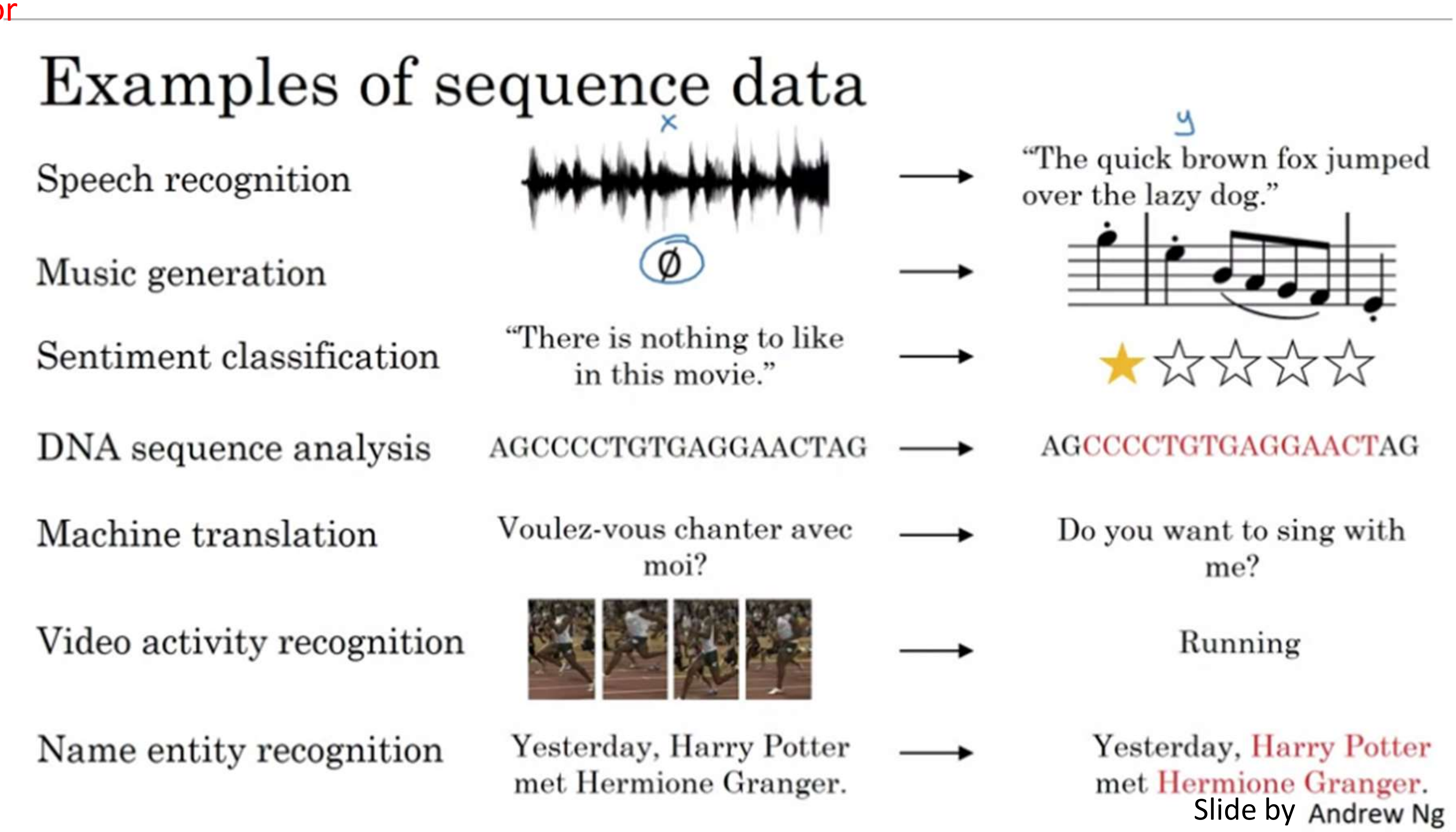
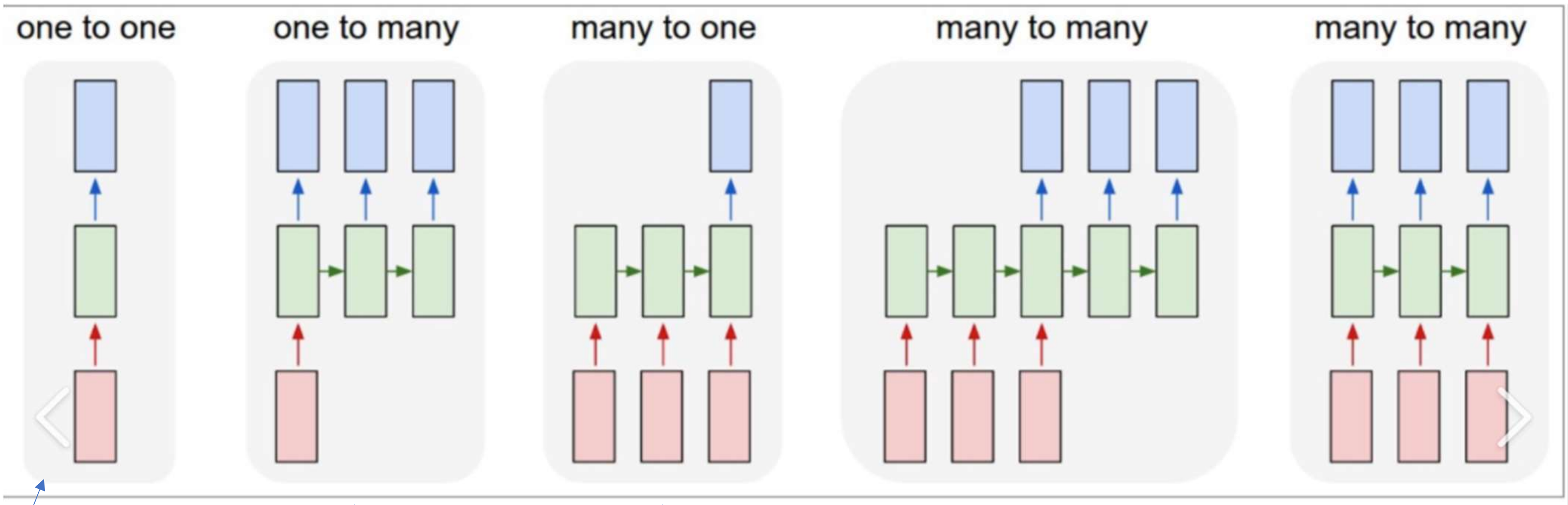
x and y can have same or different dimensions!




Language models. It was raining, I went out without my → umbrella

# Several kinds of RNN




Ex: image classifi cation

Ex: image captioning; music generation

Ex: sentiment analysis: from a sentence to a sentiment

Ex: translation

Ex: Frame by frame video classifiction

# Notation

- Example: named entity recognition
- x: Harry Potter and Hermione Granger invented a new spell

  $x^{<1>}$ $x^{<2>}$ $x^{<q>}$

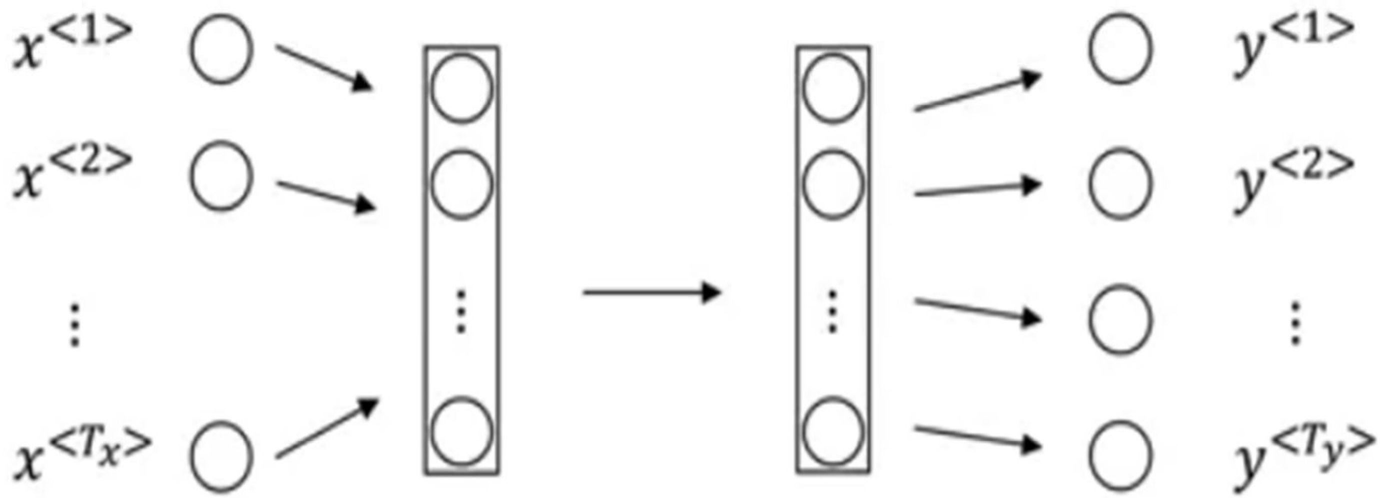
- y: 1 1 0 1 1 0 0 0 0 [if proper name]

