Introduction to Recurrent Neural Networks (RNNs)

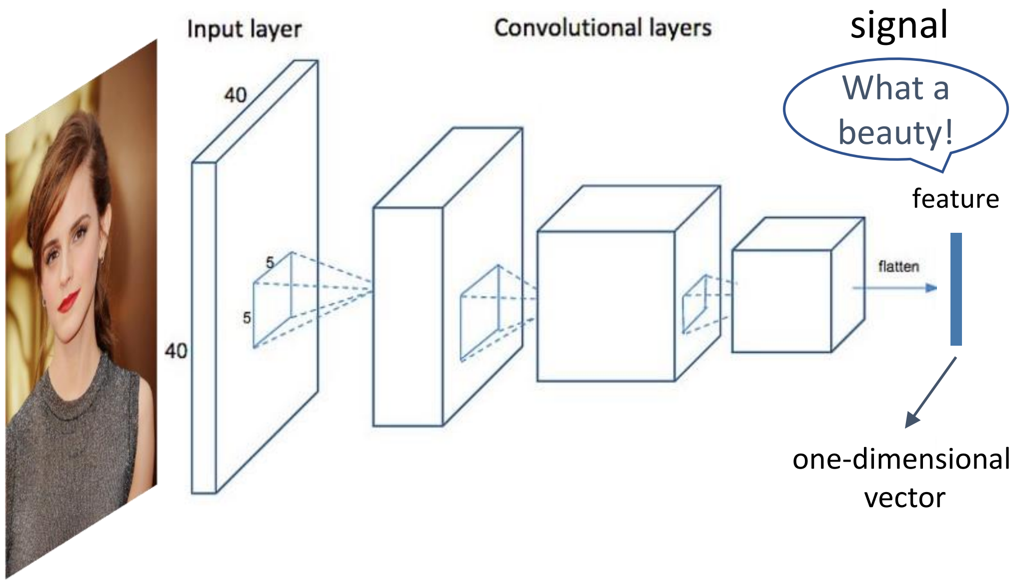
Outline

- What are RNNs?
- RNN units/cells
  - Vanilla RNN unit
  - Long Short-Term Memory (LSTM) unit
- RNN structures
  - Stacked LSTM
  - Bidirectional RNN
  - Hierarchical RNN
- Applications
  - Machine translation
  - Image captioning
  - Visual Question Answering
  - Visual Dialog

Fundamental neural networks

CNN

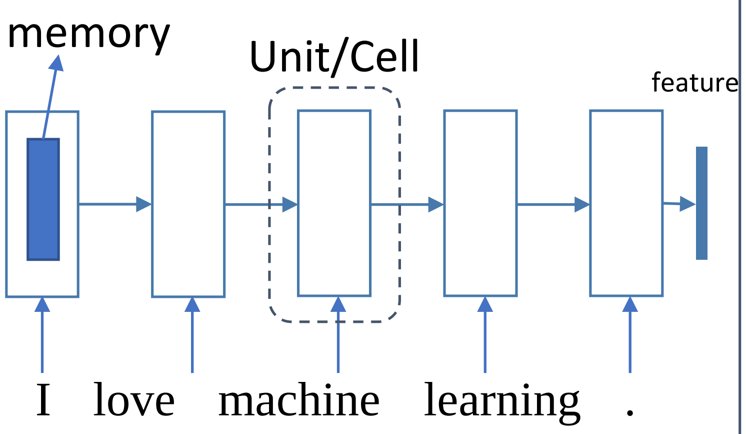
Eyes | Optic nerves | Brain



Layers are different.

RNN

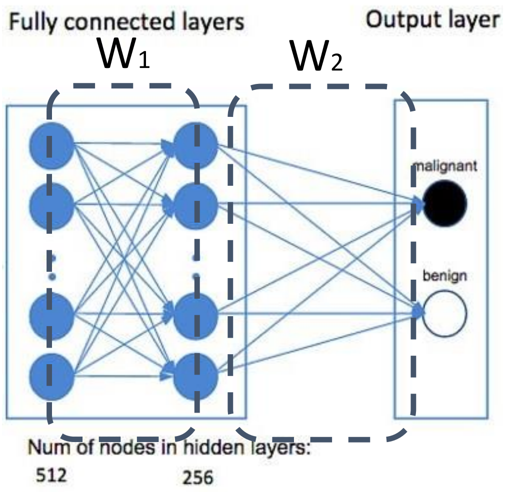
- Speech Recognition
- Machine Translation
- Time-series Prediction



Units/Cells are repeated.

FC

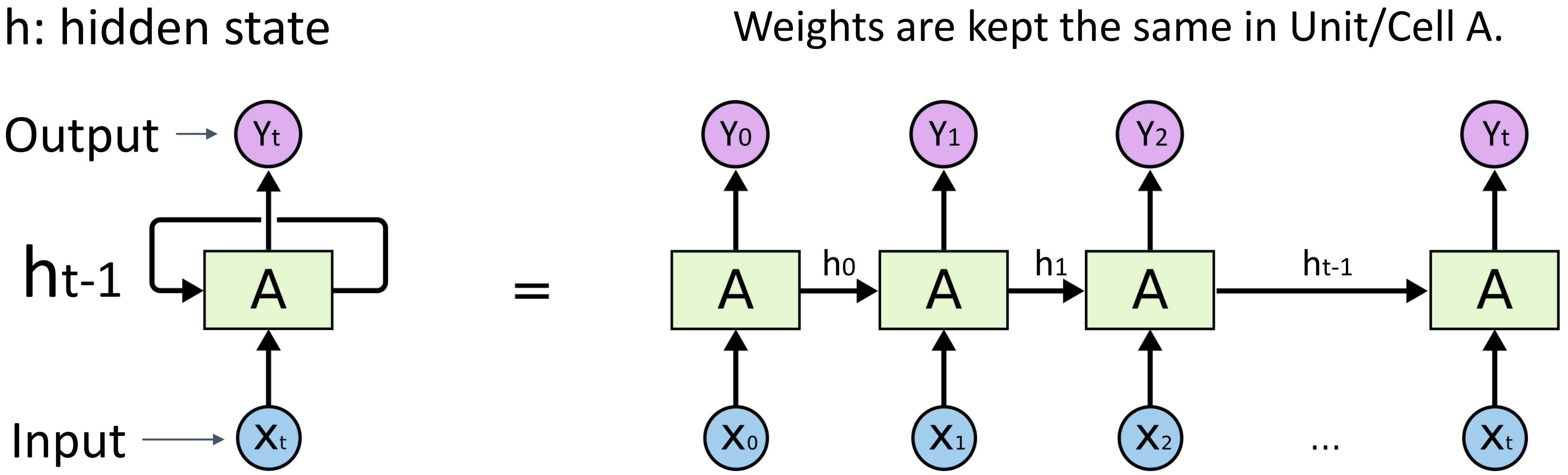
- Feature transformation



Matrix Multiplication

Layers are different.

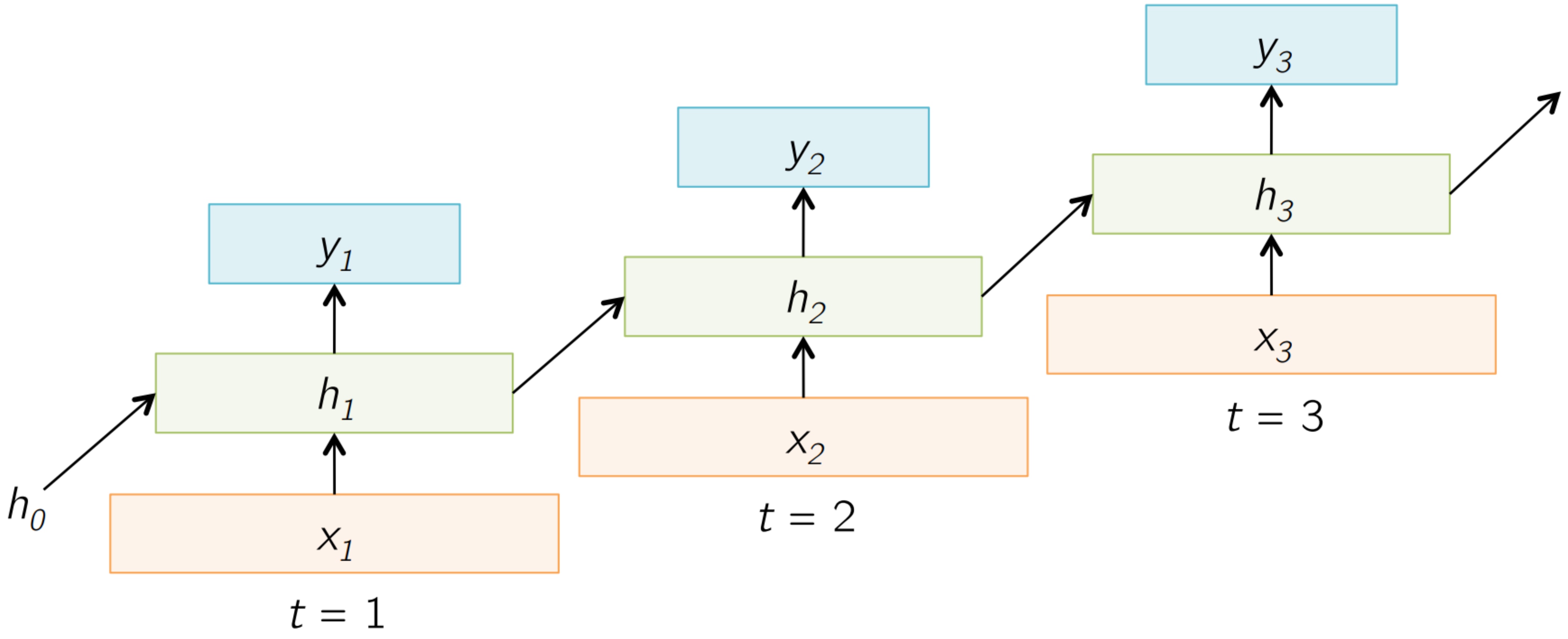
What are RNNs?



