RNN & LSTM

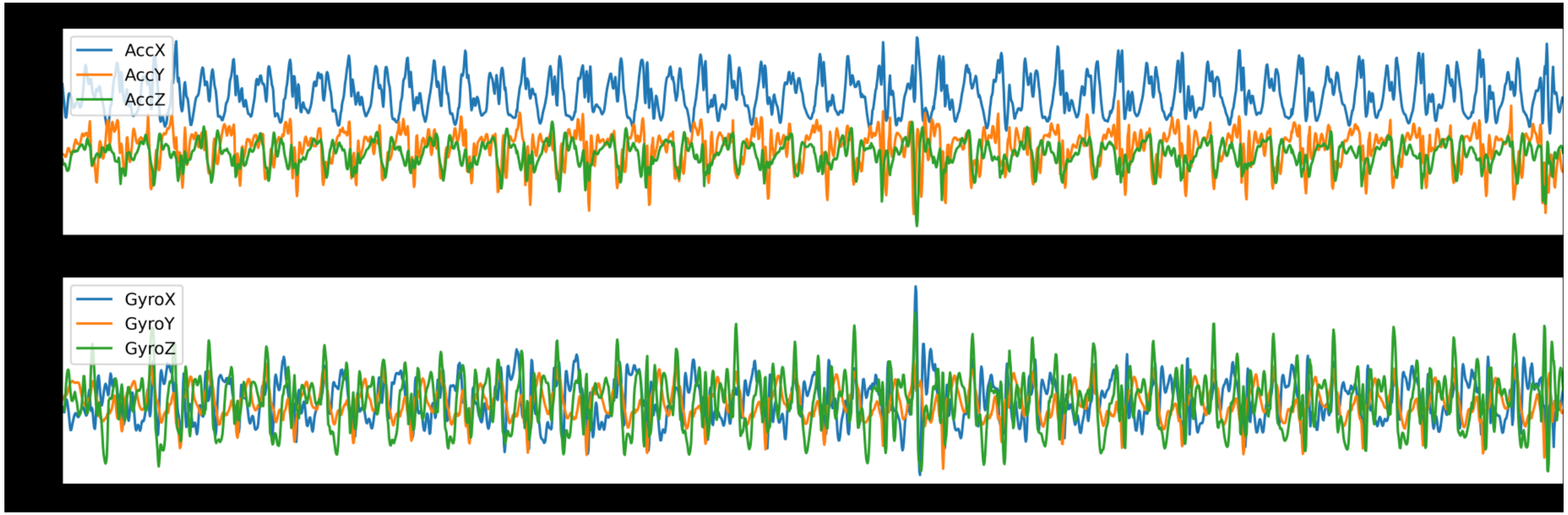
Nguyen Van Vinh
QAI, Fsoft

Content

- Recurrent Neural Network
- The vanishing/exploding gradient problem
- LSTM
- Applications for LSTM

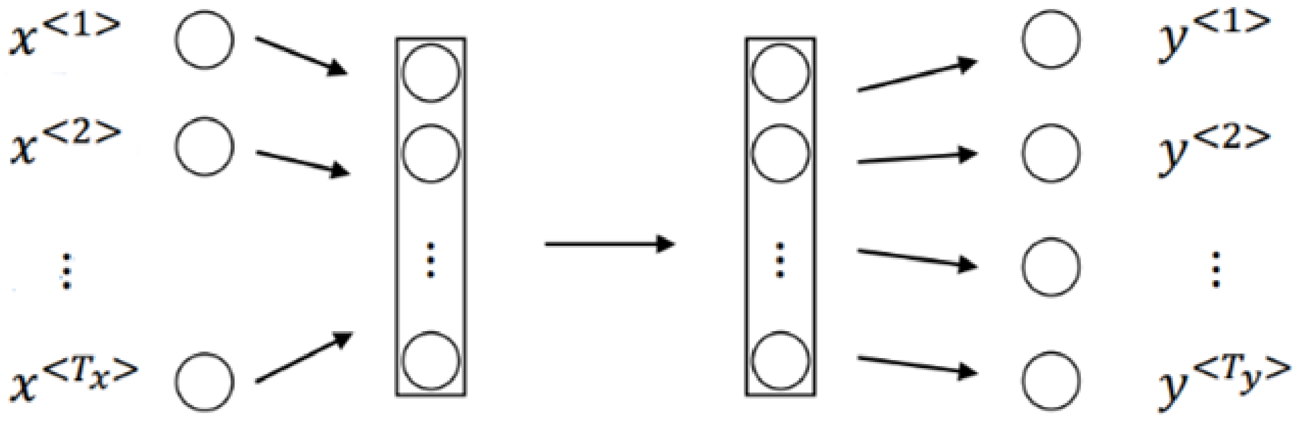
Sequence Data

- Time Series Data
- Natural Language



Data Source: https://dl.acm.org/doi/10.1145/2370216.2370438

Why not standard NN?



Problems:

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

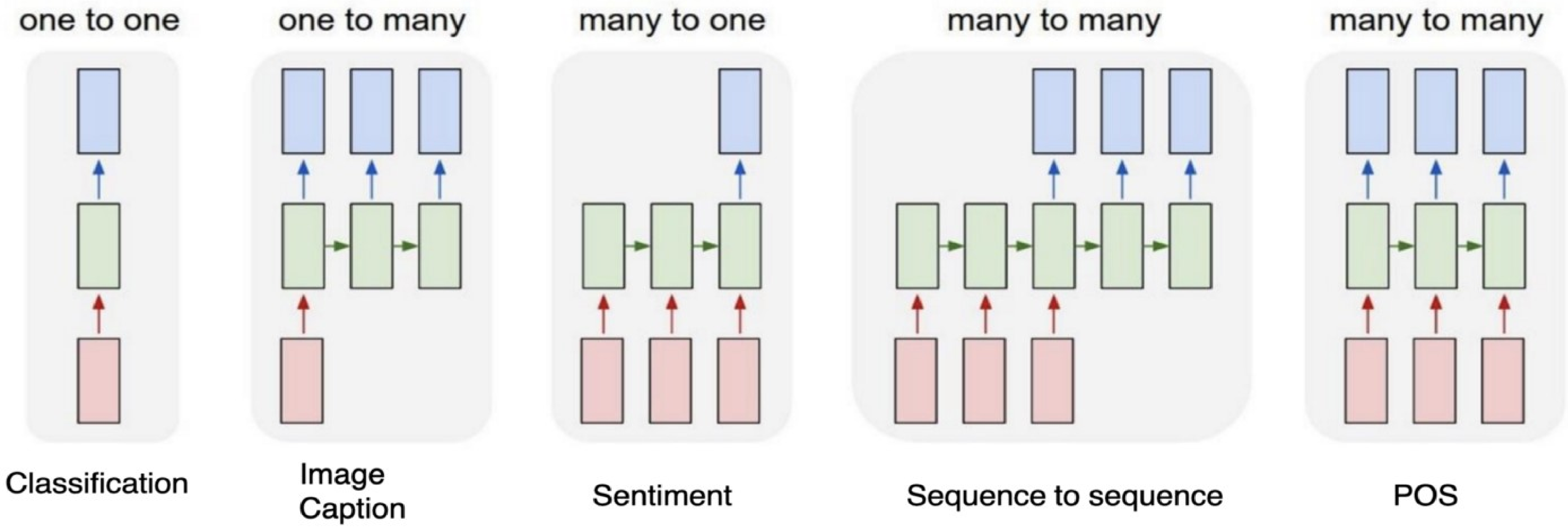
What is RNN?

A recurrent neural network (RNN) is a type of artificial neural network which used for sequential data or time series data.

Application:

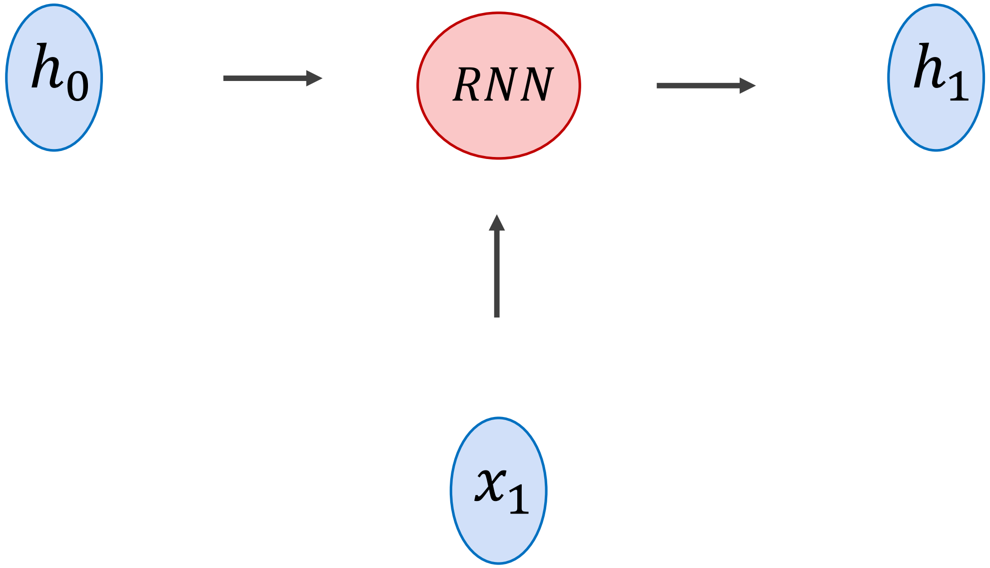
+ Language translation.
+ Natural language processing (NLP).
+ Speech recognition.
+ Image captioning.