  $y^{<1>}$ $y^{<2>}$ $y^{<q>}$

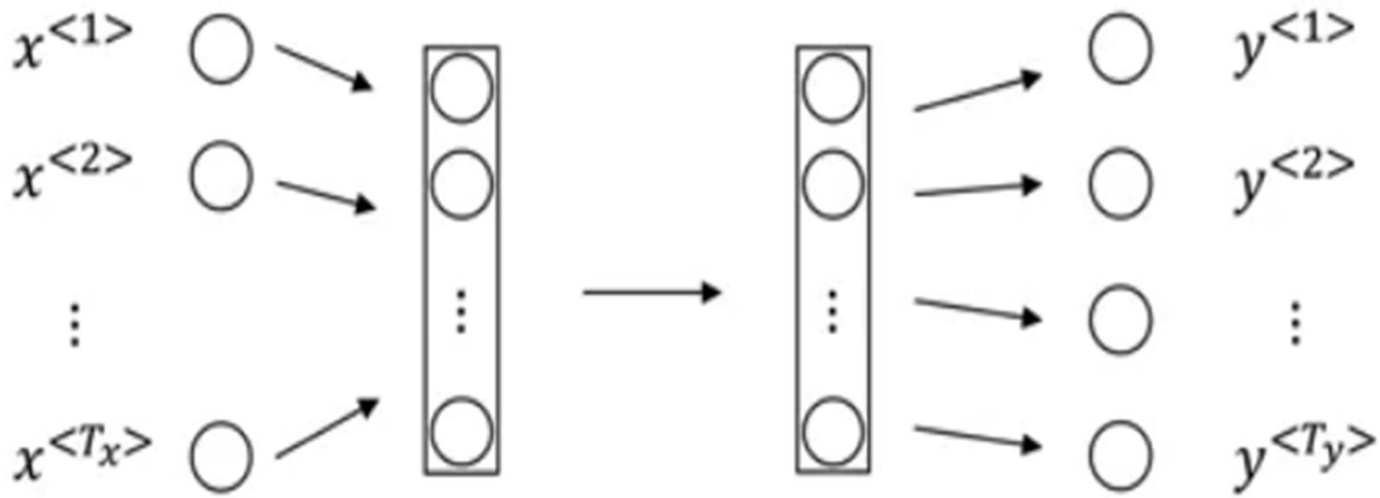
- $x^{<t>}$(or $x^{(i)<t>}$ x(i) <t>) $t^{th}$ element of the sequence (in training example i)
- Similarly, $y^{<t>}$ (or $y^{(i)<t>}$ ) $t^{th}$ element of the output sequence ( i)
- $Tx^{(i)}$ length of input sequence $x^{(i)}$. (which can be different from $Tx^{(j)}$ and from $Ty^{(i)}$)

- Goal: learn a mapping to the target output y. In a supervised manner.

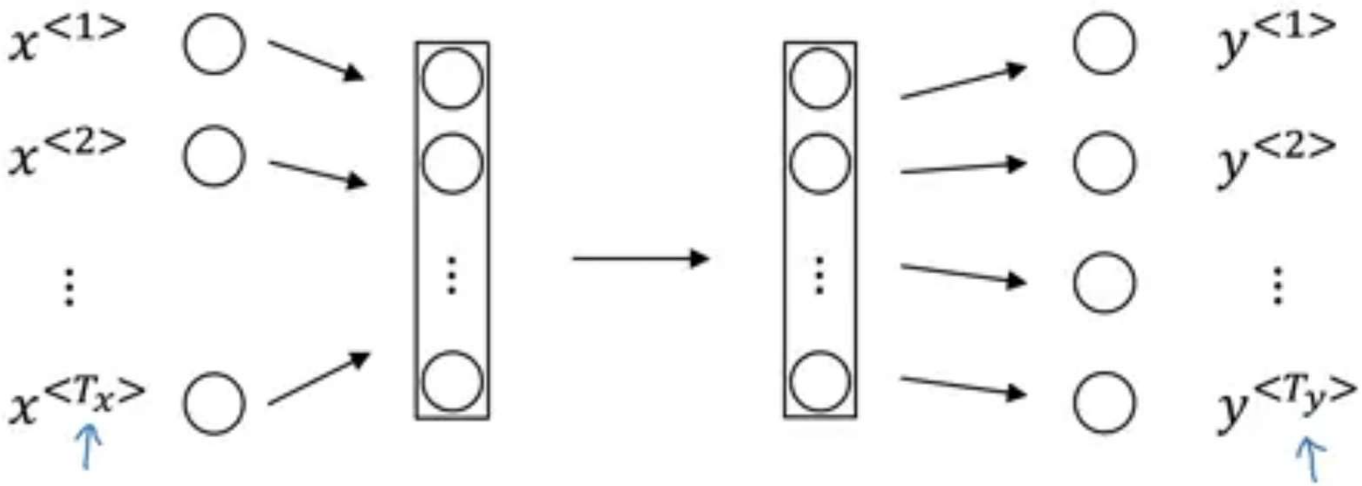
# Why not a standard network? (e.g. MLP)



# Why not a standard network? (e.g. MLP)



# Why not a standard network?



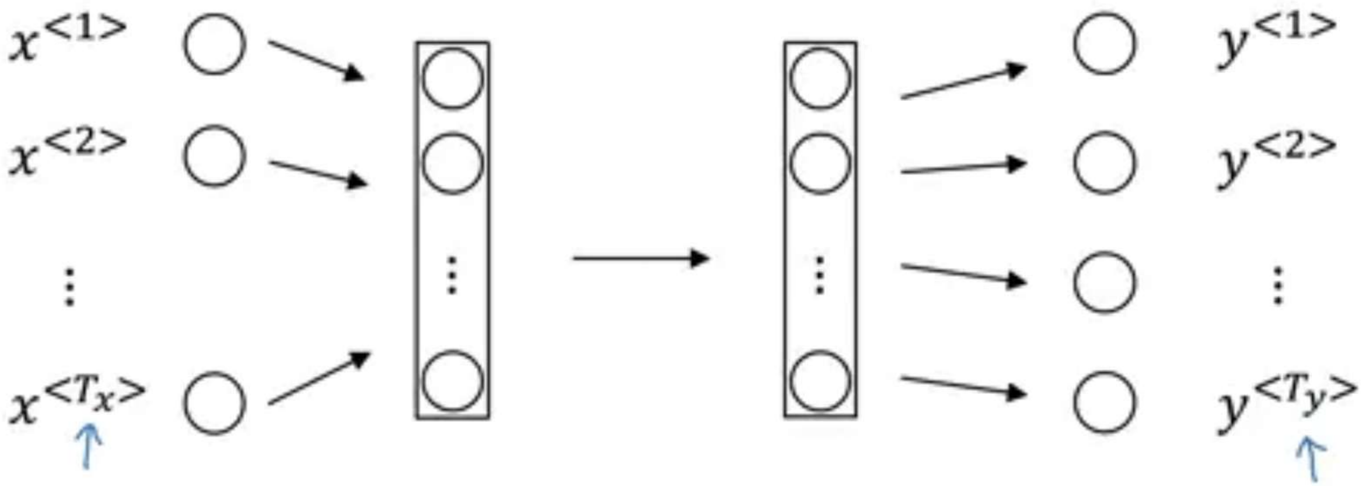

Problems:

- Inputs, outputs can be different lengths in different examples.

# Why not a standard network?

The dog is nice.

She really loves her dog ,and her cat too.