- The hidden state at time *t* has some historical information about the happenings before time *t*.
- RNNs are useful as their intermediate state can store information about past inputs.

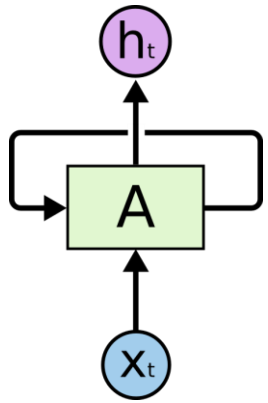
http://colah.github.io/posts/2015-08-Understanding-LSTMs/

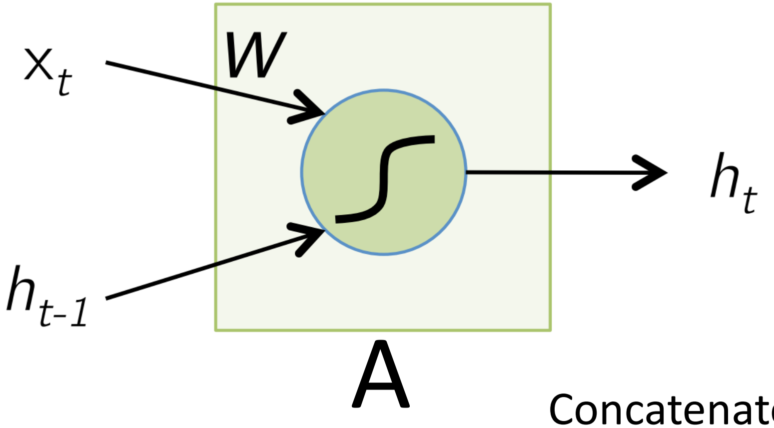
What are RNNs?



The Vanilla RNN Unit/Cell

W is the weights that the RNN needs to learn.
For Vanilla RNN cell, the hidden state *h* is its output.



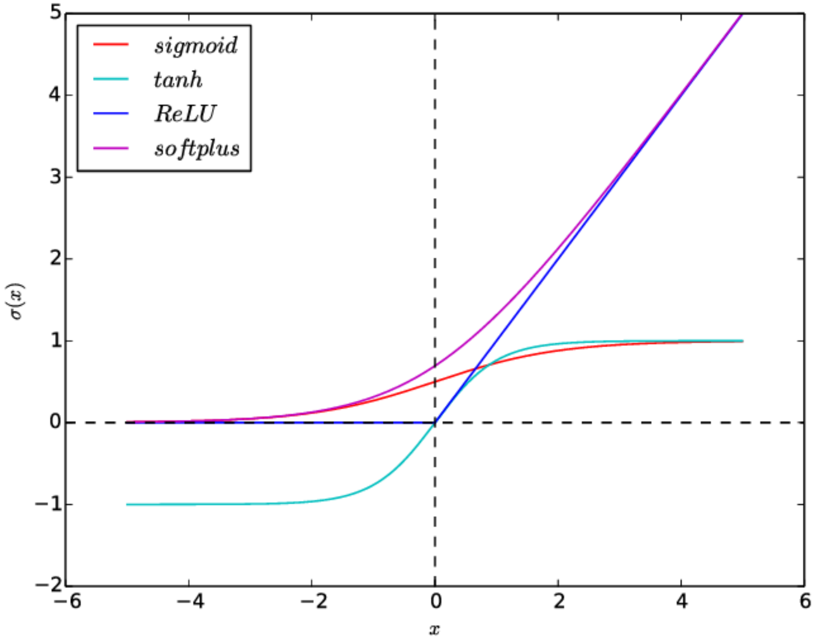


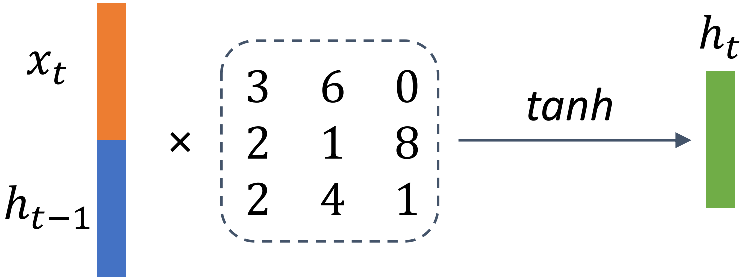
Concatenate

$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

Matrix Multiplication

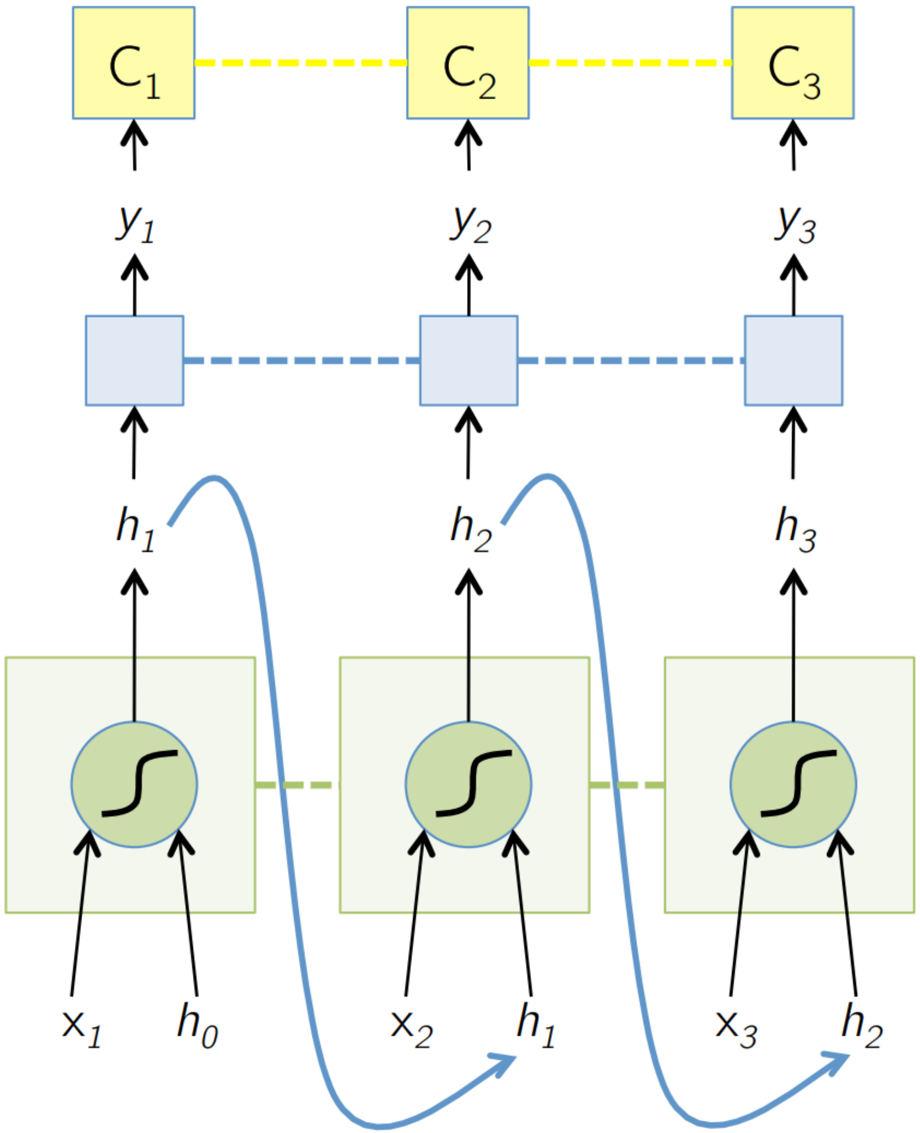
Activation functions (non-linear)





W: a matrix

The Vanilla RNN Forward



- Note that the weights are shared over time.
- Essentially, copies of the RNN cell are made over time, with different inputs at different time steps

$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

------ indicates shared weights

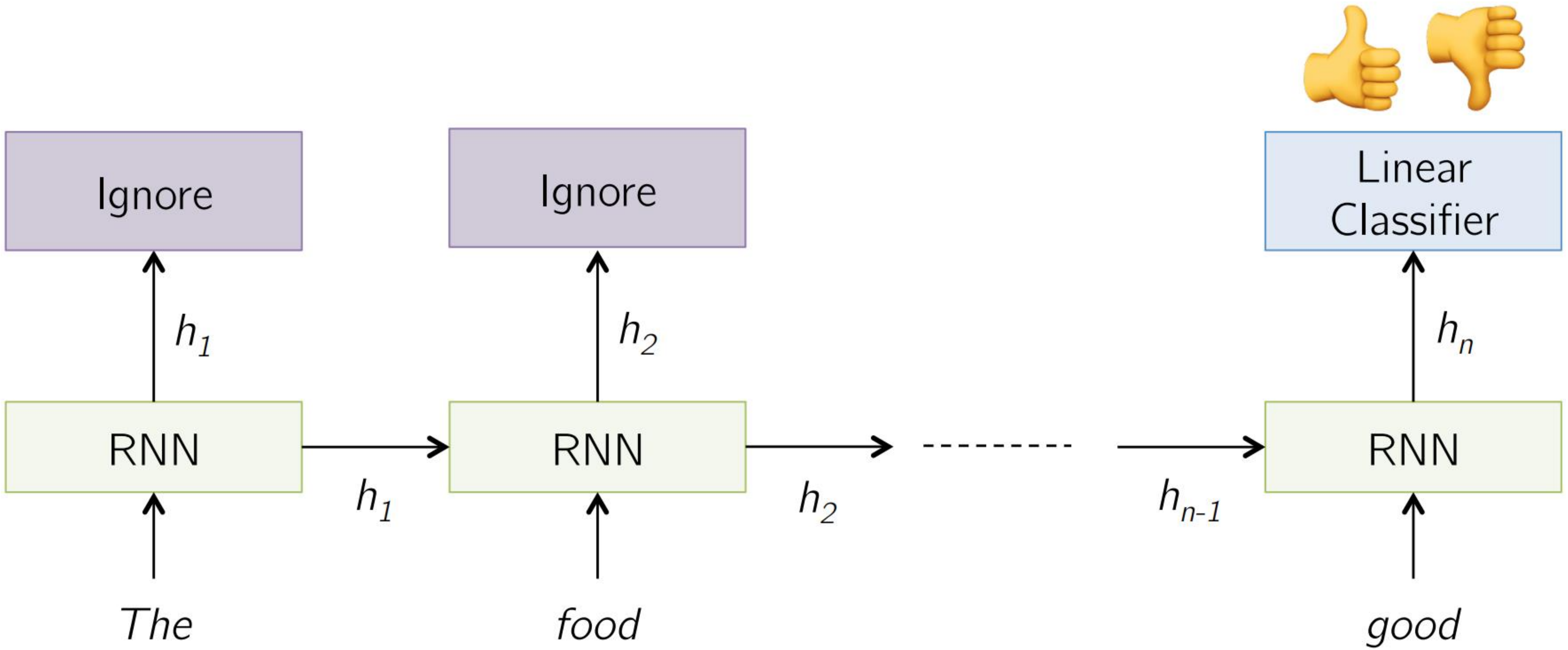
Sentiment Classification

- Classify a
  restaurant review from Yelp! OR
  movie review from IMDB OR
  …
  as positive or negative