Types of RNN



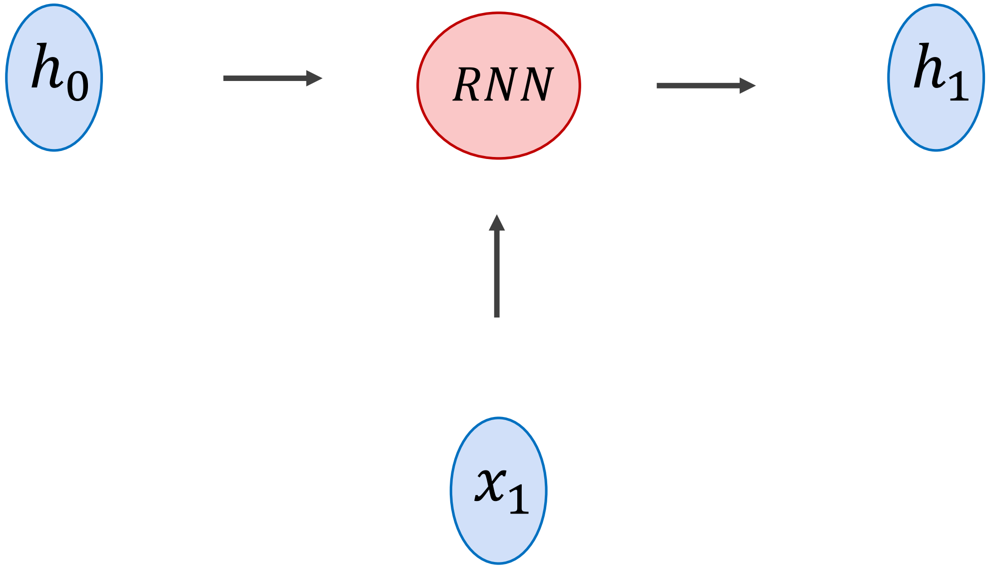
one to one
one to many
many to one
many to many
many to many
Classification
Image Caption
Sentiment
Sequence to sequence
POS