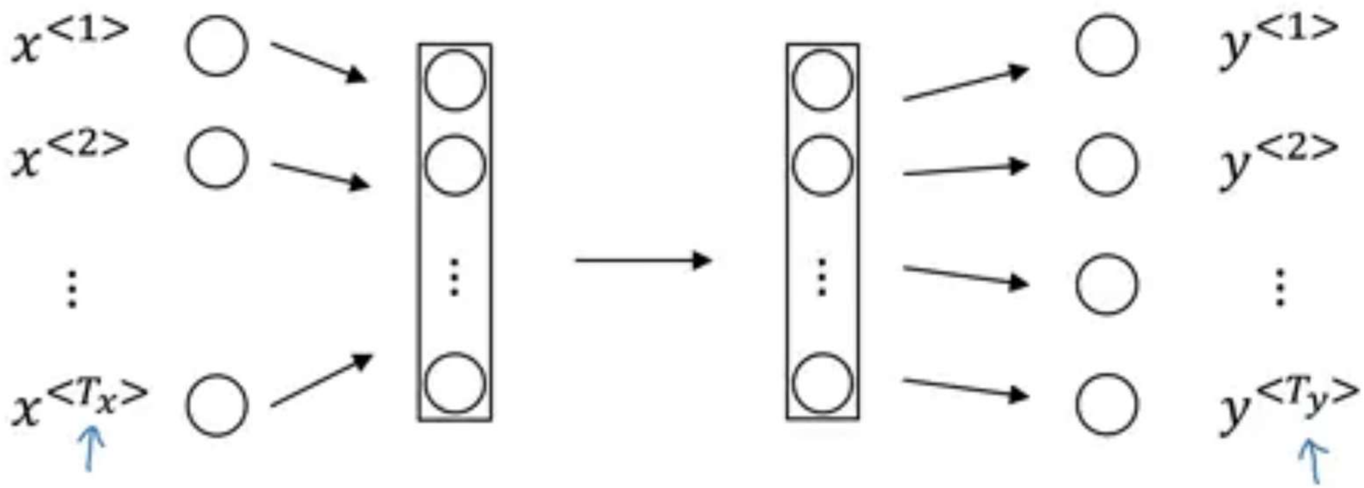

Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.

Example (same word, different position): The dog is nice.

She really loves her dog, and her cat too.

# Why not a standard network?

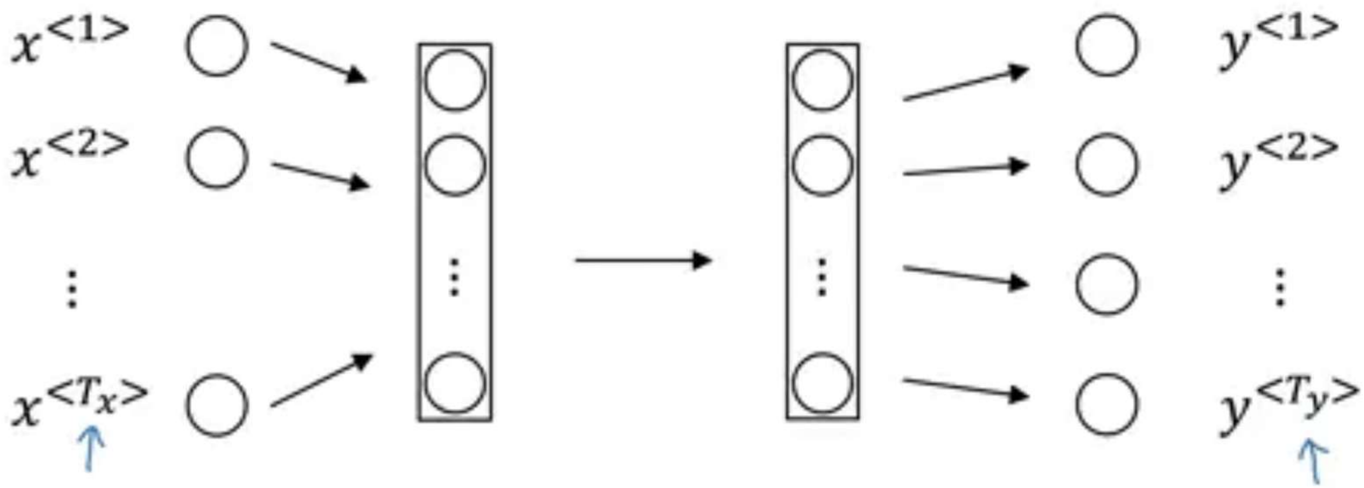



Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.

- Context must be taken into account: EXAMPLE 1:
- " Any law written by the UK governement must be approved by the Parliament"
- vs. "I don't have any <u>cash</u> on me"

# Why not a standard network?

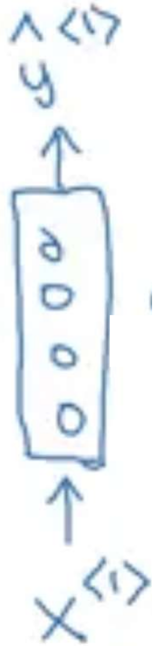



Problems:

- Inputs, outputs can be different lengths in different examples.
- Doesn't share features learned across different positions of text.

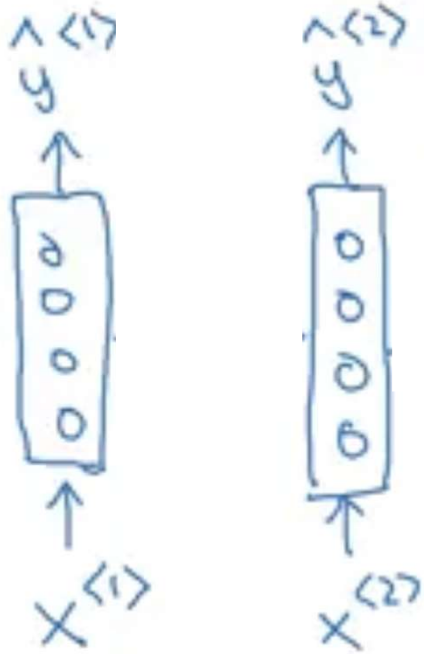
- Context must be taken into account: EXAMPLE 2:
- "That student says"
- vs. "Students say"

# Recurrent Neural Networks




Input at
time 1

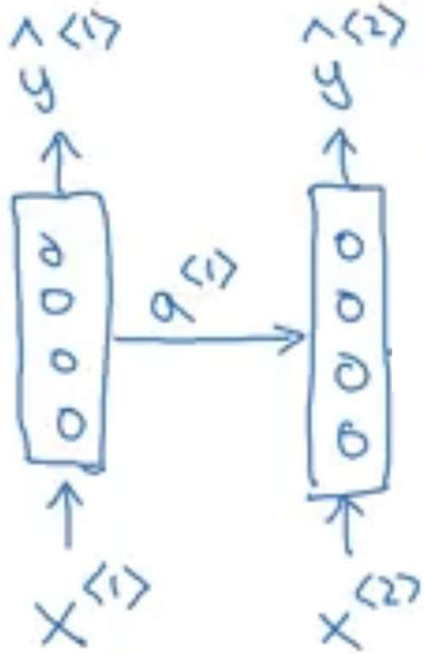
# Recurrent Neural Networks




Input at time 1

Input at time 2

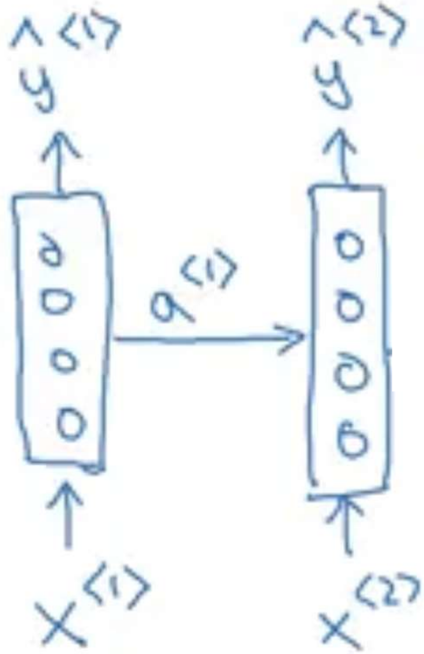
# Recurrent Neural Networks




Input at time 1

Input at time 2

# Recurrent Neural Networks




Input at time 1

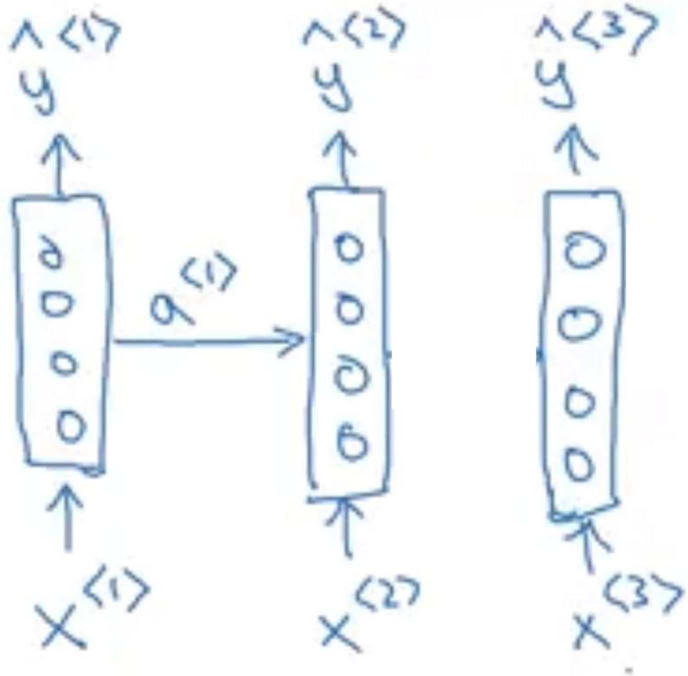
Input at time 2

Examples of importance of keeping into account the history:

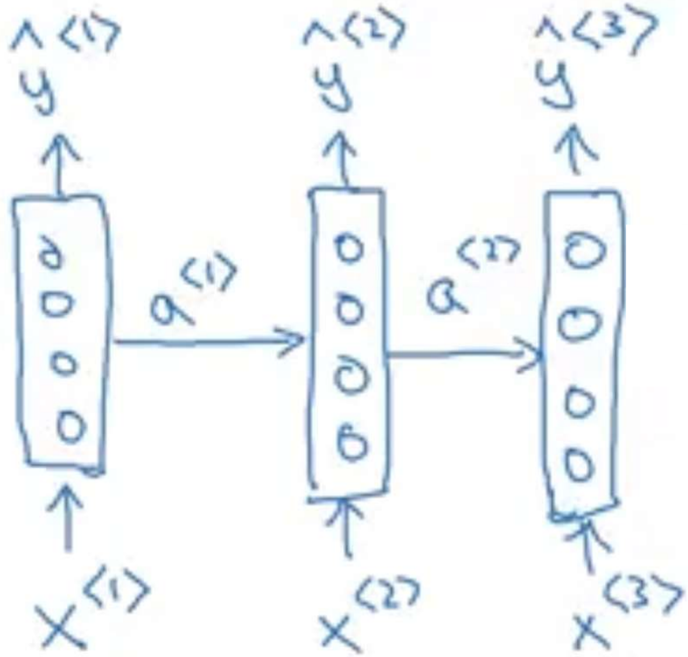
- «**Any** law proposed by the governement must be approved by the Parliament»
  VS
  «I don't have **any** cash on me»

- A student says VS. Students say.

# Recurrent Neural Networks

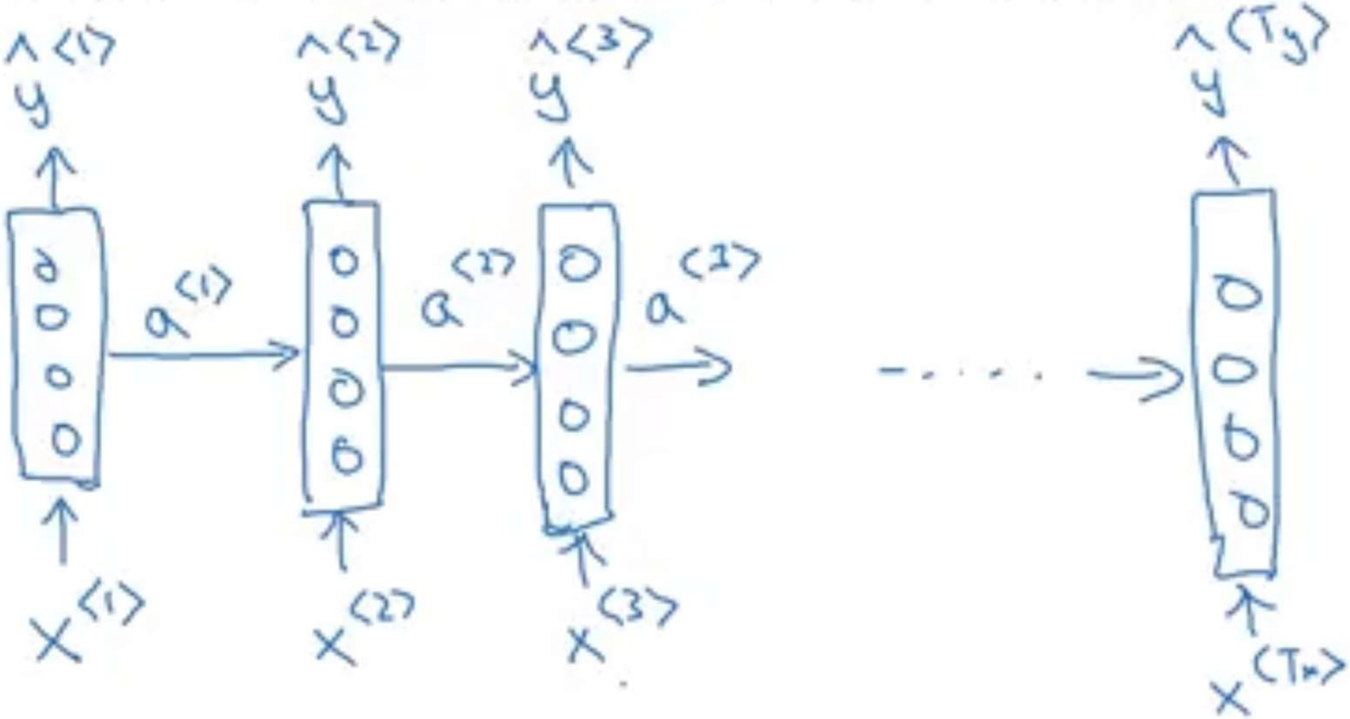


# Recurrent Neural Networks

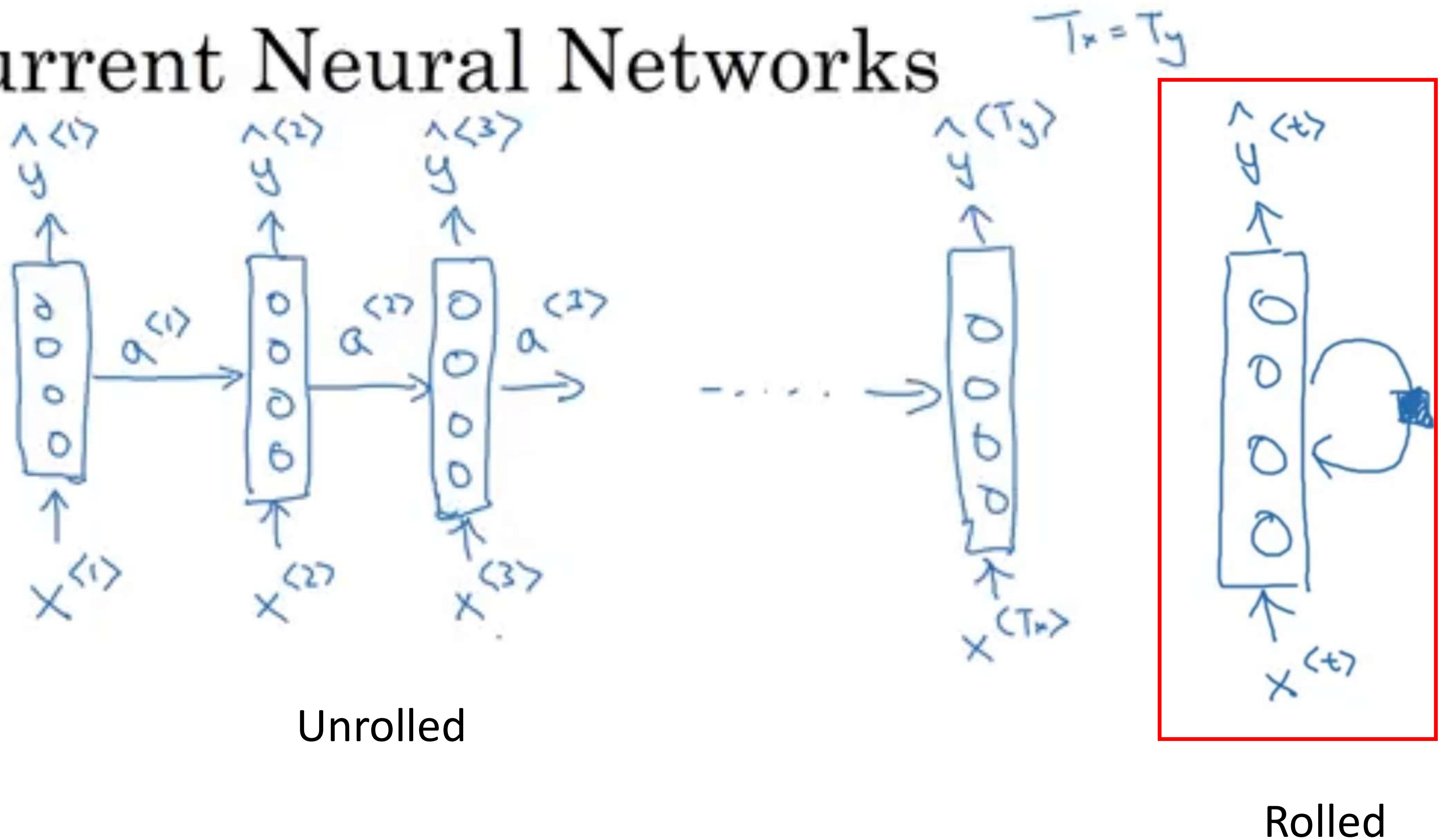


# Recurrent Neural Networks

$T_x = T_y$

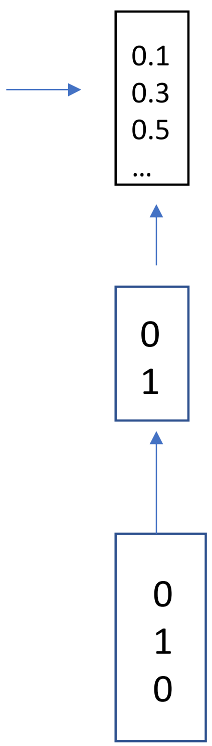


# Recurrent Neural Networks



# Example: Predicting the Next Word

- Probabilities for next output=$i^{th}$ word.