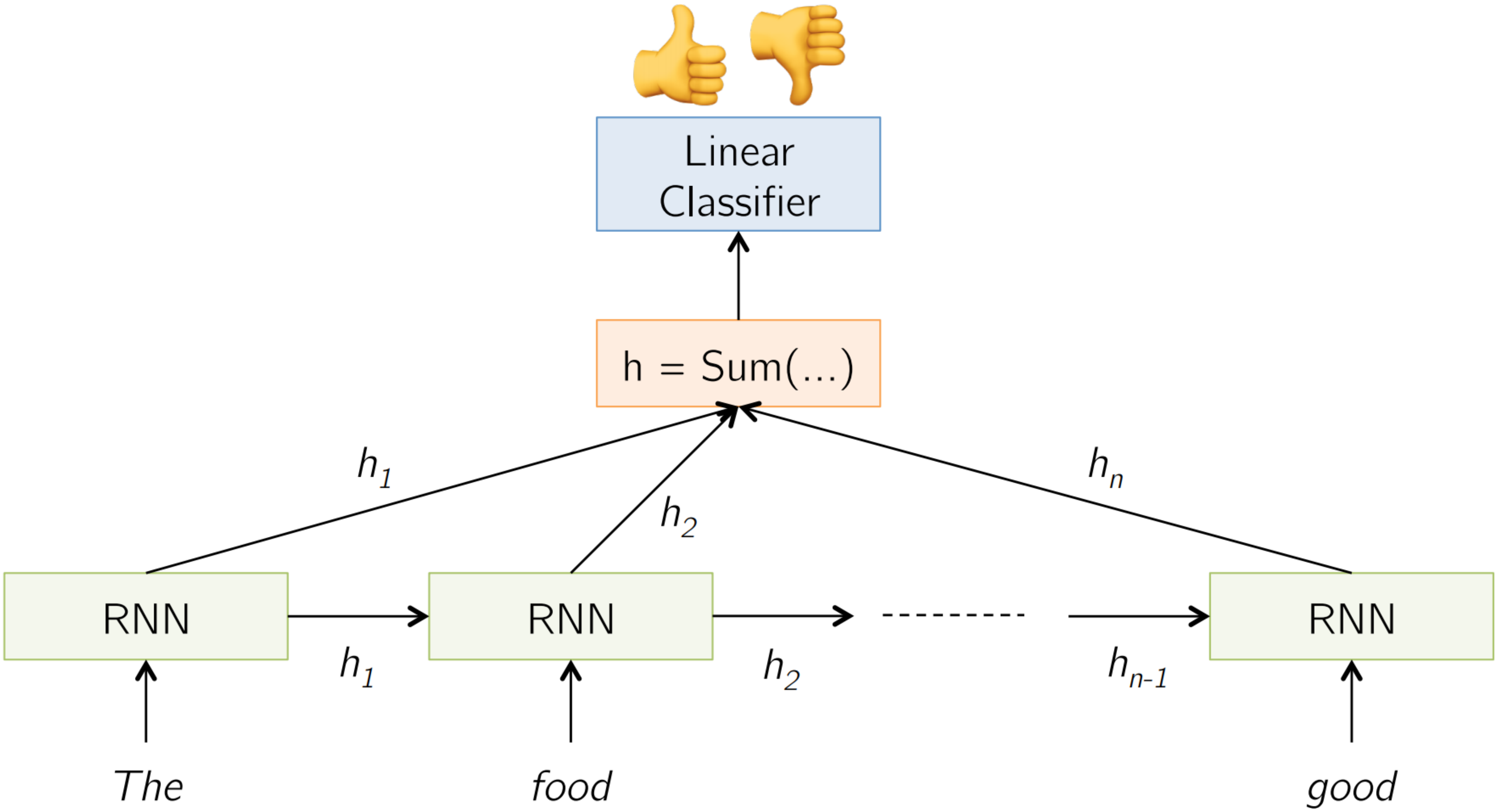
- **Inputs:** Multiple words, one or more sentences
- **Outputs:** Positive / Negative classification

- "The food was really good"
- "The chicken crossed the road because it was uncooked"

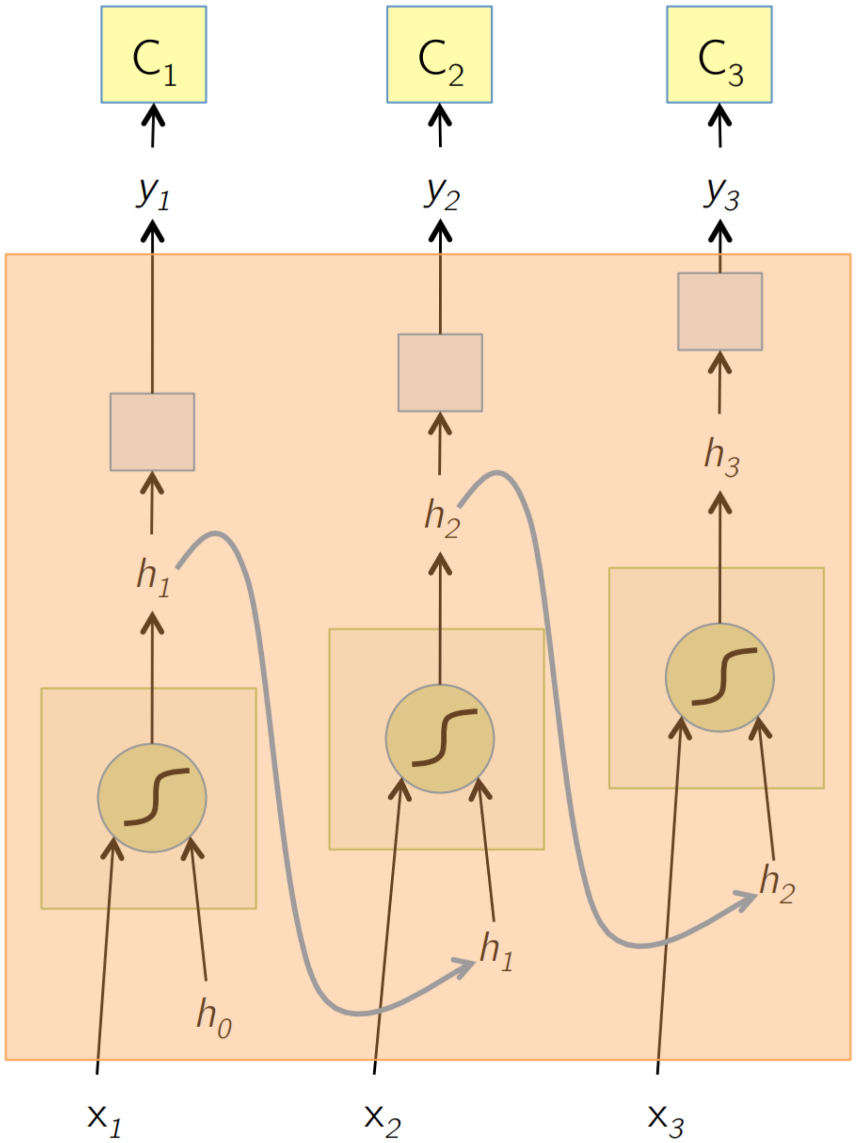
Sentiment Classification



Sentiment Classification



Backpropagation



We compute gradients through back propagation, similar to normal deep learning

Difference: weights are shared!

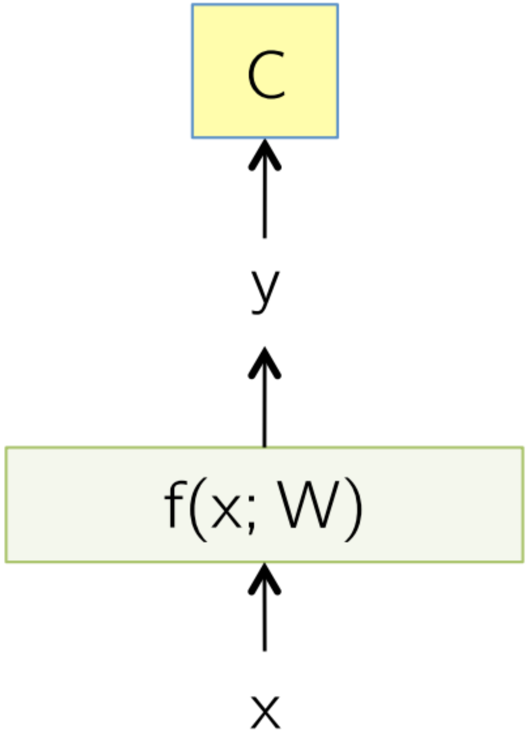
How can we update the weights when they are shared?