Recurrent Neural Network Cell

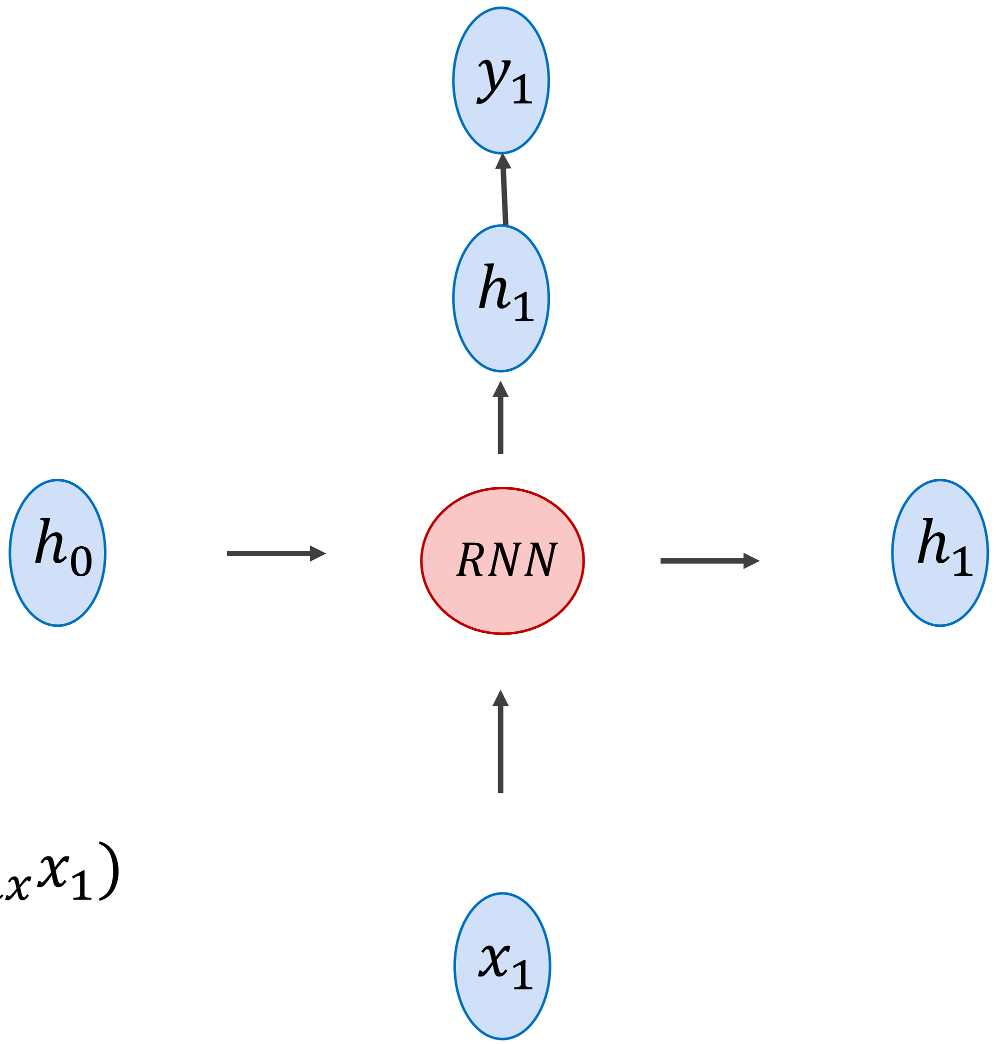


Recurrent Neural Network Cell

$$h_1 = \tanh(W_{hh}h_0 + W_{hx}x_1)$$



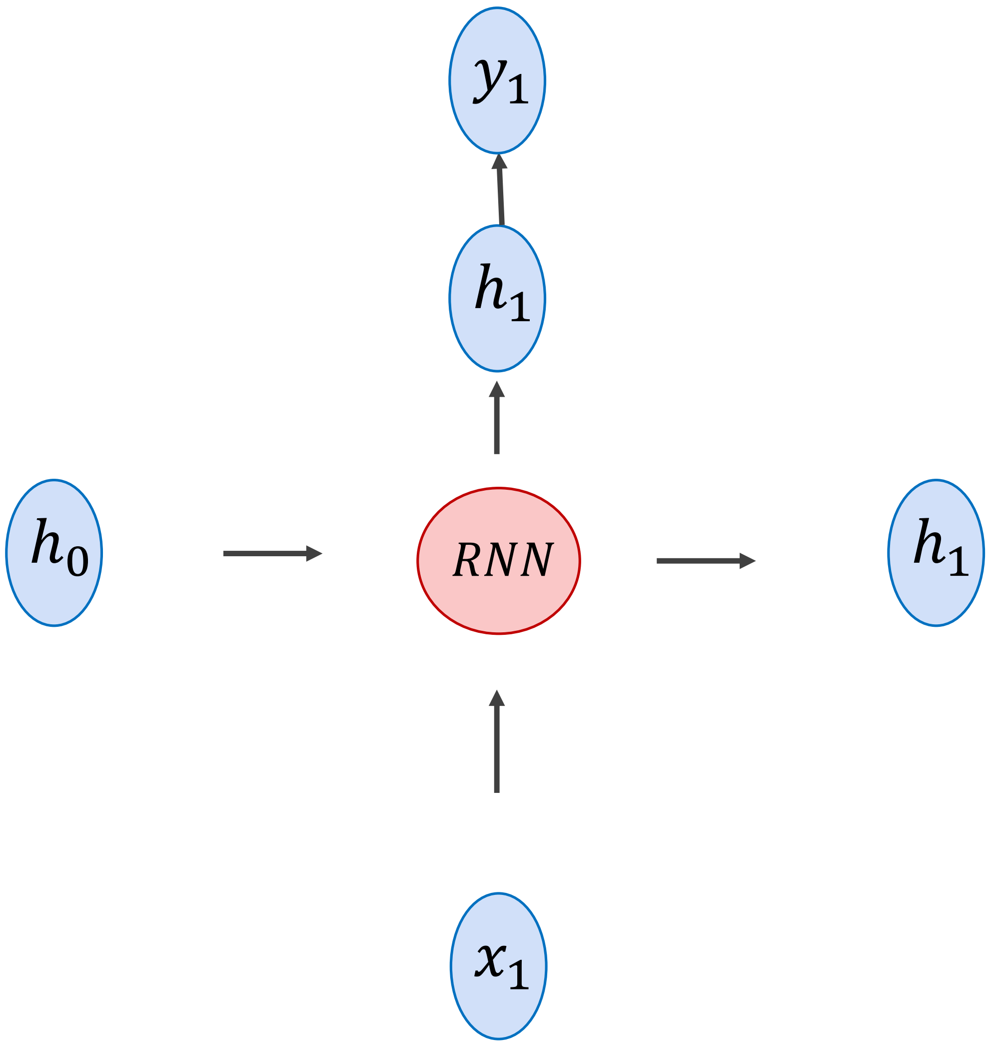
Recurrent Neural Network Cell



$$h_1 = \tanh(W_{hh}h_0 + W_{hx}x_1)$$

$$y_1 = \mathrm{softmax}(W_{hy}h_1)$$

Recurrent Neural Network Cell



Recurrent Neural Network Cell

$y_1 = [0.1, 0.05, 0.05, 0.1, 0.7]$ → **e (0.7)**

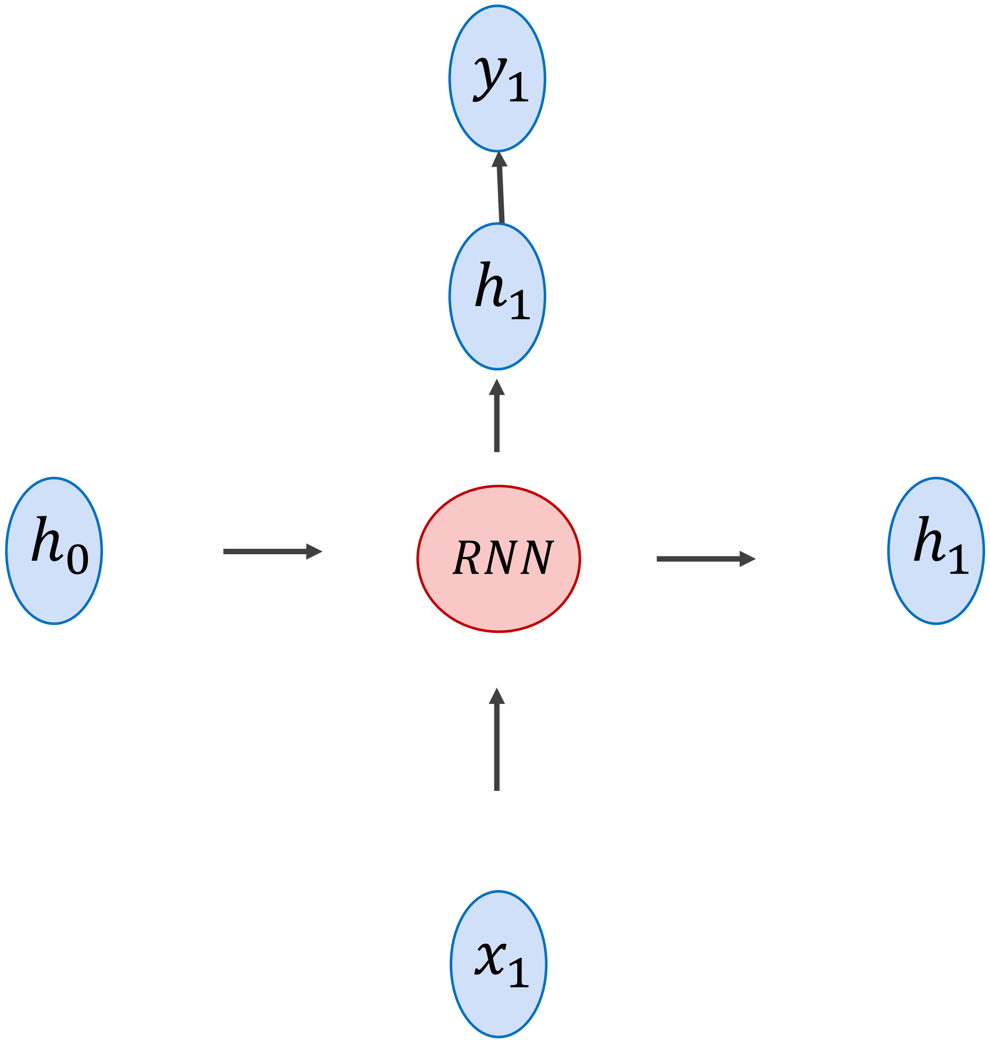
$h_1 = [0.1 \quad 0.2 \; 0 \; -0.3 \; -0.1\,]$

$h_0 = [0\;0\;0\;0\;0\,]$ → RNN → $h_1 = [0.1 \quad 0.2 \;\; 0 \; -0.3 \; -0.1\,]$