«The»

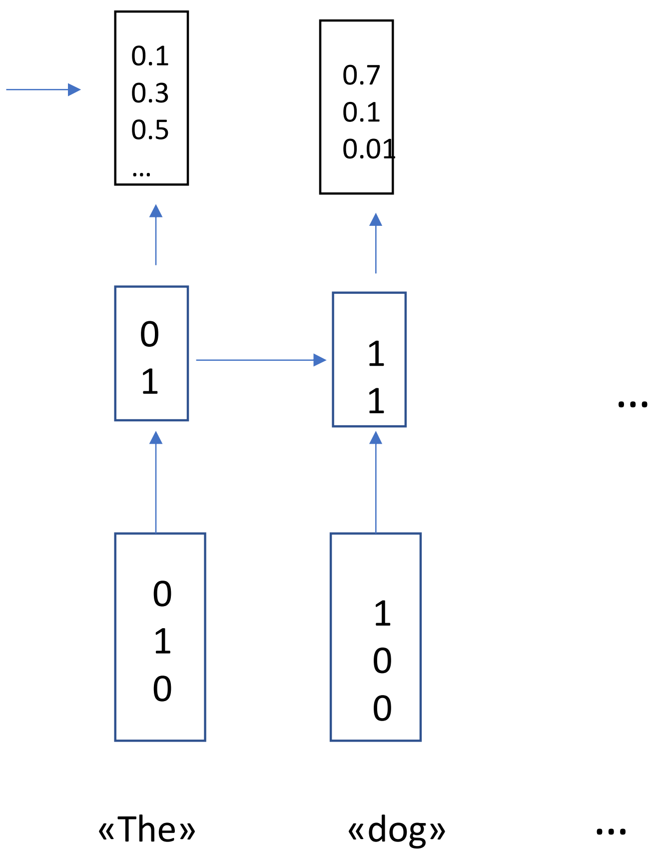
- Vocabulary= [a: 1
  and: 2
  .....
  Harry: 567
  .....
  <unkn>:..]

one-hot representations: [1,0,0,0...]
[0,1,0,0,...]
....

In language models, richer word representations (embeddings)

# Example: Predicting the Next Word

- Probabilities for next output=$i^{th}$ word.

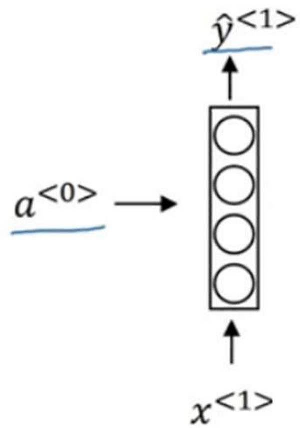



- Vocabulary= [a: 1
  and: 2
  .....
  Harry: 567
  .....
  <unkn>:..]

→ one-hot representations: [1,0,0,0...]
[0,1,0,0,...]
....

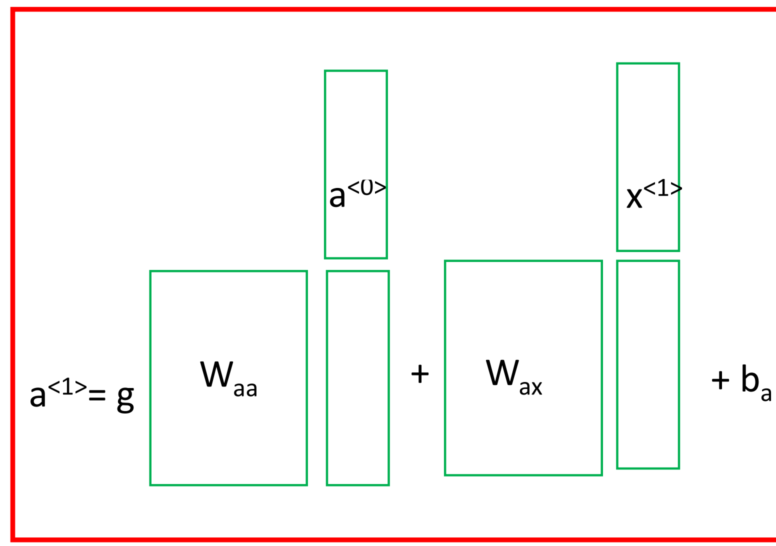
In language models, richer word representations (embeddings), e.g. Word2Vec

# RNN. Actvation Calculation

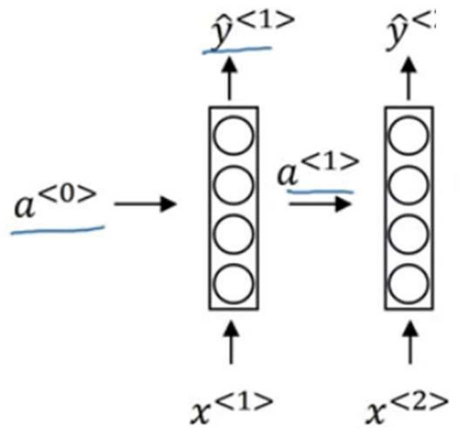



$a^{<1>} = g(W_{ax}\, x^{<1>} + b_a)$
$\hat{y}^{<1>} = g'(W_{ya}\, a^{<1>} + b_y)$

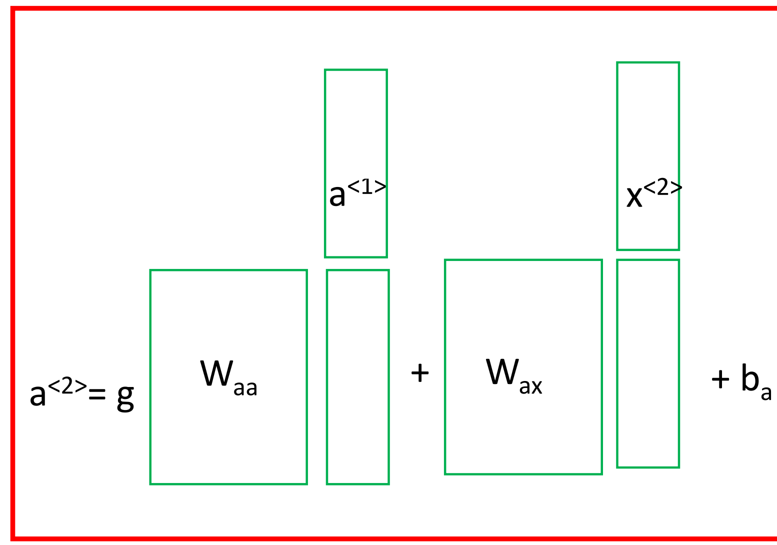
$a^{<0>} =$ zero vector



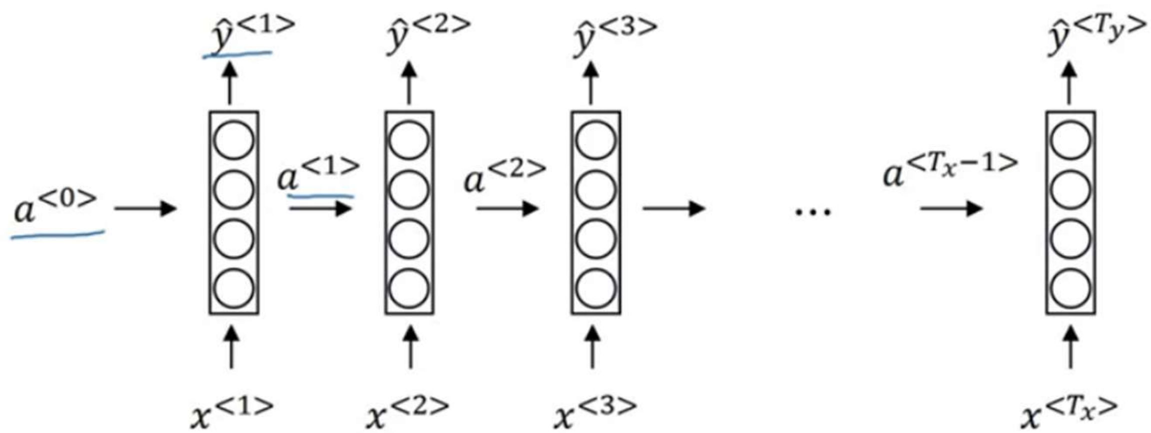
# RNN. Actvation Calculation




$a^{<2>}= g(W_{ax} *x^{<2>} + b_a + W_{aa} * a^{<1>})$
$\hat{y}^{<2>}= g'(W_{ya} *a^{<2>} + b_y)$



