

Deep Learning



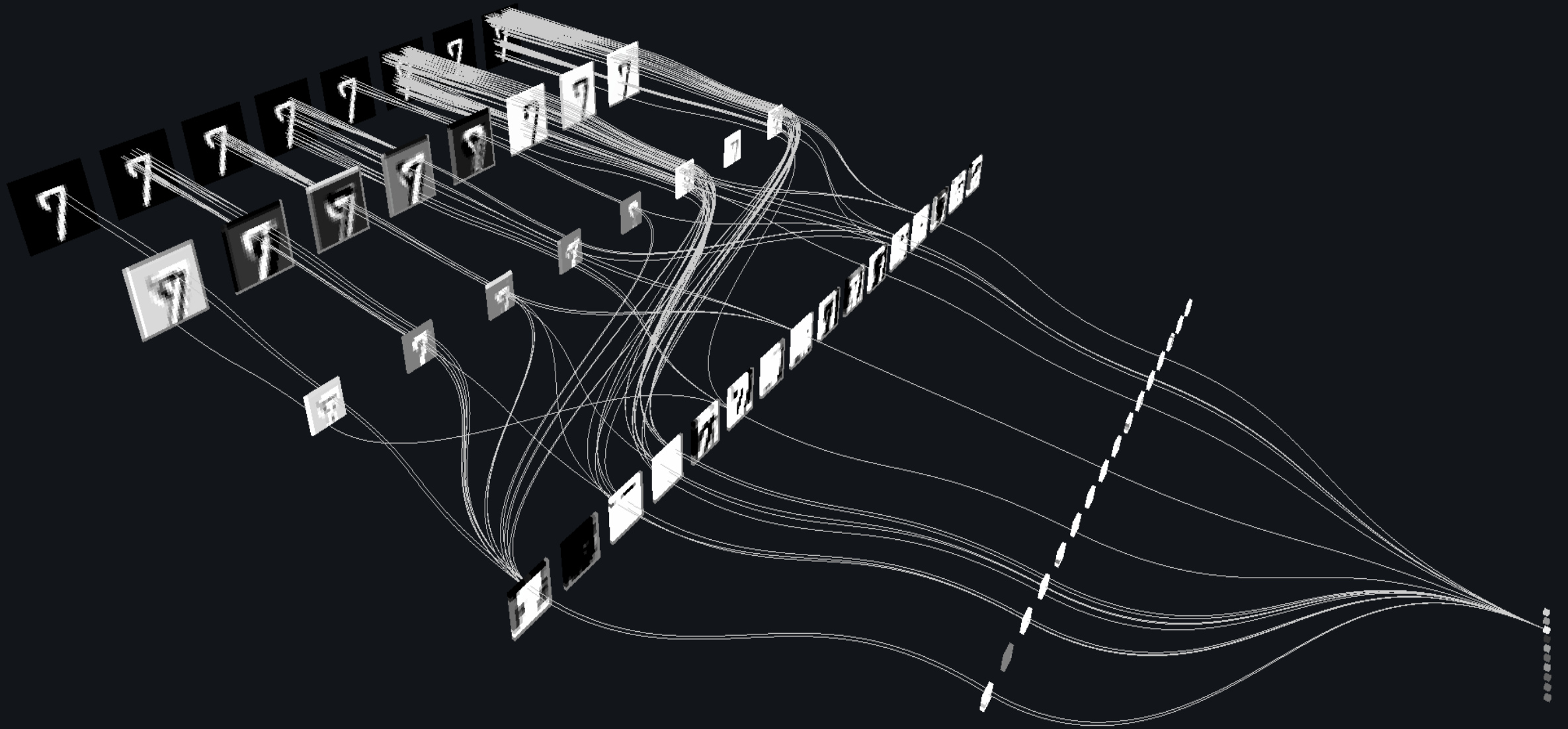
Md. Jalil Piran, PhD
Asst. Professor
Computer Science and Engineering
Sejong University
Spring, 2021

- Introduction to Deep Learning (DL)
- The History of DL
- Programming Tools
- Artificial Neural Networks (ANNs)
- Optimization in DL
- Convolutional Neural networks (CNNs)
- Unsupervised Pre-trained Networks (UPNs)

Outline



- Recurrent Neural Networks (RNNs)
- **Long Short-Term Memory (LSTM)**



ARCHITECTURE OF DEEP LEARNING

Some pre-RNN good results



This is a picture of one sky, one road and one sheep. The gray sky is over the gray road. The gray sheep is by the gray road.



Here we see one road, one sky and one bicycle. The road is near the blue sky, and near the colorful bicycle. The colorful bicycle is within the blue sky.



This is a picture of two dogs. The first dog is near the second furry dog.

Results with Recurrent Neural Networks



Man in black shirt is playing guitar.



Two young girls are playing with lego toy.



Construction worker in orange safety vest is working on road.



Boy is doing backflip on wakeboard.

Some pre-RNN bad results

Missed detections:



Here we see one potted plant.



This is a picture of one dog.

False detections:



There are one road and one cat. The furry road is in the furry cat.