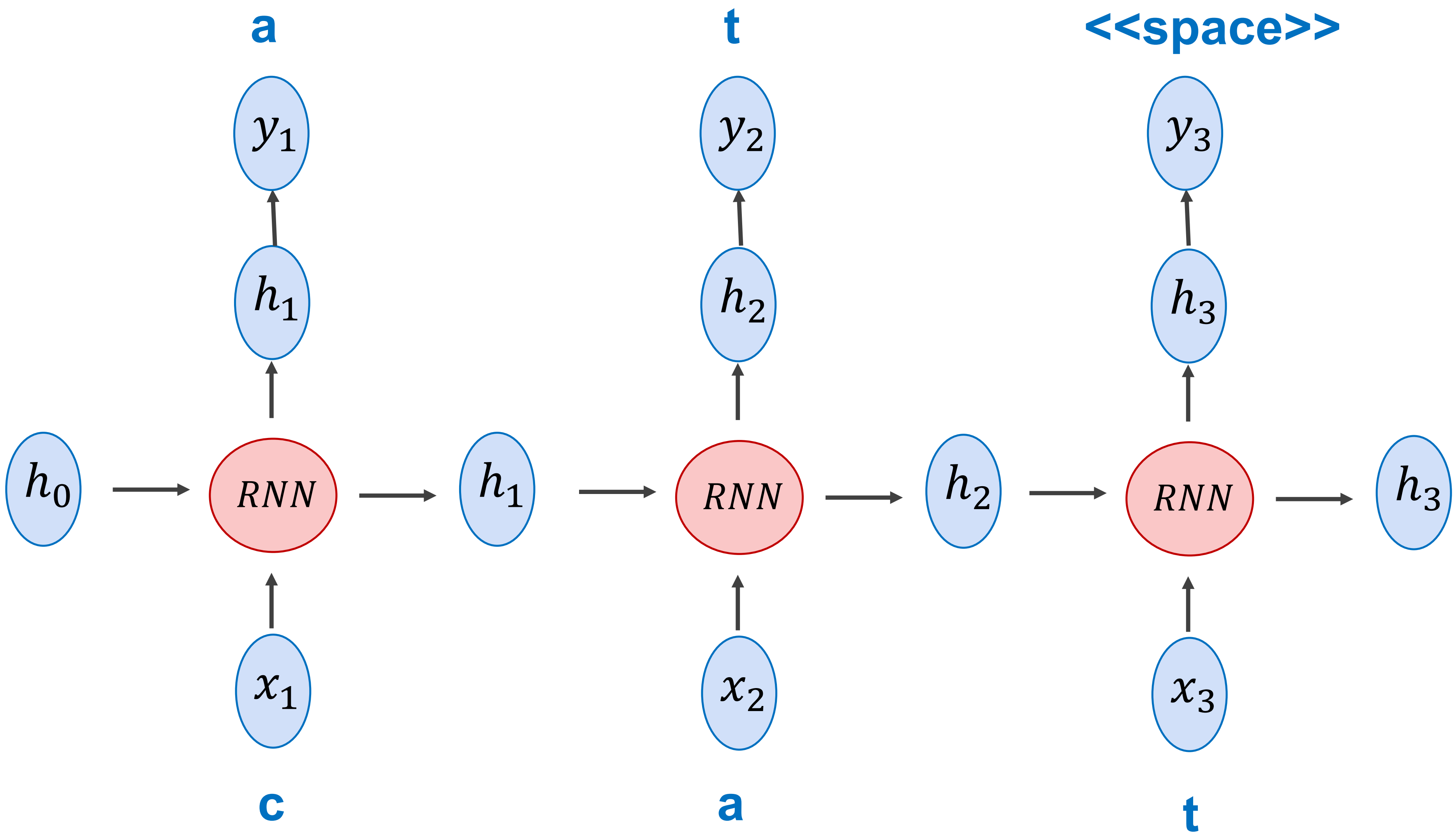
$x_1 = \; [0\;0\;1\;0\;0]$

a b **c** d e

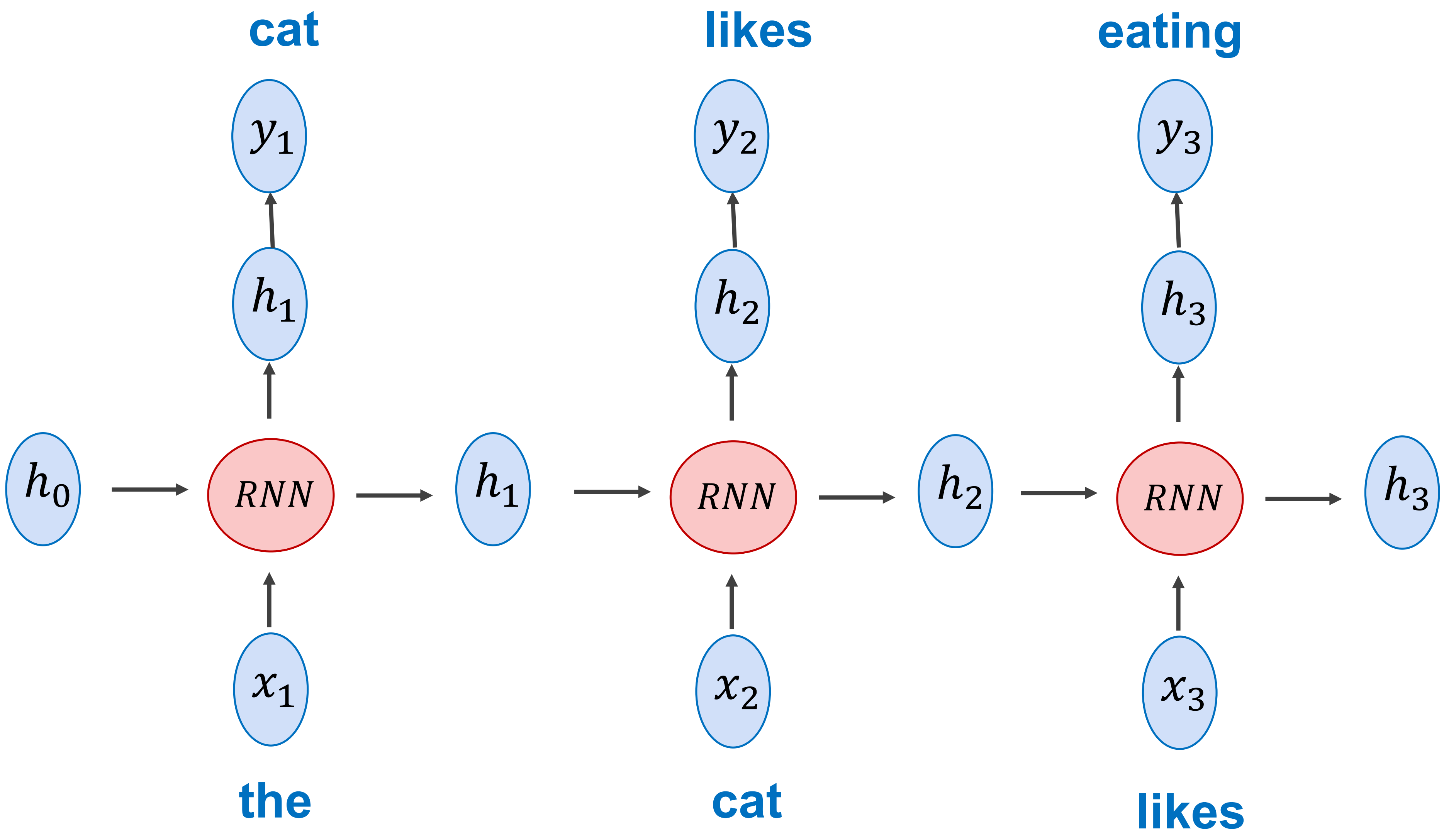
Recurrent Neural Network Cell



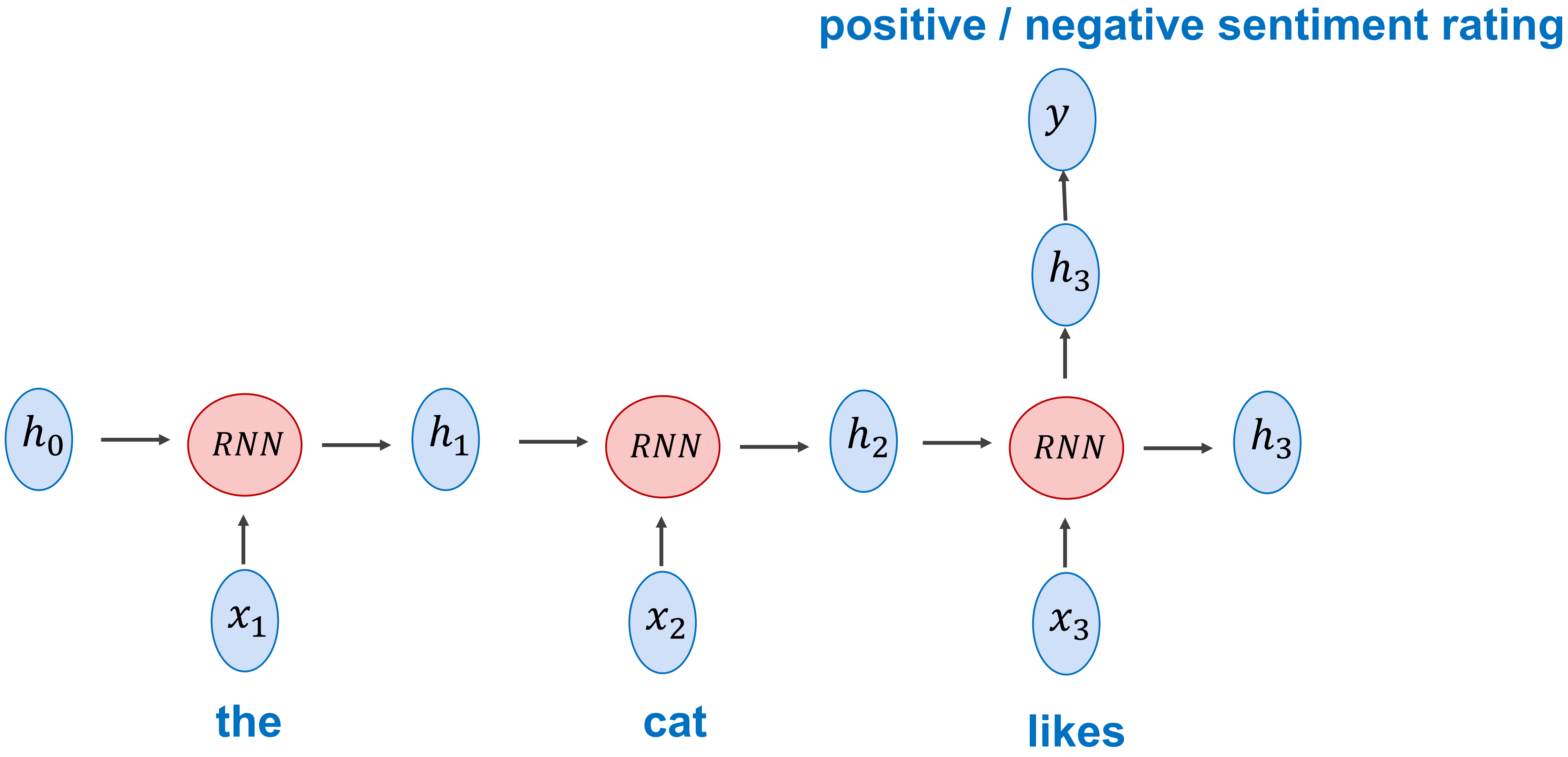
(Unrolled) Recurrent Neural Network



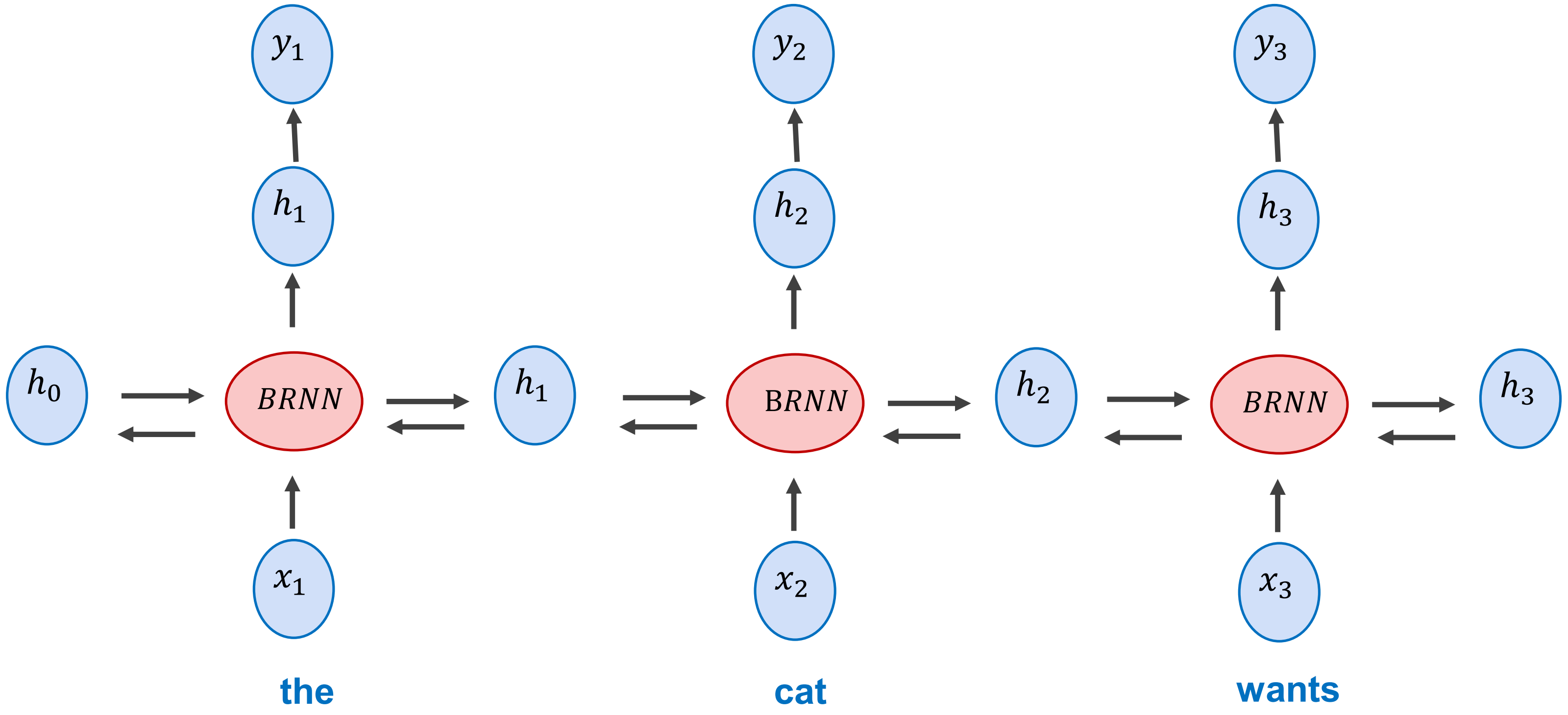
(Unrolled) Recurrent Neural Network



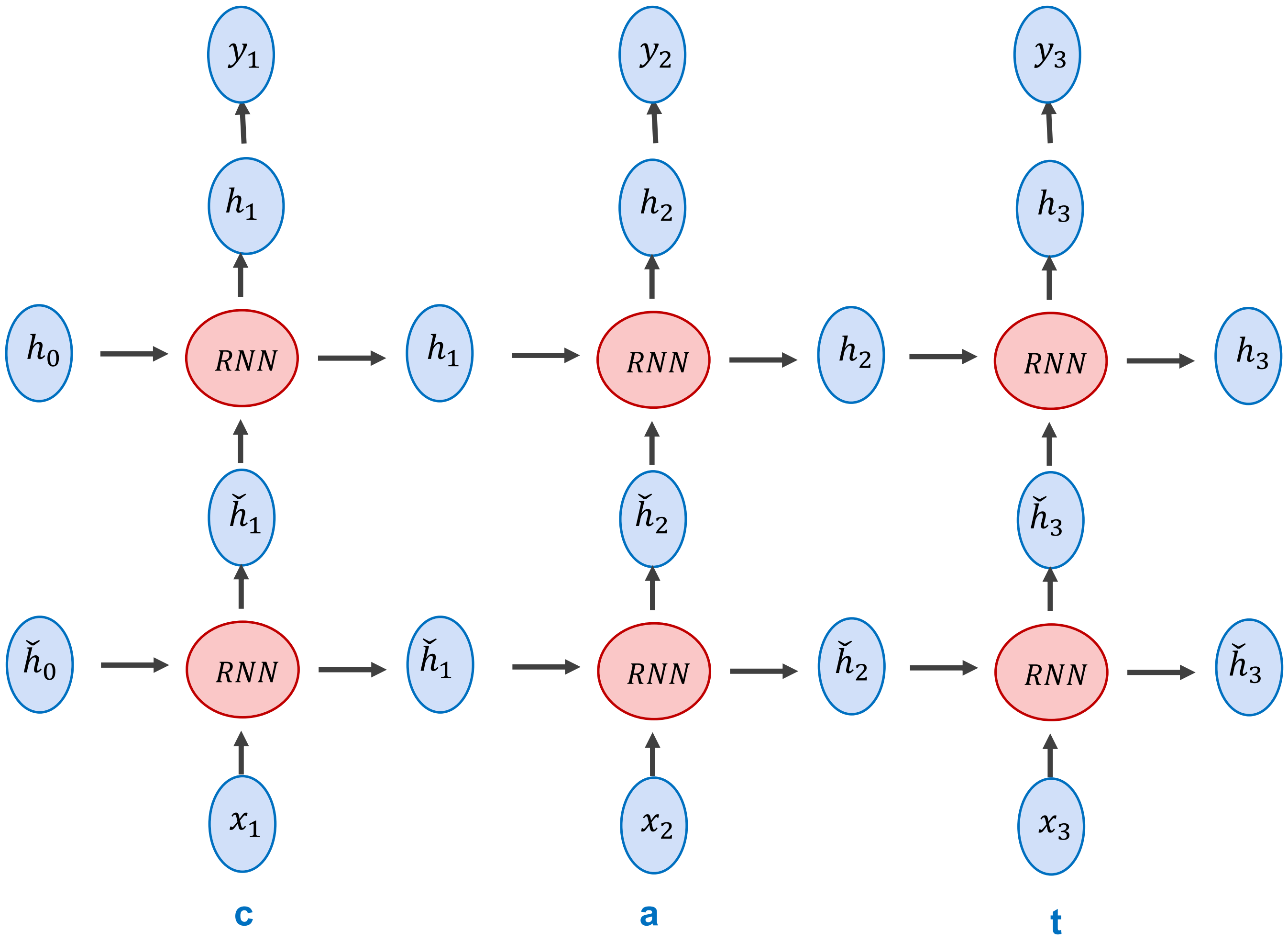
(Unrolled) Recurrent Neural Network

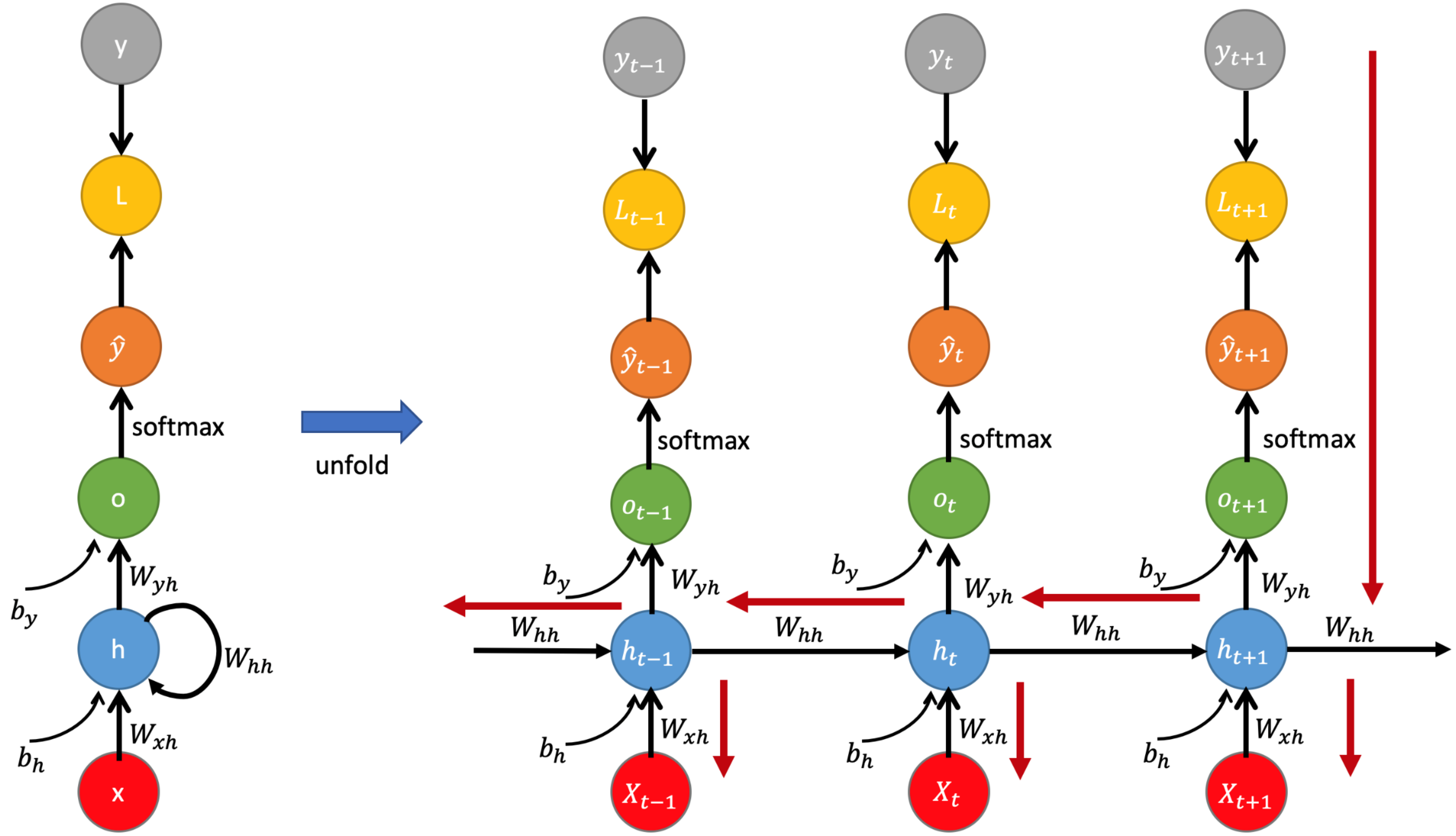


Bidirectional Recurrent Neural Network

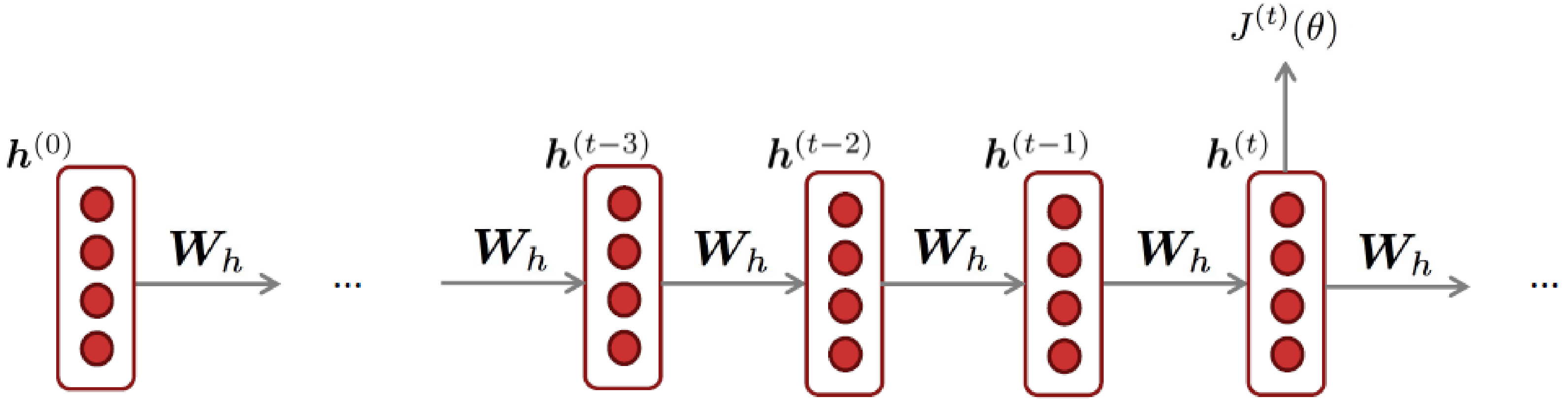


Stacked Recurrent Neural Network




y
L
$\hat{y}$
softmax
o
b_y
W_{yh}
h
W_{hh}
b_h
W_{xh}
x
unfold
y_{t-1}
y_t
y_{t+1}
L_{t-1}
L_t
L_{t+1}
$\hat{y}_{t-1}$
$\hat{y}_t$
$\hat{y}_{t+1}$
softmax
softmax
softmax
o_{t-1}
o_t
o_{t+1}
b_y
W_{yh}
b_y
W_{yh}
b_y
W_{yh}
W_{hh}
W_{hh}
W_{hh}
W_{hh}
h_{t-1}
h_t
h_{t+1}
b_h