# RNN. Actvation Calculation



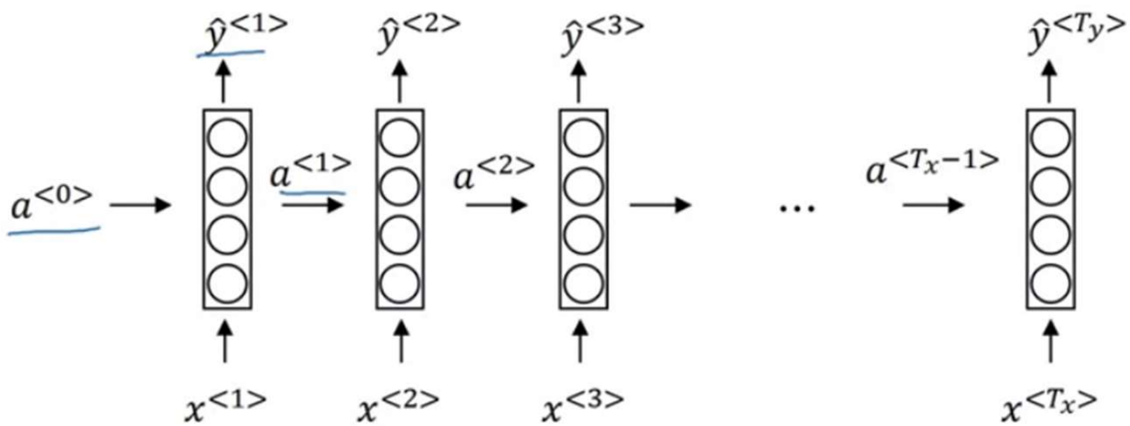

In general:

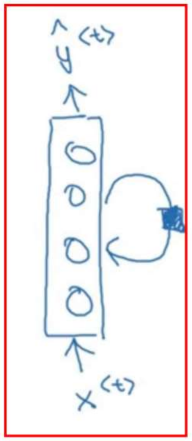
$a^{<i>} = g(W_{ax} * x^{<i>} + b_a + W_{aa} * a^{<i-1>})$

$\hat{y}^{<i>} = g'(W_{ya} * a^{<i>} + b_y)$

$a^{<0>}$= zero vector

# RNN. Actvation Calculation




$a^{<0>} =$

**Rolled**
Shared Weights

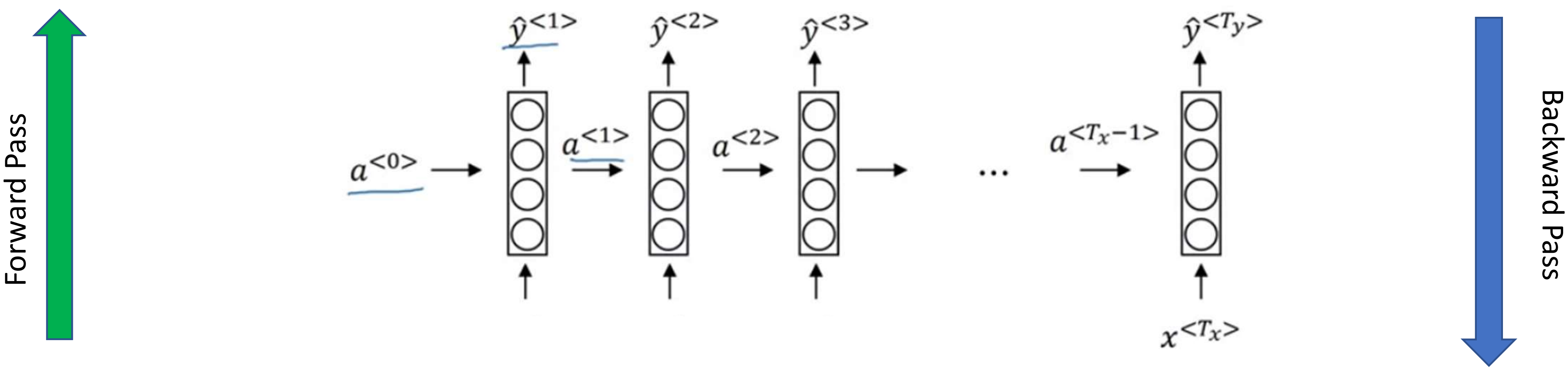
In general:

$a^{<i>} = g(W_{ax} * x^{<i>} + b_a + W_{aa} * a^{<i-1>})$

$\hat{y}^{<i>} = g'(W_{ya} * a^{<i>} + b_y)$

$a^{<0>}$= zero vector

# Backpropagation Through Time (BPTT)

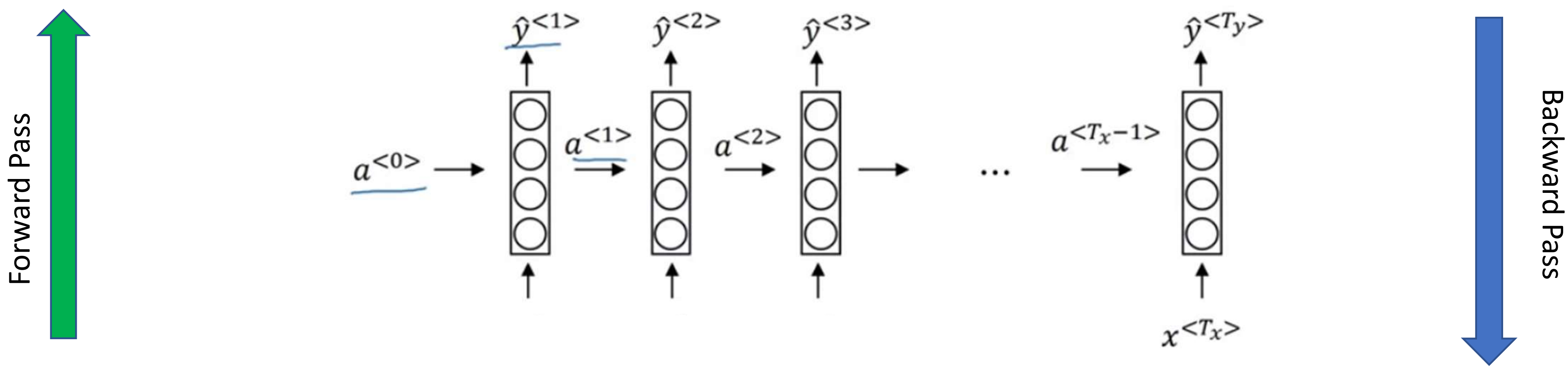



Loss function: $\mathcal{L}^{<t>}(\hat{y}^{<t>}, y^{<t>}) = -y^{<t>} \log \hat{y}^{<t>} - (1-y^{<t>}) \log(1-\hat{y}^{<t>})$

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

Ex: Cross-entropy

# Backpropagation Through Time (BPTT)

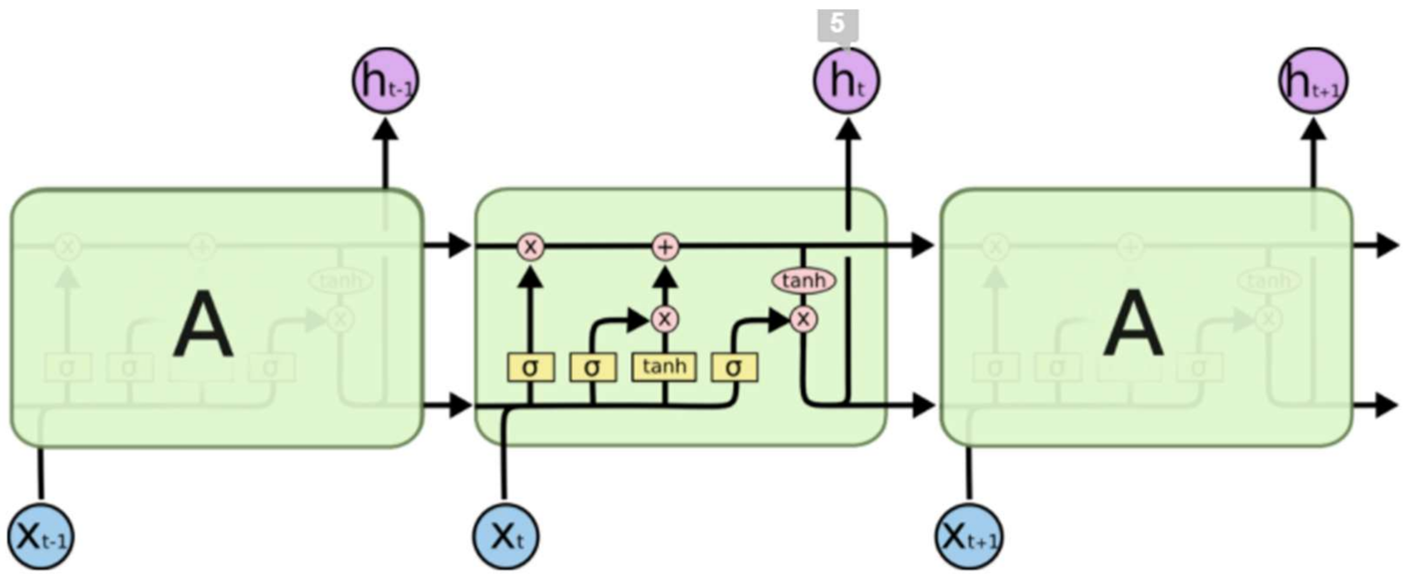



These RNN have a technical problem: vanishing or exploding gradient.
And not very good at capturing long distance dependencies.