BackPropagation Through Time (BPTT)

- One of the methods used to train RNNs
- RNN network accepts the whole time series as input
- The weight updates are computed for each cell in the network, then summed (or averaged) and then applied to the weights

- What is the difference with normal deep learning BP?

BackPropagation Revision



$$y = f(x; W)$$

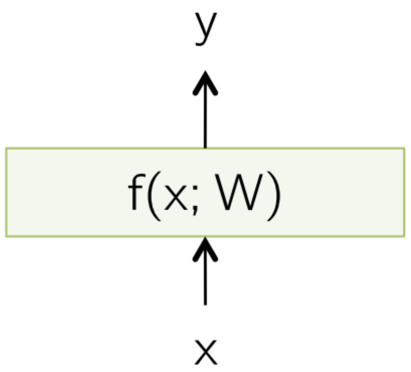
$$C = \text{Loss}(y, y_{GT})$$

SGD Update

$$W \leftarrow W - \eta \frac{\partial C}{\partial W}$$

$$\frac{\partial C}{\partial W} = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right)$$

Chain Rule for Gradient Computation



Given: $\left(\frac{\partial C}{\partial y}\right)$

We are interested in computing: $\left(\frac{\partial C}{\partial W}\right), \left(\frac{\partial C}{\partial x}\right)$

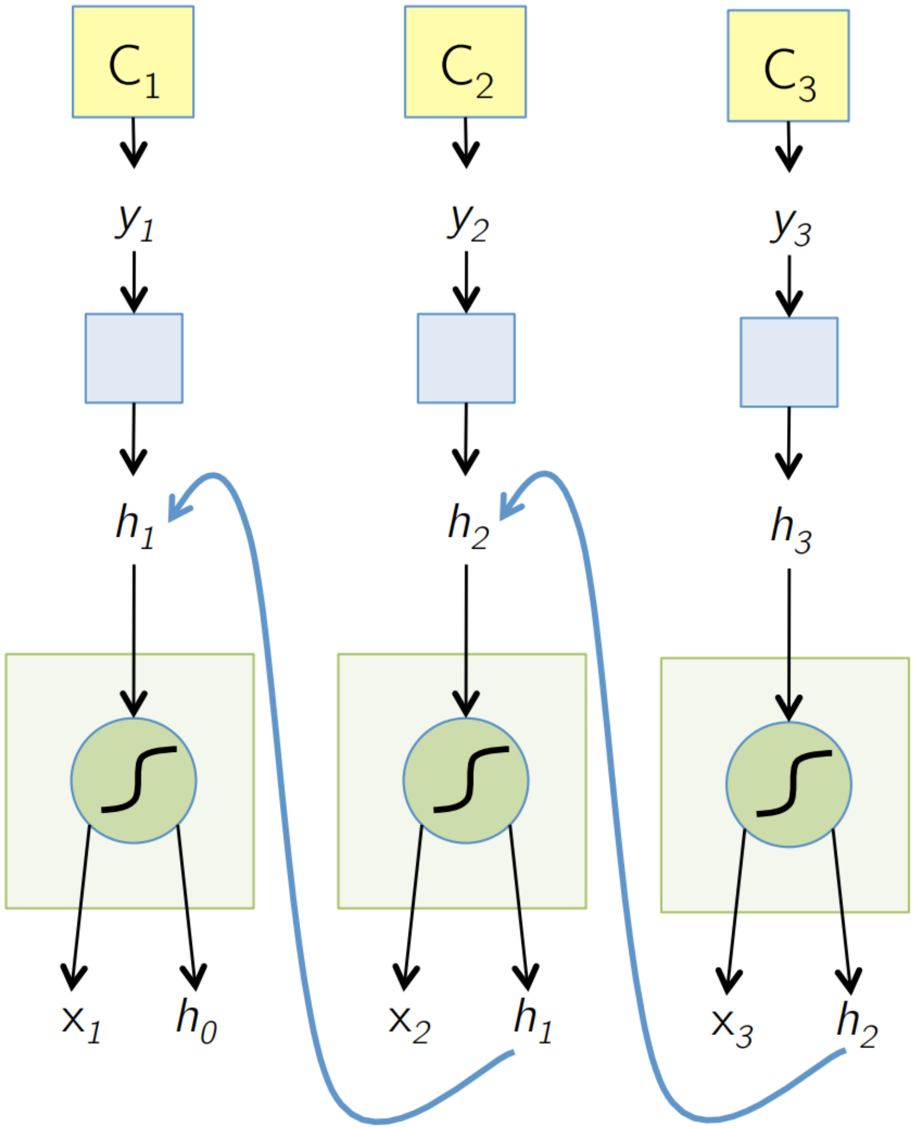
Intrinsic to the layer are:

$\left(\frac{\partial y}{\partial W}\right)$ – How does output change due to params

$\left(\frac{\partial y}{\partial x}\right)$ – How does output change due to inputs

$$\left(\frac{\partial C}{\partial W}\right) = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial W}\right) \qquad \left(\frac{\partial C}{\partial x}\right) = \left(\frac{\partial C}{\partial y}\right)\left(\frac{\partial y}{\partial x}\right)$$

BackPropagation of The Vanilla RNN



$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$ F: fully connected layer

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

$$\frac{\partial C_t}{\partial h_1} = \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_1}\right)$$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\boxed{\left(\frac{\partial h_t}{\partial h_{t-1}}\right)}\cdots\boxed{\left(\frac{\partial h_2}{\partial h_1}\right)}$$

- > 1 ?
- < 1 ?

The Identity Relationship

$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

$$y_t = \mathrm{F}(h_t)$$

$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

- Recall $\dfrac{\partial C_t}{\partial h_1} = \left(\dfrac{\partial C_t}{\partial y_t}\right)\left(\dfrac{\partial y_t}{\partial h_1}\right)$

$$= \left(\frac{\partial C_t}{\partial y_t}\right)\left(\frac{\partial y_t}{\partial h_t}\right)\left(\frac{\partial h_t}{\partial h_{t-1}}\right)\cdots\left(\frac{\partial h_2}{\partial h_1}\right)$$

- < 1 vanishing gradients
- > 1 exploding gradients

Consider a long sentence

- Suppose that instead of a matrix multiplication, we had an **identity relationship** between the hidden states

$$h_t = h_{t-1} + F(x_t)$$

$$\Rightarrow \left(\frac{\partial h_t}{\partial h_{t-1}}\right) = 1$$

- The gradient does not decay as the error is propagated all the way back aka "Constant Error Flow"

Long Short-Term Memory (LSTM)

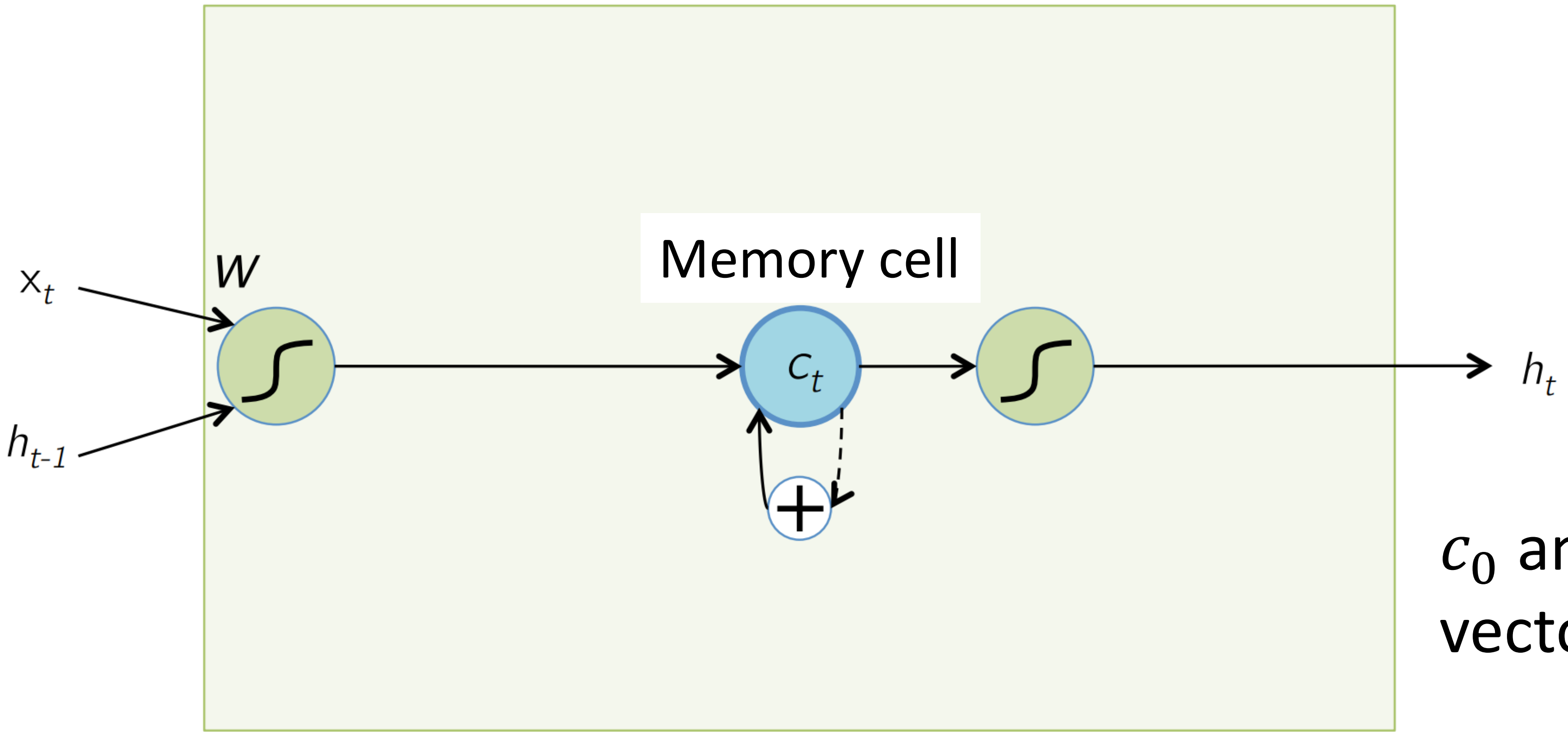
- The LSTM uses this idea of “Constant Error Flow” for RNNs to create a “Constant Error Carousel” (CEC) which ensures that gradients don’t decay