W_{xh}
b_h
W_{xh}
b_h
W_{xh}
X_{t-1}
X_t
X_{t+1}

Backpropagation for RNNs



Question: What's the derivative of $J^{(t)}(\theta)$ w.r.t. the repeated weight matrix $\boldsymbol{W}_h$?

Answer: $$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \left. \frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} \right|_{(i)}$$

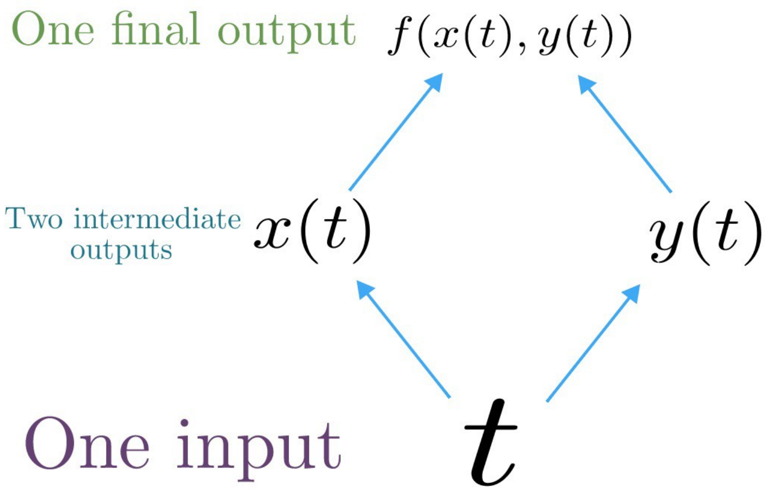
"The gradient w.r.t. a repeated weight is the sum of the gradient w.r.t. each time it appears"

Why?

Qui tắc chuỗi đa biến

- Given a multivariable function $f(x, y)$, and two single variable functions $x(t)$ and $y(t)$, here's what the multivariable chain rule says:

$$\underbrace{\frac{d}{dt} f(x(t), y(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$



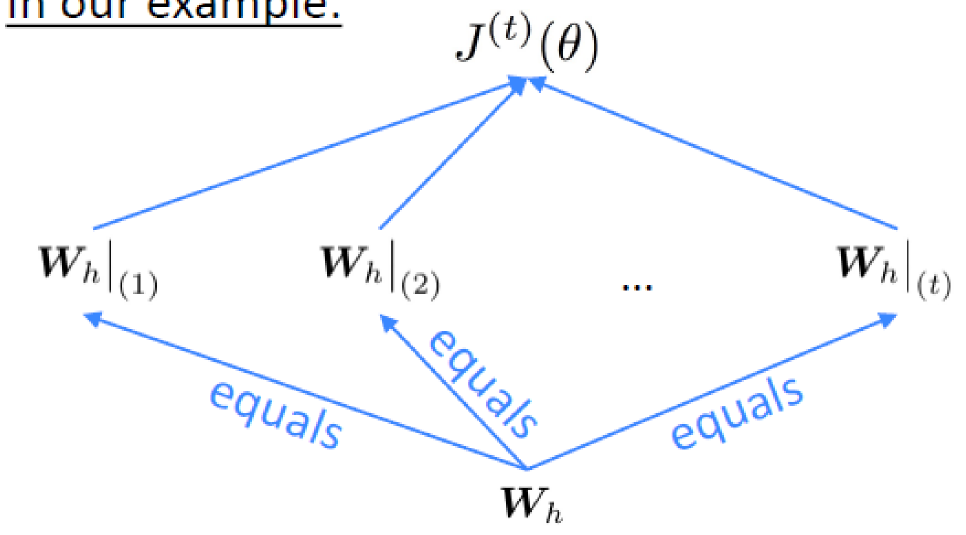