This is a picture of one tree, one road and one person. The rusty tree is under the red road. The colorful person is near the rusty tree, and under the red road.

Incorrect attributes:



This is a photograph of two sheep and one grass. The first black sheep is by the green grass, and by the second black sheep. The second black sheep is by the green grass.



This is a photograph of two horses and one grass. The first feathered horse is within the green grass, and by the second feathered horse. The second feathered horse is within the green grass.

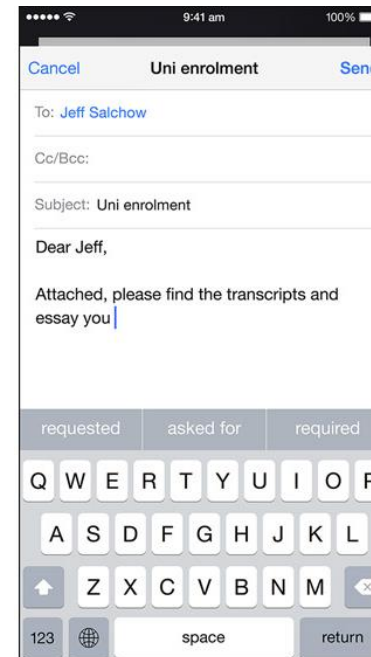
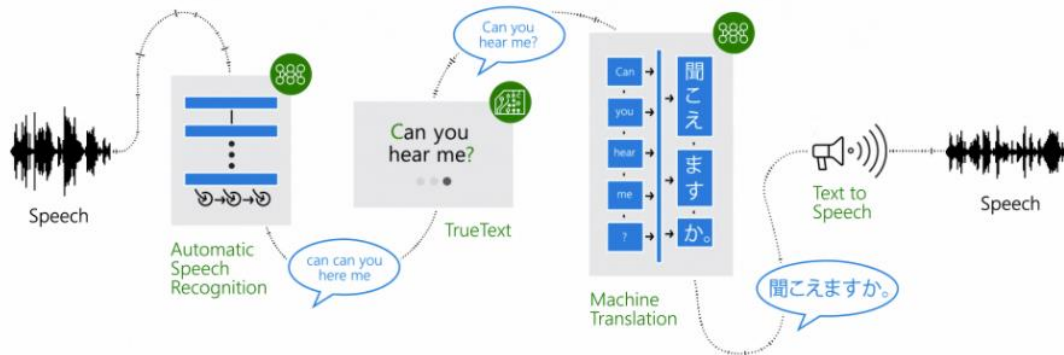
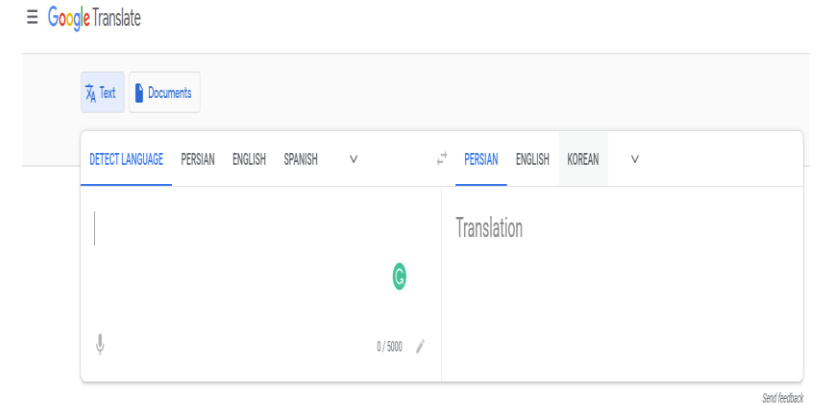
- Introduction
- Gradient vanishing
- Long Short-term Memory (LSTM)
- GRU.

- **Long short-term memory (LSTM) block/network**
 - A type of **RNNs**
 - A model for **short-term** memory, which can last for a **long** period of time.
 - LSTM cell '**remembers**' a value for some period of time, e.g. short or long
 - A solution for **exploding** and **vanishing gradient** problem.

Applications



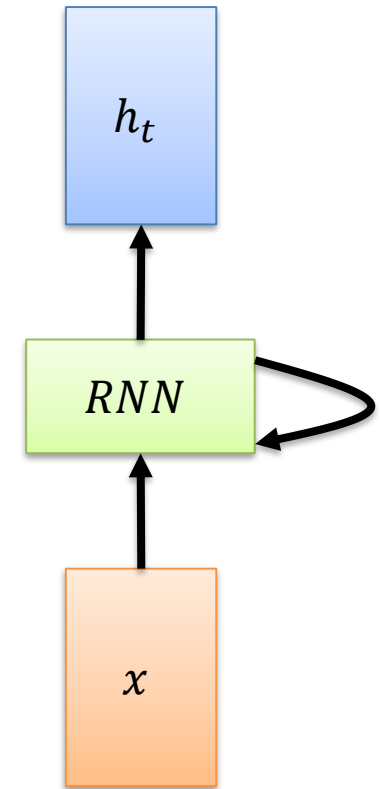
- Samsung Bixby
- Google Translator,
- Gmail text predictor
- Apple Quicktype
- Amazon Alexa
- Microsoft end-to-end translation