- The key component is a memory cell (***C***) that acts like an accumulator (contains the identity relationship) over time

- Instead of computing new state as a matrix product with the old state, LSTM computes the difference between them. Gradients are better behaved

$$\boxed{h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}} \quad \times$$

The idea of LSTM



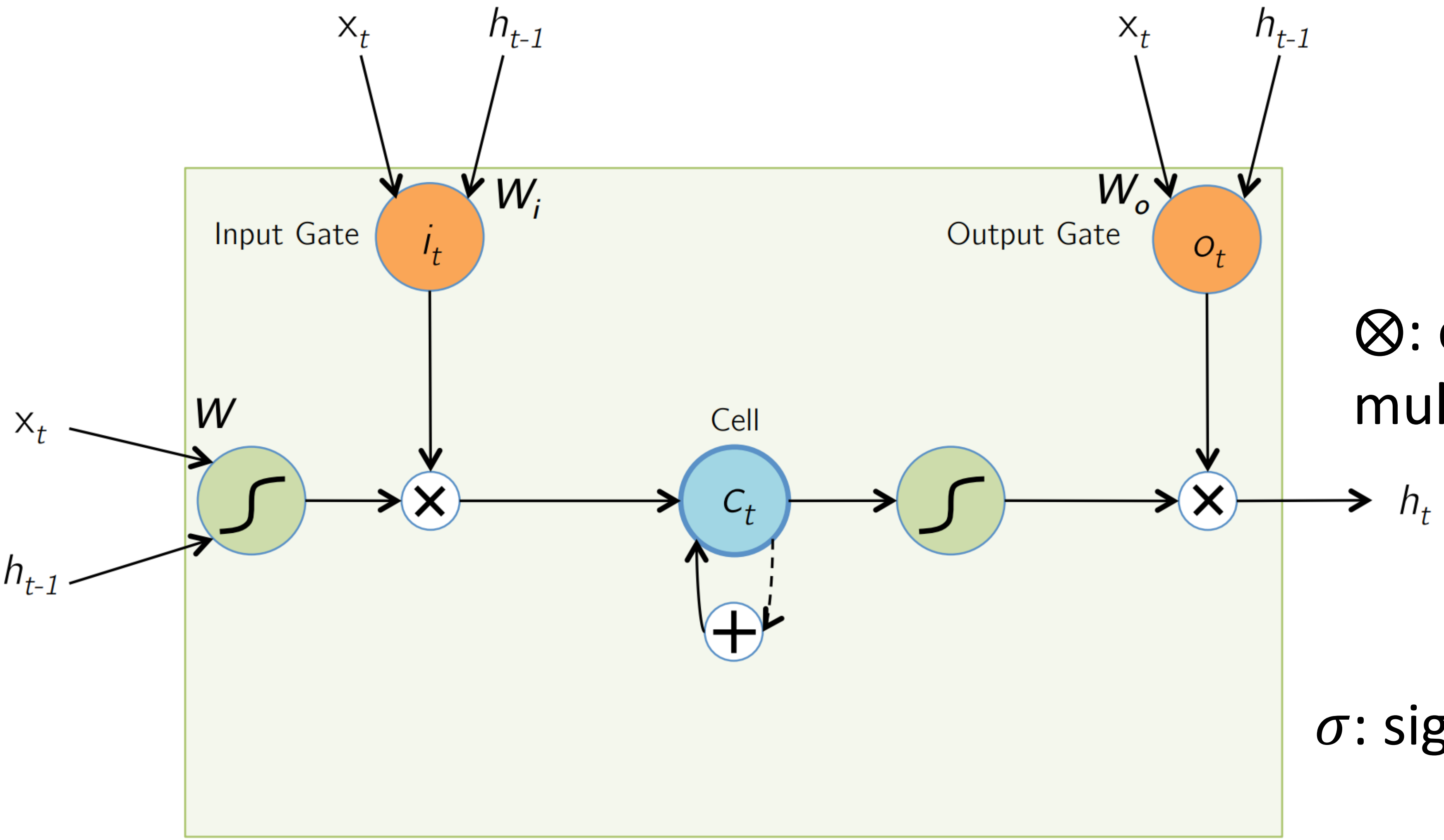
c_t is called the **cell state**

We call c_t and h_t the **state**

c_0 and h_0 are initialized as a vector of zeros

$$c_t = c_{t-1} + \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \qquad h_t = \tanh c_t$$

The Original LSTM Cell



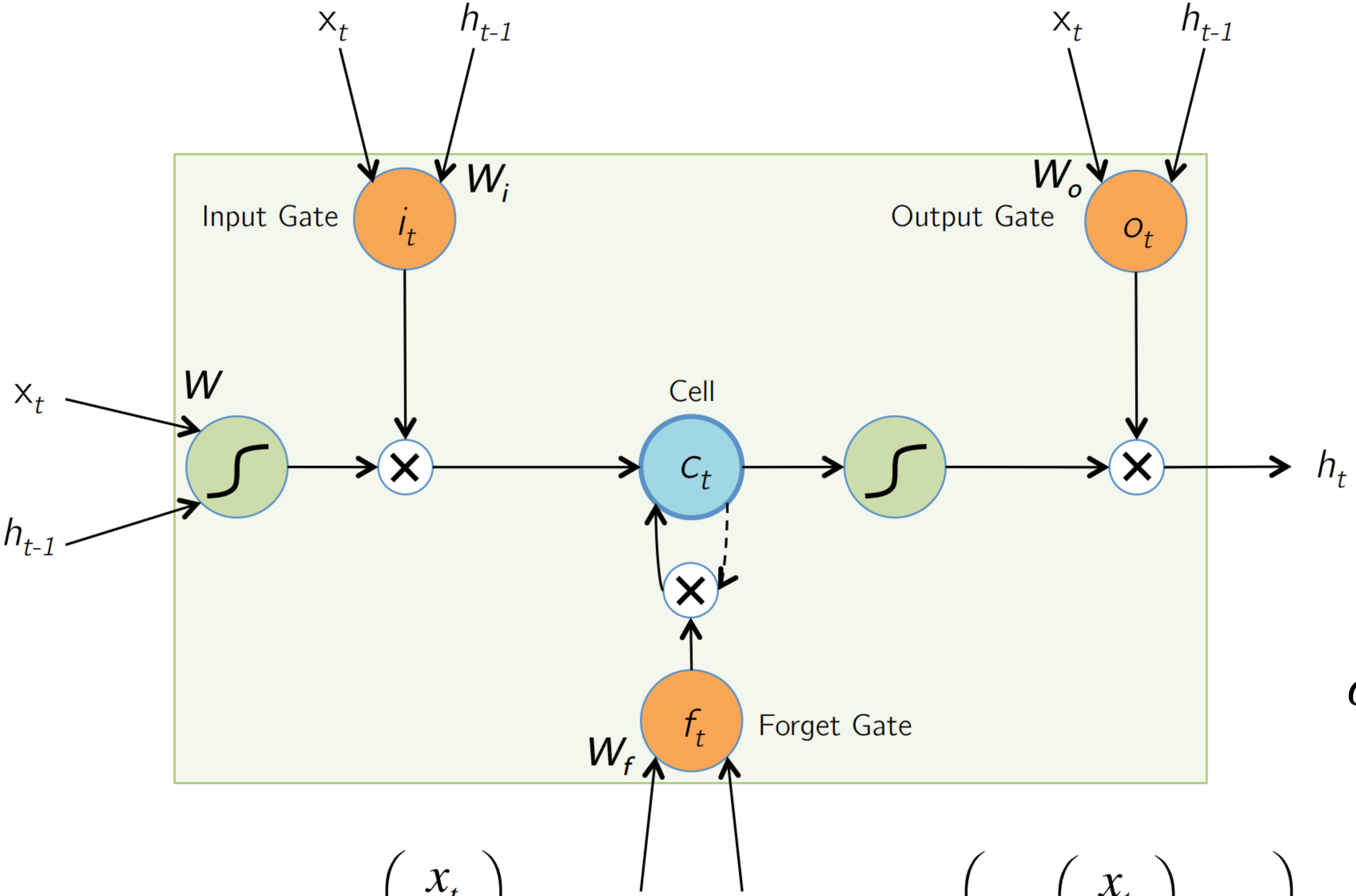
$\otimes$: element-wise multiplication

σ: sigmoid function

$$c_t = c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} \quad h_t = o_t \otimes \tanh c_t \quad i_t = \sigma\left(W_i \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_i \right)$$