Nguồn: https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/differentiating-vector-valued-functions/a/multivariable-chain-rule-simple-version

Backpropagation for RNNs

- Given a multivariable function $f(x, y)$, and two single variable functions $x(t)$ and $y(t)$, here's what the multivariable chain rule says:

$$\underbrace{\frac{d}{dt} f(x(t), y(t))}_{\text{Derivative of composition function}} = \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt}$$

In our example:



Apply the multivariable chain rule:

$$\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h} = \sum_{i=1}^{t} \left.\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)} \overset{= 1}{\boxed{\frac{\partial \boldsymbol{W}_h|_{(i)}}{\partial \boldsymbol{W}_h}}}$$

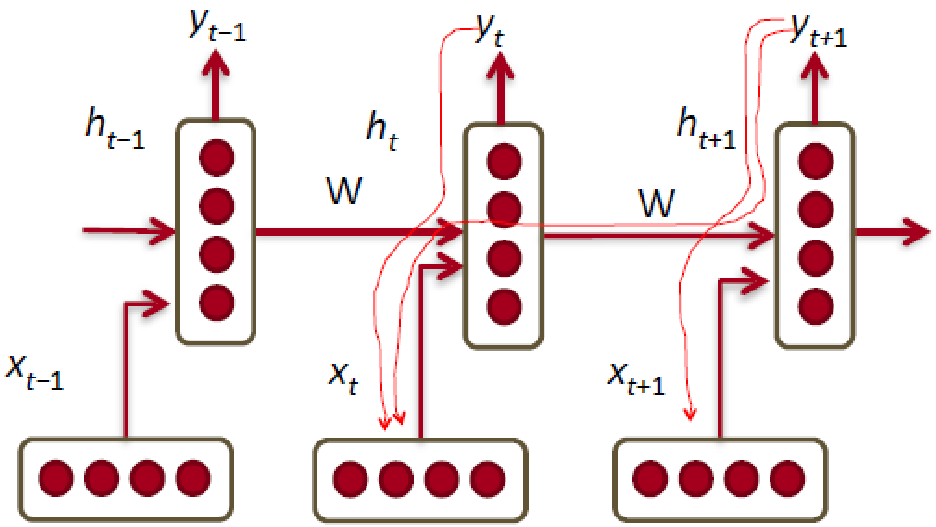
$$= \sum_{i=1}^{t} \left.\frac{\partial J^{(t)}}{\partial \boldsymbol{W}_h}\right|_{(i)}$$