# LSTM: Long Short Term Memory

- To capture long distance dependencies, we should have a kind of memory
- «The cat, that had alreay eaten a lot, was full»
- Or «I grew up in France… I speak fluent …. »
- Units have gates that decide when to update memory cells states, what informations to keep, what to forget.
- For instance, you can ignore ‘The’ but consider ‘cat’ as possible subject candidates
- Or something like: to predict verb form, refresh when «,» or «.» .
- Or refresh when new subject

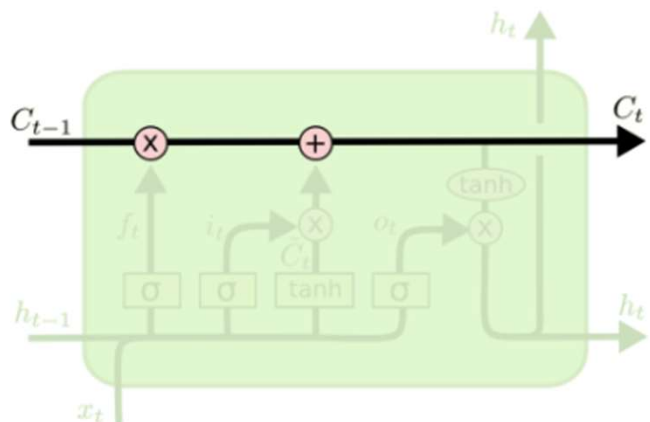
# Long Short Term Memory




LSTM cell

# LSTM: Long Short Term Memory

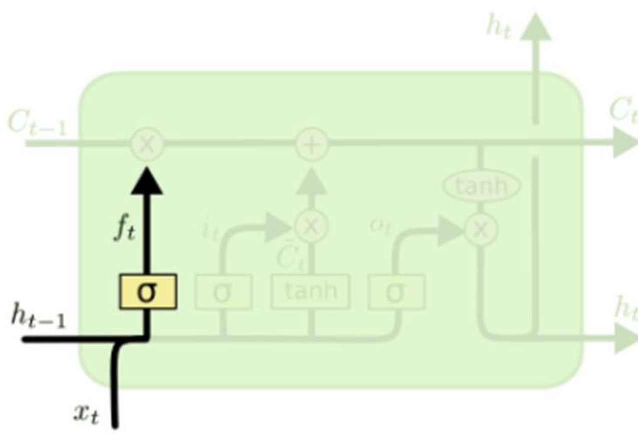
Cell state, the horizontal line running through the top of the diagram.
It transports informations down the entire chain, with only some minor linear interactions.
LSTM have the ability to remove or add information to the cell state, carefully regulated by structures called gates.




**LSTM cell state**

# LSTM: Long Short Term Memory

Forget gate: what information we're going to eliminate from the cell state




Forget gate

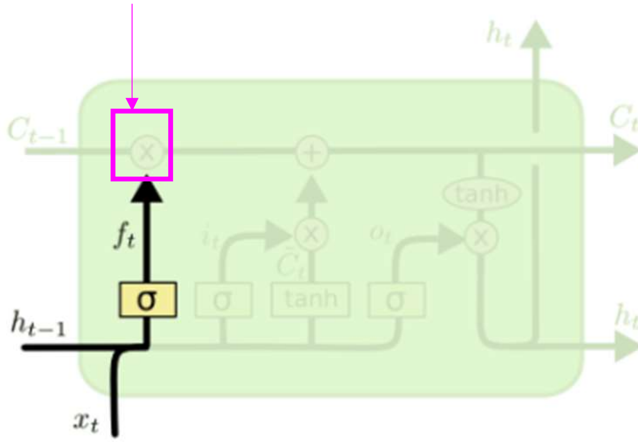
$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Given $h_{t-1}$ and $x_t$, $f_t$ is a vector of values between 1 and 0.
Same dimension as $C_{t-1}$

# LSTM: Long Short Term Memory

Forget gate: what information we’re going to eliminate from the cell state

Values of $f_t$ are then multiplied elementwise with $C_{t-1}$, thus gating what values of $C_{t-1}$ are kept, what thrown away.




Forget gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

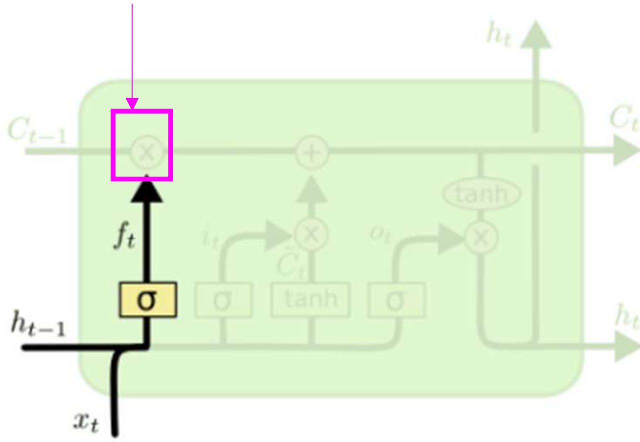
Given $h_{t-1}$ and $x_t$, $f_t$ is a vector of values between 1 and 0.
Same dimension as $C_{t-1}$

# LSTM: Long Short Term Memory

Forget gate: what information we're going to eliminate from the cell state

Example: throw away if subject singular or plural from $C_{t-1}$ when new subject.

Values of $f_t$ are then multiplied elementwise with $C_{t-1}$, thus gating what values of $C_{t-1}$ are kept, what thrown away.




Forget gate

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

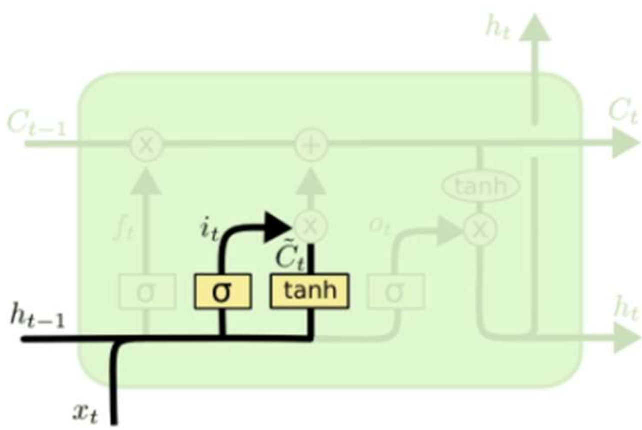
Given $h_{t-1}$ and $x_t$, $f_t$ is a vector of values between 1 and 0.
Same dimension as $C_{t-1}$

# Informal example forget gate $f_T$

- Let (for the example only!) $\sigma$= 1 for arguments > 0, 0 otherwise
- $W_f$=[1,1,1,-10; 1,1,2,-10], $b_f$=0
- $[h_{t-1}, x_t]$ = [1,1,0,1] (suppose $x_t$= [0,1] codifies «.»)
- Then $f_T$= [0,0]  Intuitively (and informally) «.» erases everything from $C_{t-1}$.