Similarly for o_t

The Popular LSTM Cell



$\otimes$: element-wise multiplication

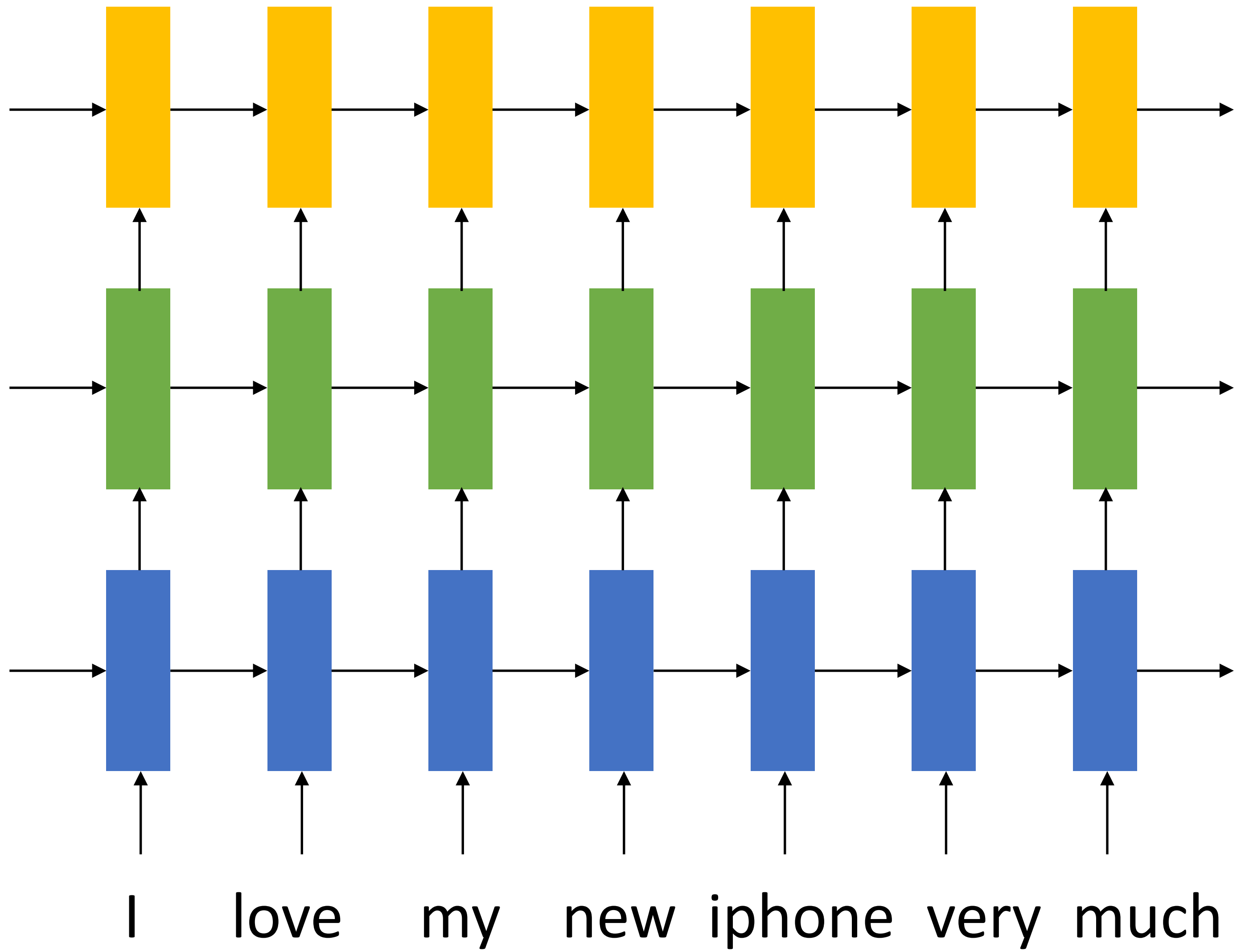
σ: sigmoid function

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

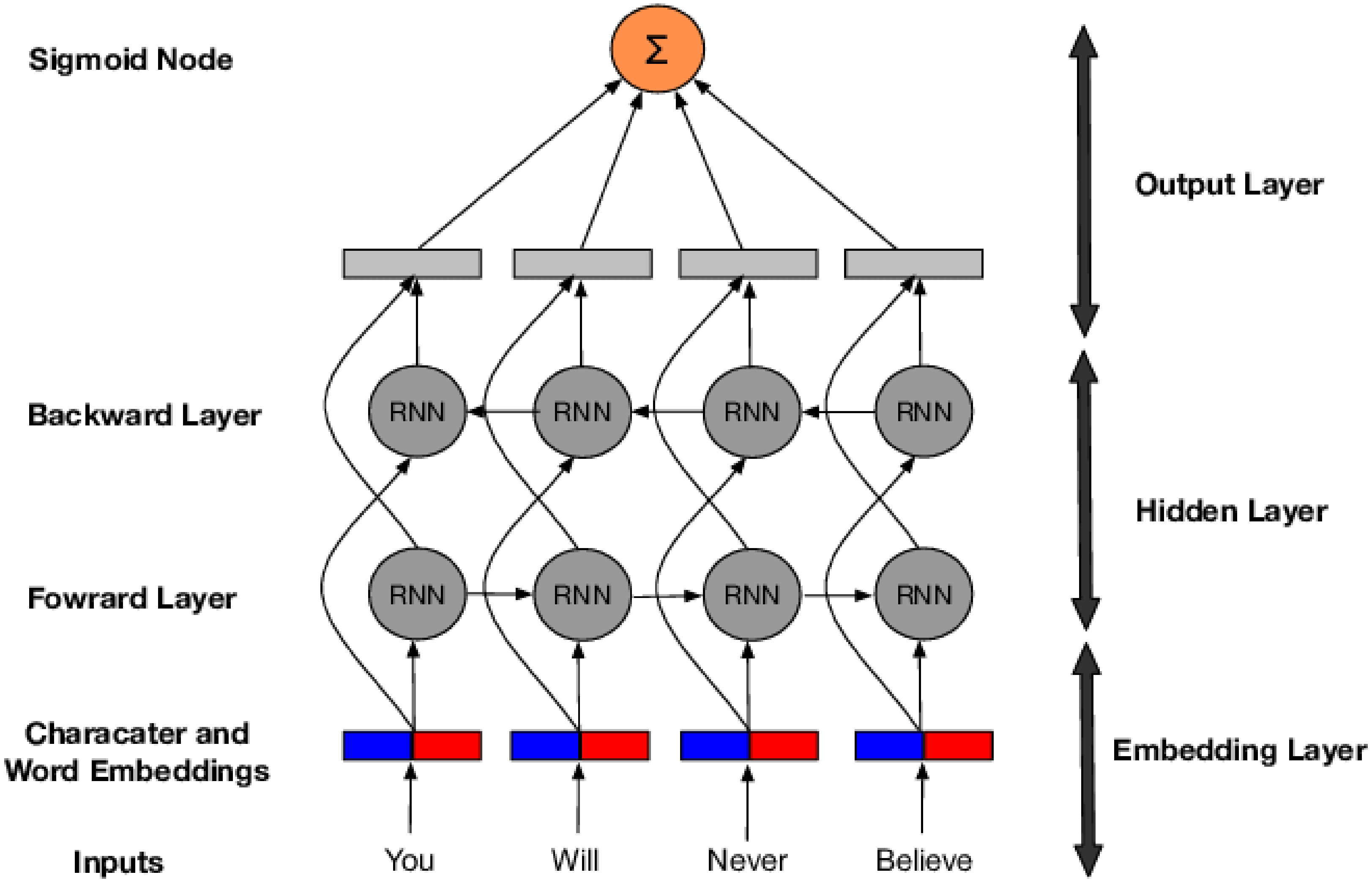
$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

Stacked RNN

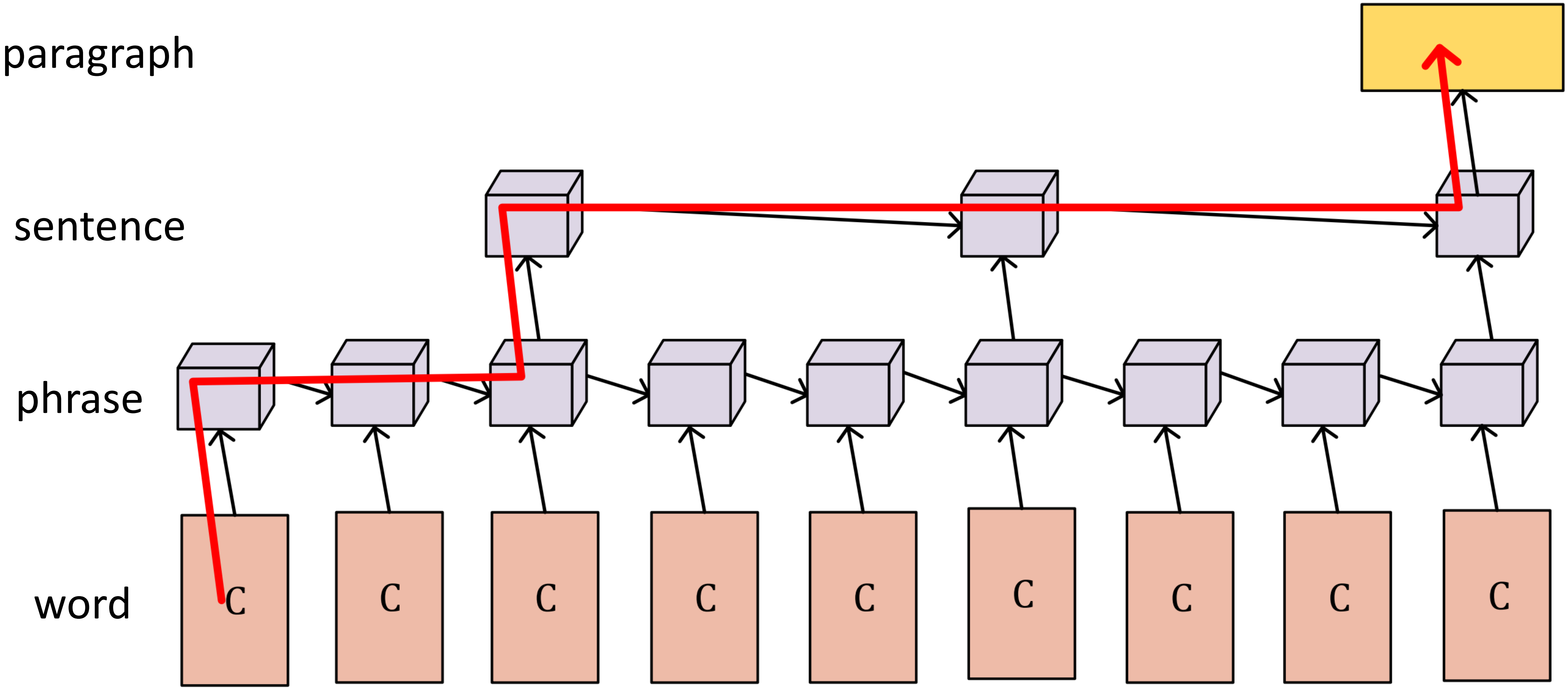
RNN is made of cells. In this case, a cell is an LSTM cell