Bài tập thực hành

- **RNN for sentiment**
- **Data set: IMDB**

The vanishing/exploding gradient problem

- Multiply the same matrix at each time step during backprop



The vanishing gradient problem

- Similar but simpler RNN formulation:

$$
\begin{aligned}
h_t &= W f(h_{t-1}) + W^{(hx)} x_{[t]} \\
\hat{y}_t &= W^{(S)} f(h_t)
\end{aligned}
$$

- Total error is the sum of each error at time steps t

$$\frac{\partial E}{\partial W} = \sum_{t=1}^{T} \frac{\partial E_t}{\partial W}$$

- Hardcore chain rule application:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_k}{\partial W}$$

The vanishing gradient problem

- Useful for analysis we will look at:

$$\frac{\partial E_t}{\partial W} = \sum_{k=1}^{t} \frac{\partial E_t}{\partial y_t} \frac{\partial y_t}{\partial h_t} \boxed{\frac{\partial h_t}{\partial h_k}} \frac{\partial h_k}{\partial W}$$

- Remember

$$h_t = W f(h_{t-1}) + W^{(hx)} x_{[t]}$$

- More chain rule, remember:

$$\frac{\partial h_t}{\partial h_k} = \prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}$$

- The gradient is a product of Jacobian matrices, each associated with a step in the forward computation.

$$\left\|\frac{\partial h_t}{\partial h_k}\right\| = \left\|\prod_{j=k+1}^{t} \frac{\partial h_j}{\partial h_{j-1}}\right\| \leq (\beta_W \beta_h)^{t-k}$$

$\Rightarrow$ **Vanishing or exploding gradient**

The vanishing gradient problem for language models

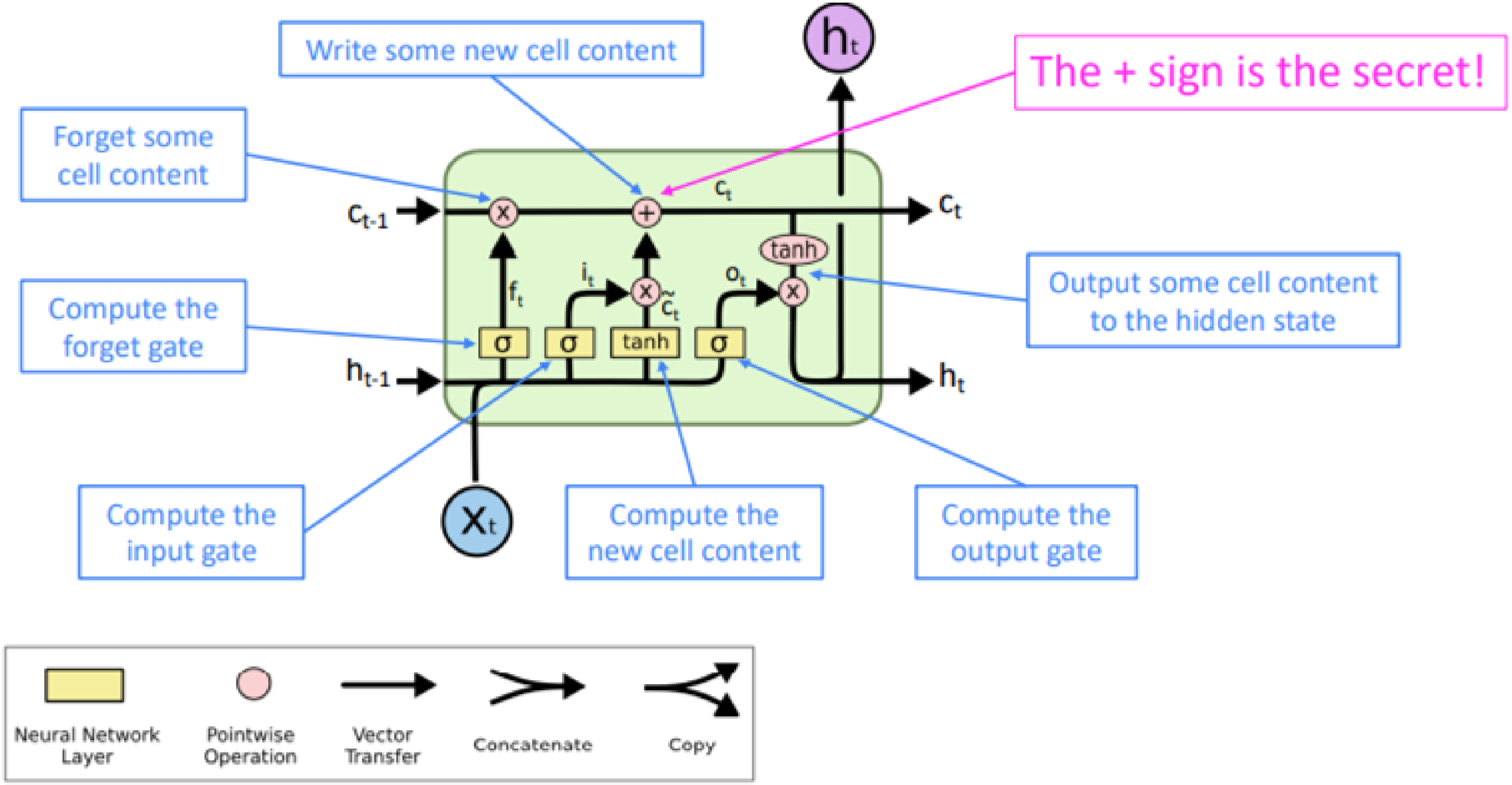
- In the case of language modeling or question answering words from time steps far away are not taken into consideration when training to predict the next word
- **Example:**

Jane walked into the room. John walked in too. It was late in the day. Jane said hi to ____

Vanishing/Exploding Solutions

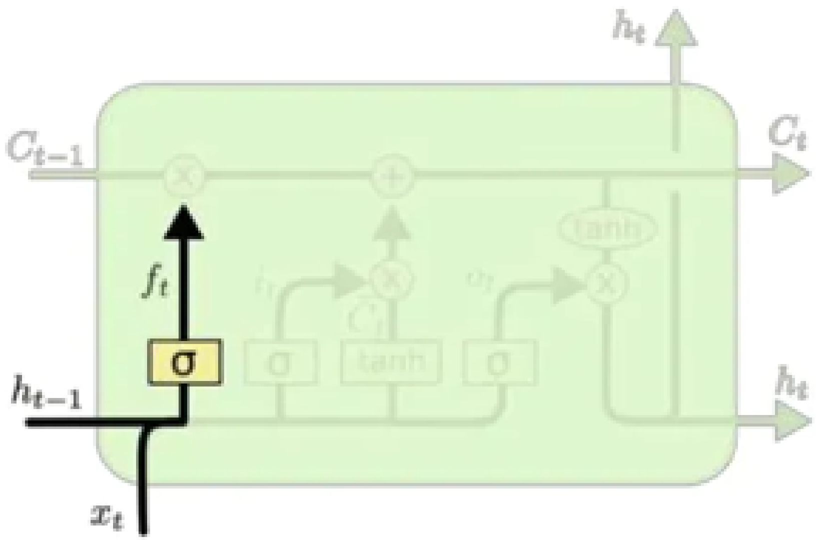
- **Vanishing Gradient:**
  - Gating mechanism (LSTM, GRU)
  - Attention mechanism (Transformer)
  - Adding skip connection through time
  - Better Initialization

Long-Short Term Memory (LSTM)



Architecture of LSTM cell

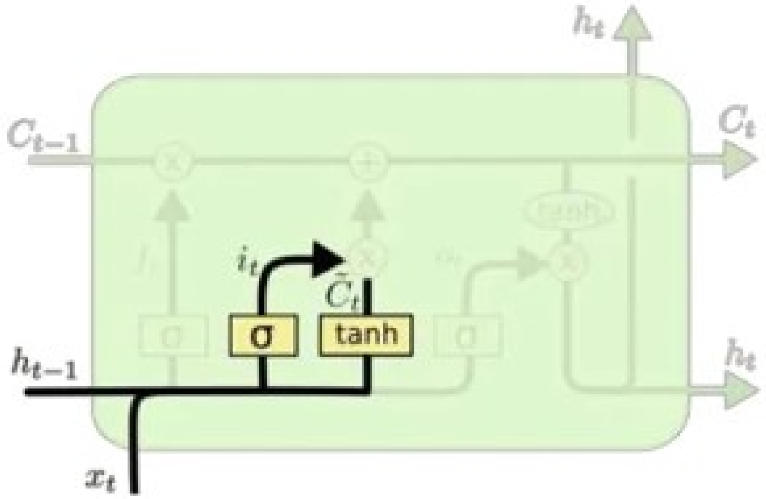
- **Forget information:**
  - Decide what information throw away from the cell state
  - **Forget gate layer:**
    - Output a number between 0 and 1