# LSTM: Long Short Term Memory

Input gate: what to add. Example: information on new subject




Input gate

Input gate: what new information should be added

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

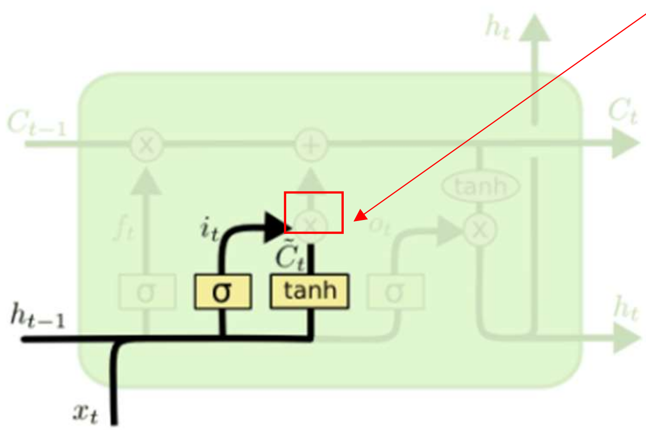
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

New candidate cell state

# LSTM: Long Short Term Memory

Input gate: what to add. Example: information on new subject

1-Input gate is multiplied with candidate cell state




Input gate

$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

New candidate cell state

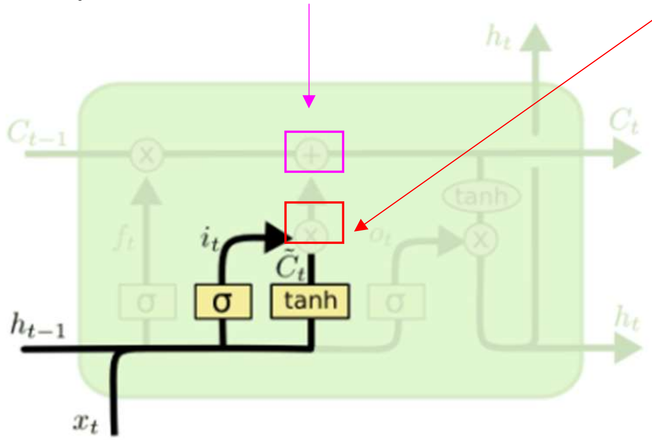
Input gate

# LSTM: Long Short Term Memory

Input gate: what to add. Example: information on new subject

2-Result is then added to the previous cell state

1-Input gate is multiplied with candidate cell state




Input gate

Input gate

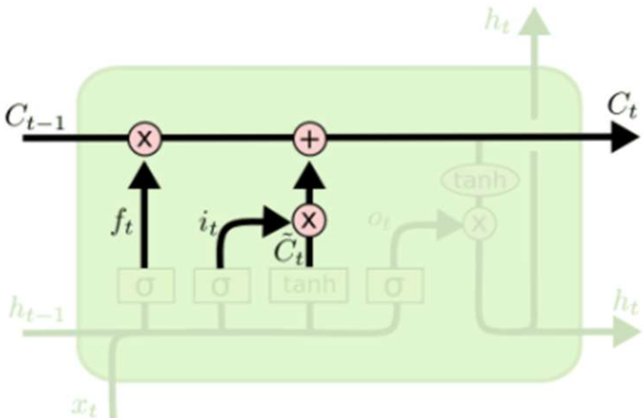
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

New candidate cell state

# LSTM: Long Short Term Memory

As a result of the previous operations by forget and input gate: a new cell state is computed.

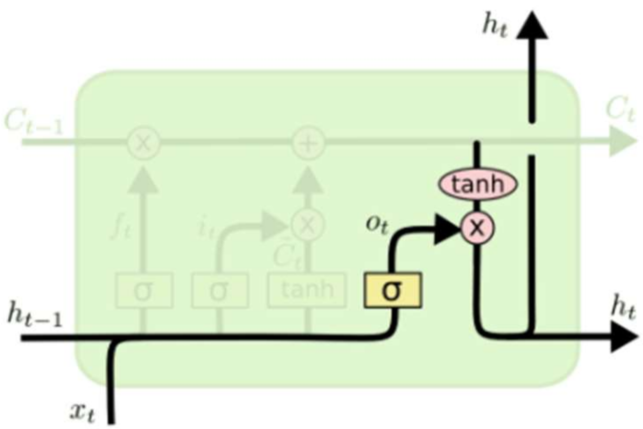



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

New cell state

# LSTM: Long Short Term Memory

And from the new cell state $C_t$, output $h_t$ is computed (with the help of output gate $o_t$)




Output gate

$$o_t = \sigma\left(W_o\left[h_{t-1}, x_t\right] + b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

ot= output gate: which part of the cell state we are going to keep

Apply $o_t$ to cell state $C_t$ to really compute (filter)what to output

# LSTM: Long Short Term Memory

Excellent blog on LSTM: colah.github.io/posts/2015-08-Understanding-LSTMs/

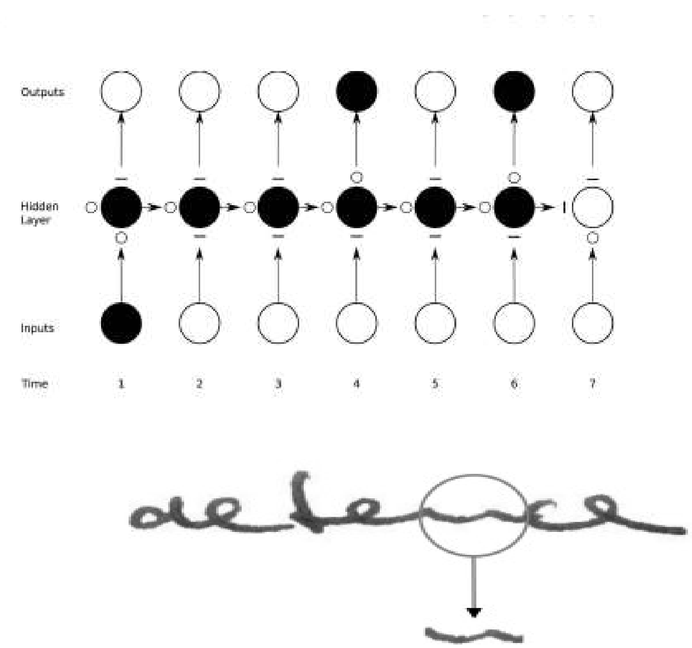
# GRU: Gated Recurrent Unit

Similar gated principles than LSTM

# EXAMPLE APPLICATION (Overall Architecture)

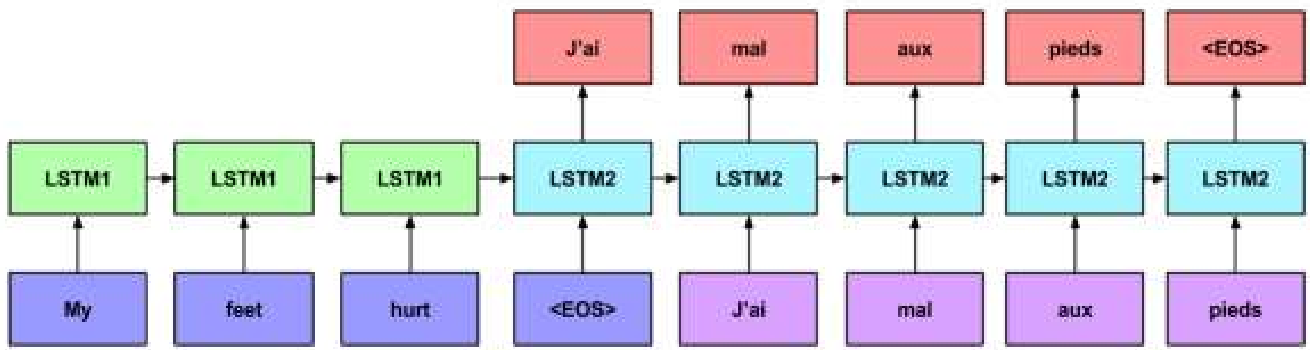
- 

D E F E N C E




Graves et al. (2009). Handwritten recognition.

# EXAMPLE APPLICATION (Overall Architecture)



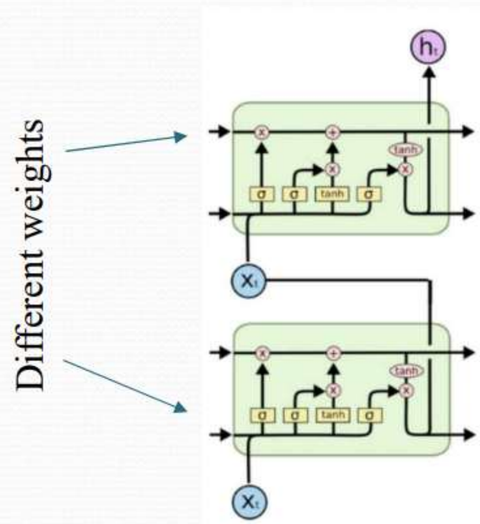

Sequence to sequence LSTM model of Sutskever et al. [2014]. The network consists of an encoding model (first LSTM) and a decoding model (second LSTM). The input blocks (blue and purple) correspond to word vectors, which are fully connected to the corresponding hidden state. Red nodes are softmax outputs. Weights are tied among all encoding steps and among all decoding time steps

# LSTM APPLICATIONS

- Speech recognition
- Machine translation
- Syntactic parsing
- Handwriting recognition
- Image captioning

# Stacked LSTM

## Stacked Lstm: going deep




”

*... building a deep RNN by stacking multiple recurrent hidden states on top of each other. This approach potentially allows the hidden state at each level to operate at different timescale*

”

Note that the output of the LSTM below is used as input for the upper one

How to Construct Deep Recurrent Neural Networks, 2013, Pascanu & al.