Bidirectional RNN

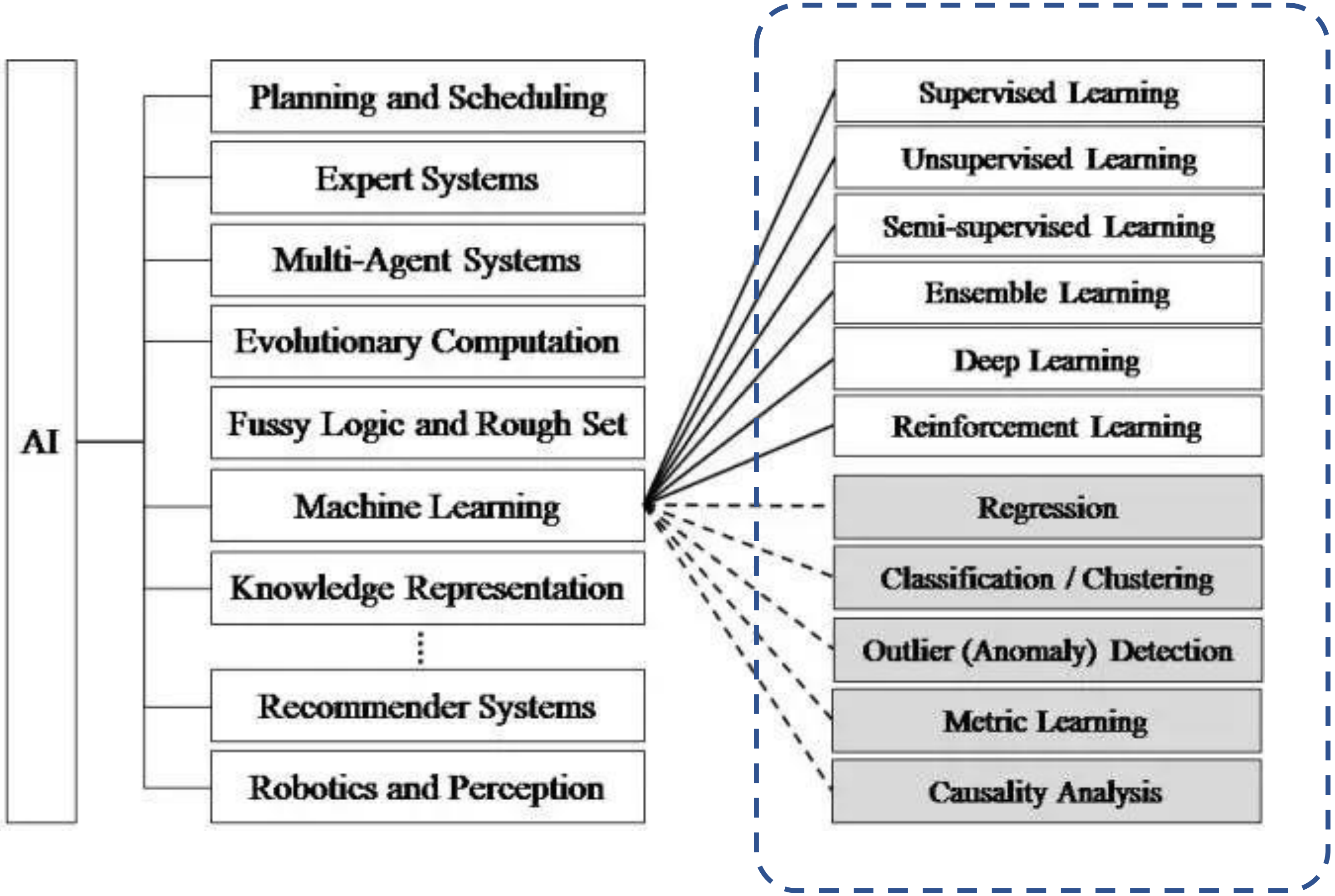


Hierarchical Recurrent Neural Network



Applications

Machine learning theories



Machine learning applications

- Computer vision (CV)
- Natural language processing (NLP)
- Speech processing

Machine Translation

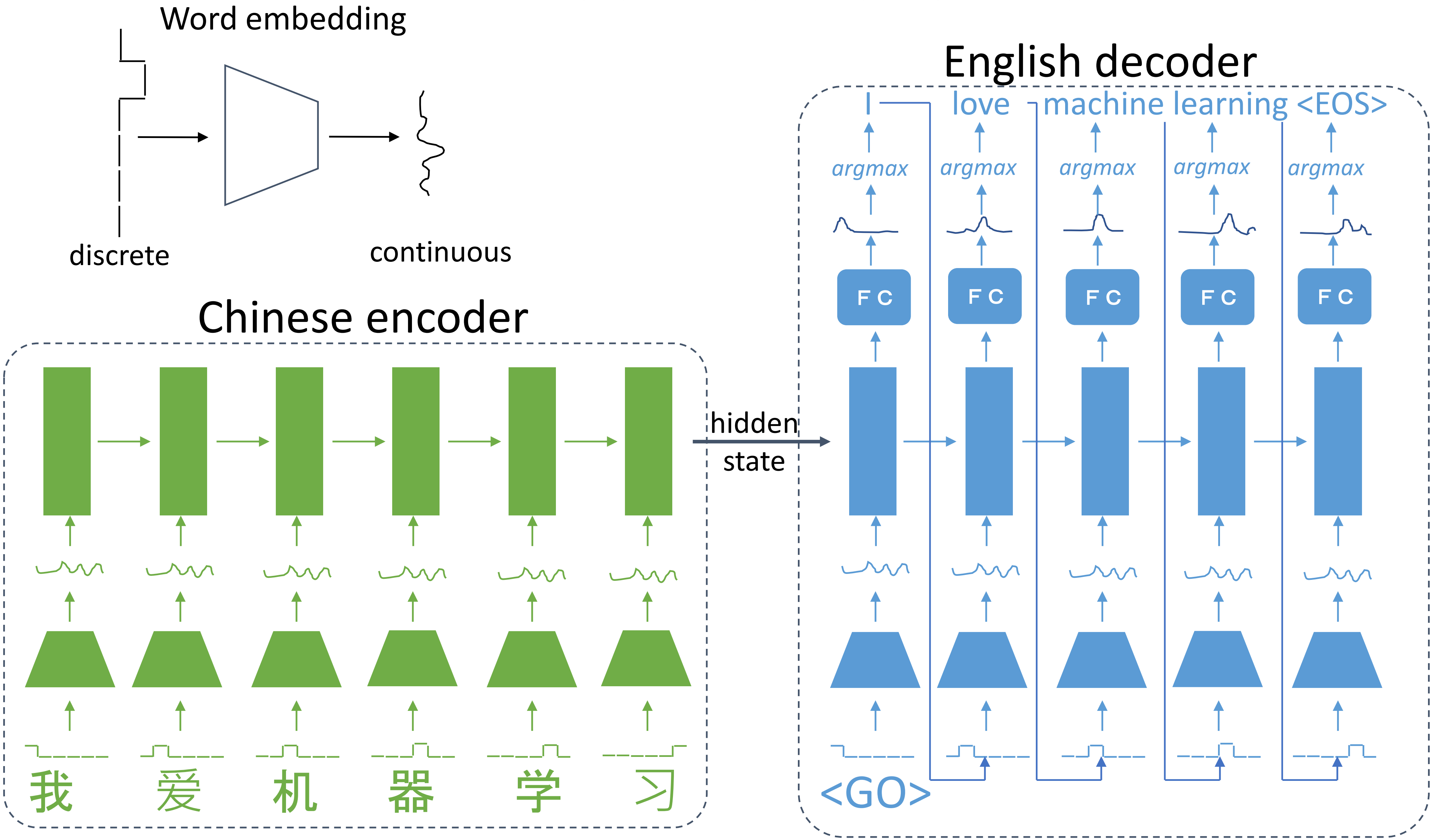
我爱机器学习 → I love machine learning

Preprocessing

ID	vocabulary	one-hot vector
0	我	100000
1	爱	010000
2	机	001000
3	器	000100
4	学	000010
5	习	000001

ID	vocabulary	one-hot vector
0	<GO>	100000
1	<EOS>	010000
2	I	001000
3	love	000100
4	machine	000010
5	learning	000001

Machine Translation



Vision and Language

Captioning

Two people are in a wheelchair and one is holding a racket.

VQA

Q: How many people on wheelchairs ?

A: Two

Q: How many wheelchairs ?

A: One

Visual Dialog

Q: How many people are on wheelchairs ?

A: Two

Q: What are their genders ?

A: One male and one female

Q: Which one is holding a racket ?

A: The woman

Visual Dialog

Q: What is the gender of the one in the white shirt ?