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

Architecture of LSTM cell

- **Add new information:**
  - Decide what new information store in the cell state
  - **Input gate layer:**
    - Decides which values we'll update
  - **Tanh layer:**
    - creates a vector of new candidate values, $\tilde{C}_t$

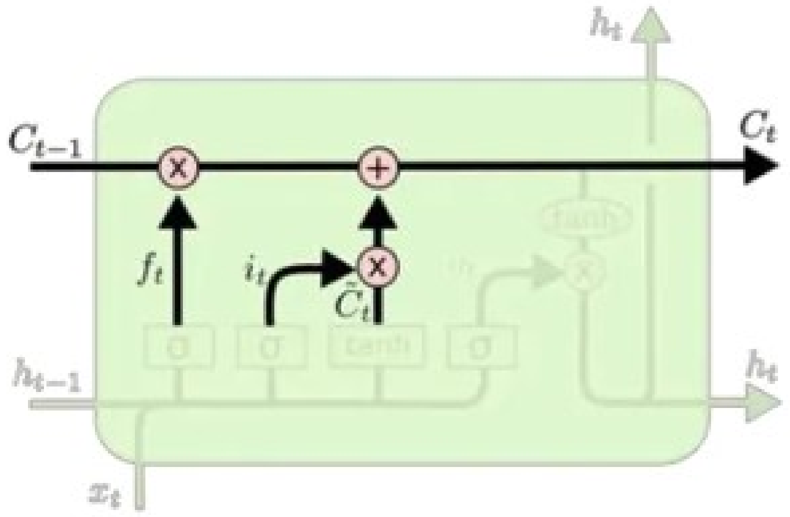


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \;+\; b_i\right)$$

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

Architecture of LSTM cell

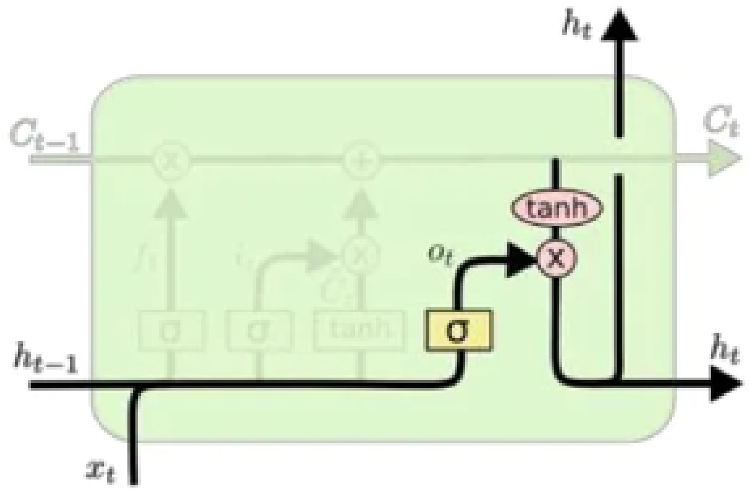
- **Update cell state:**
  - Forgetting the things we decided to forget earlier: $f_t * C_{t-1}$
  - Adding information we decide to add: $i_t * \tilde{C}_t$



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

Architecture of LSTM cell

- **Create output:**
  - Decide what we're going to output
  - **Output gate layer:**
    - Decides what parts of the cell state we're going to output
  - **Tanh layer:**
    - Push the values between -1 and +1



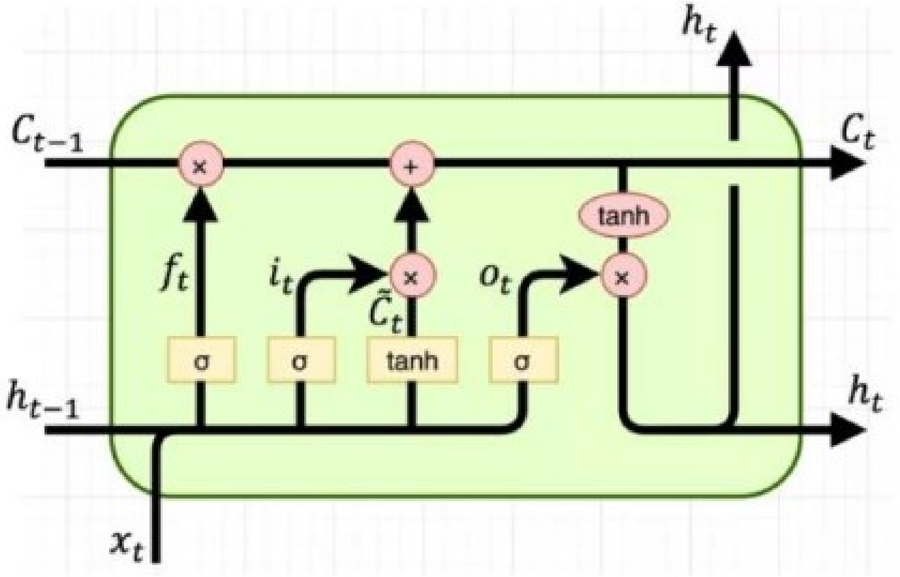
$$o_t = \sigma\left(W_o \, [h_{t-1}, x_t] \; + \; b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

Shadow state/ Short memory

Architecture of LSTM cell

- **Conclusion**:
  - Step 1: Forget gate layer.
  - Step 2: Input gate layer.
  - Step 3: Combine step 1 & 2.
  - Step 4: Output the cell state.

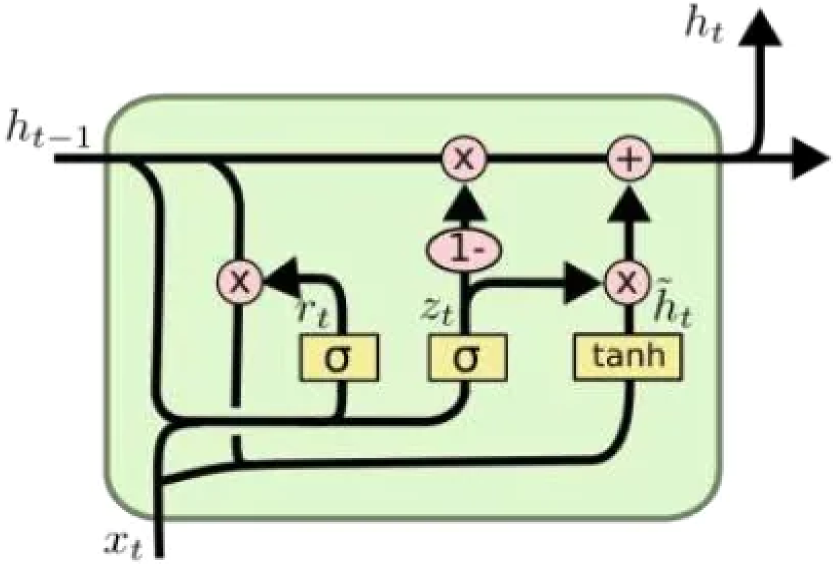


How does LSTM can solve vanishing gradient

- The LSTM architecture makes it **easier** for the RNN to **preserve** information **over many timesteps**.
- LSTM ***doesn't guarantee*** that there is **no vanishing/ exploding** gradient.
- LSTM provides an **easier way** for the model to learn **long-distance dependencies**.

LSTM Variations (GRU)

- **Gated Recurrent Unit (GRU)**
  - Combine the forget and input layer into a single “update gate”
  - Merge the cell state and the hidden state
  - Simpler.



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Compare LSTM vs. GRU

- **GRUs train faster** and perform better than LSTMs on **less training data** if you are doing language modeling (not sure about other tasks).
- **GRUs are simpler** and thus easier to modify, for example adding new gates in case of additional input to the network. It's just less code in general.
- **LSTMs** should in theory **remember longer sequences** than GRUs and outperform them in tasks requiring modeling long-distance relations.

Successful Applications of LSTMs

- Speech recognition: Language and acoustic modeling
- Sequence labeling
  - POS Tagging
    https://www.aclweb.org/aclwiki/index.php?title=POS_Tagging_(State_of_the_art)
  - NER
  - Phrase Chunking
- Neural syntactic and semantic parsing
- Image captioning: CNN output vector to sequence
- Sequence to Sequence
  - Machine Translation (Sustkever, Vinyals, & Le, 2014)
  - Video Captioning (input sequence of CNN frame outputs)

Bài tập thực hành

- **LSTM for sentiment**
- **Data set: IMDB**
-

Summary

- Recurrent Neural Network is one of the best deep NLP model families
- Most important and powerful RNN extensions with LSTMs and GRUs

Question and Discussion!