A: She is a woman

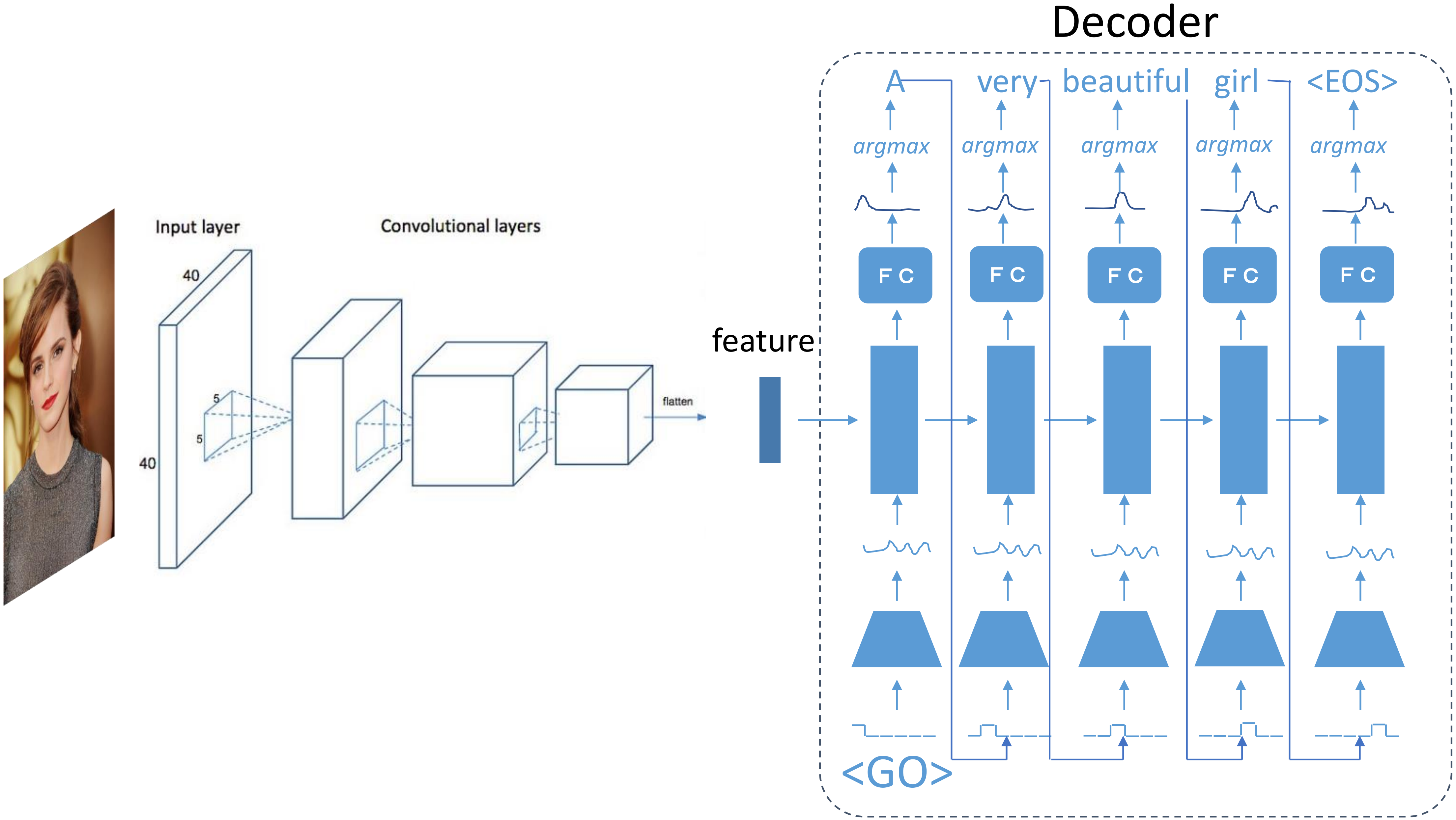
Q: What is she doing ?

A: Playing a Wii game

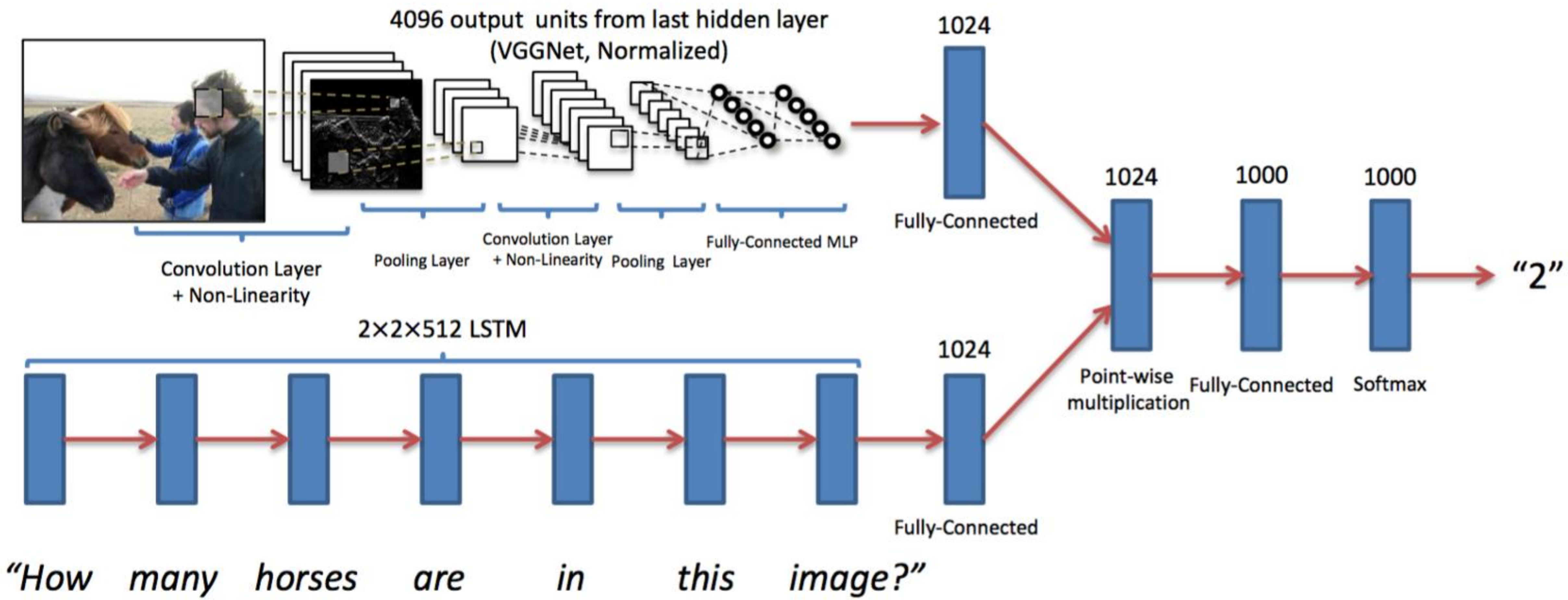
Q: Is that a man to her right

A: No, it’s a woman

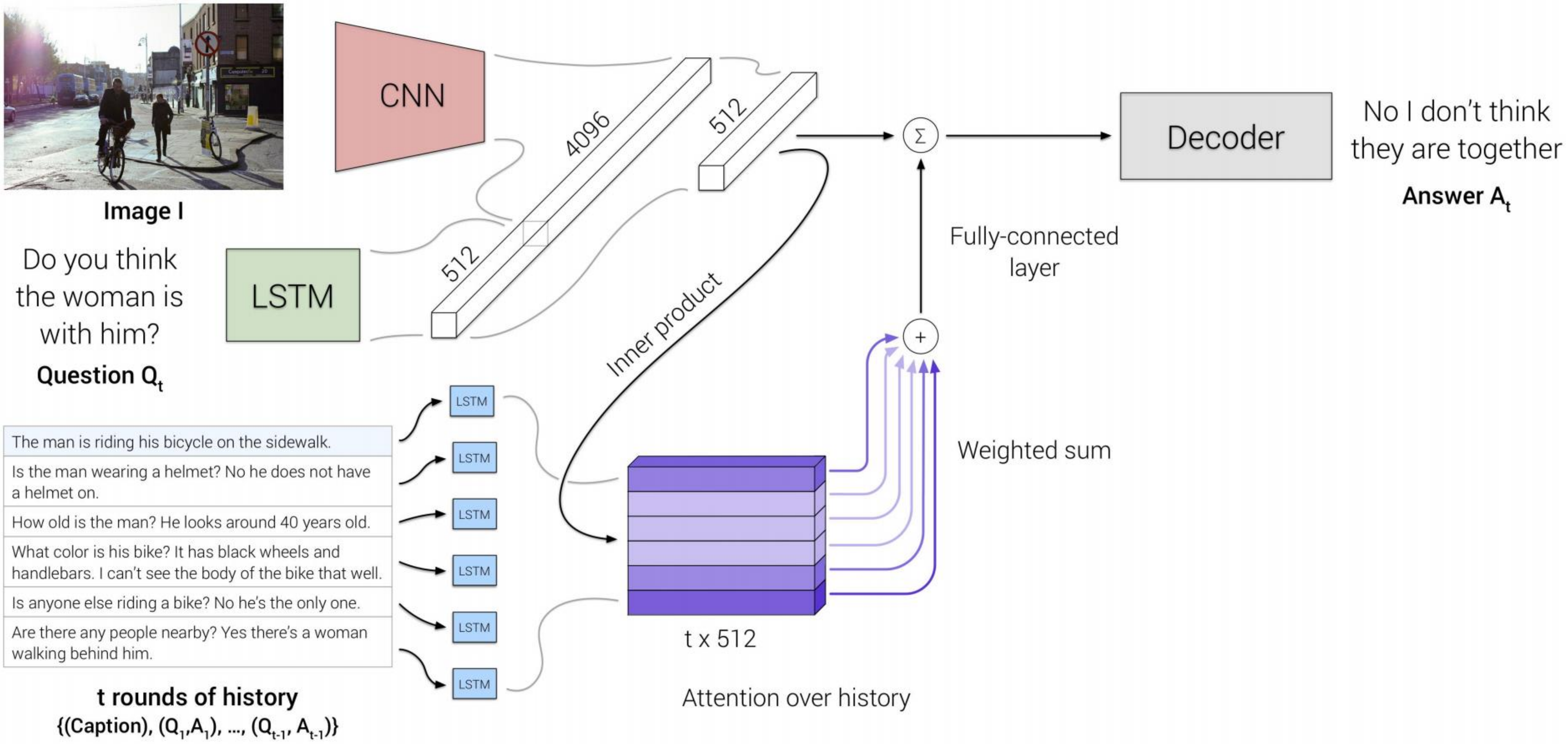
Image Captioning



Visual Question Answering (VQA)



Visual Dialog



References

- http://slazebni.cs.illinois.edu/spring17/lec02_rnn.pdf
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/

Thanks!