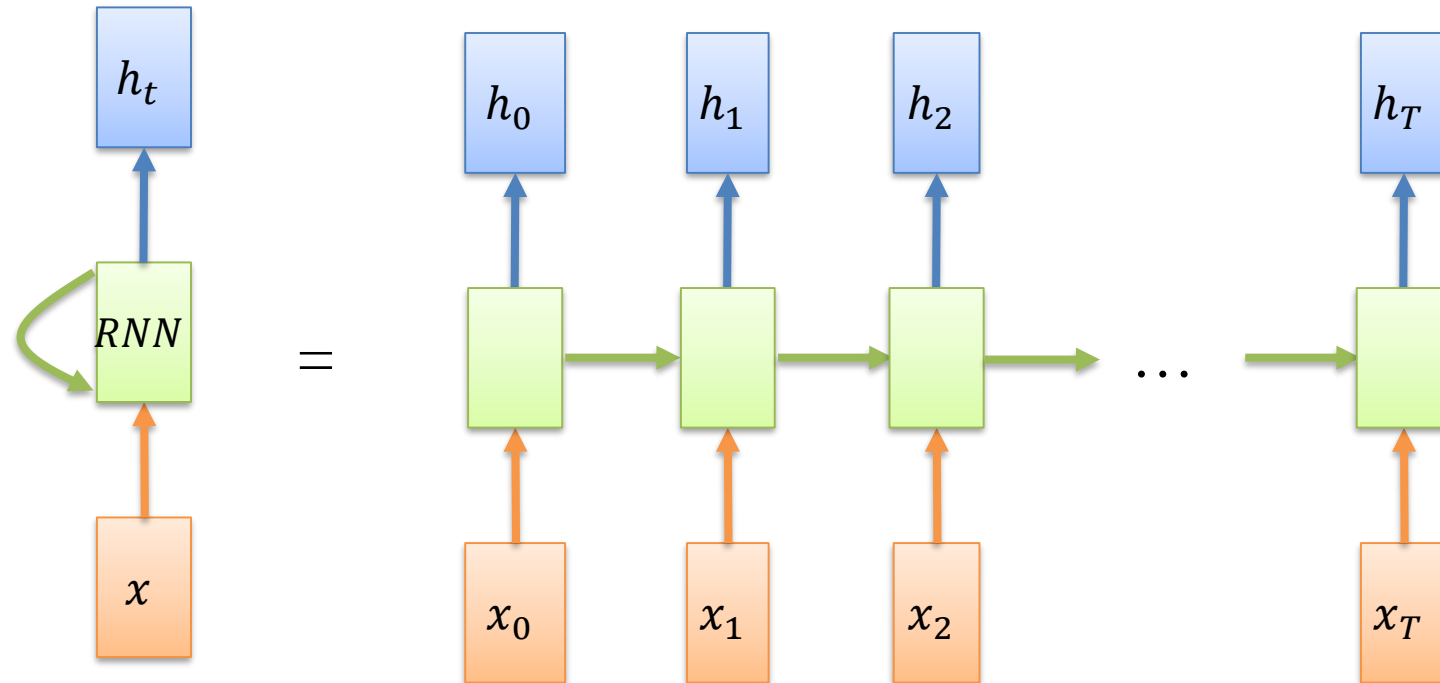
amazon echo dot



- **Long-term Dependencies**
 - Allows information to persist using the interaction between the layers
 - e.g. word prediction

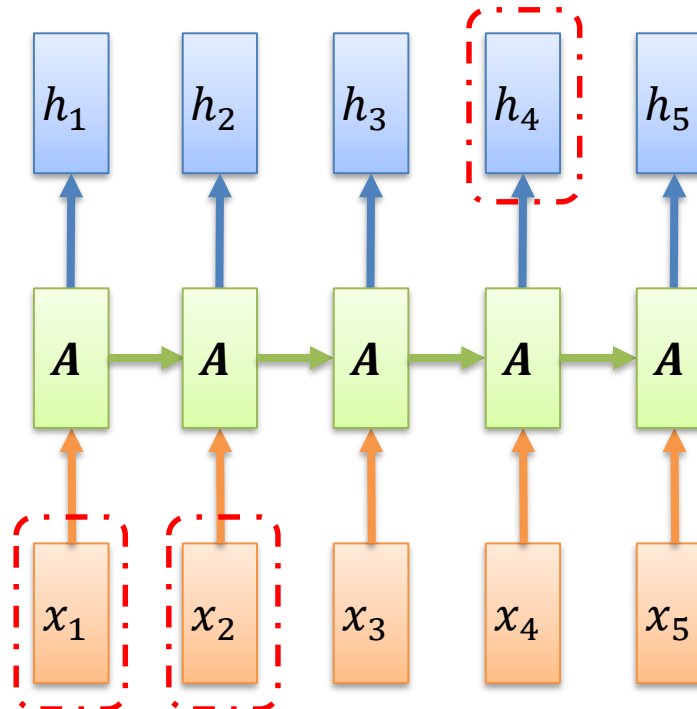


- **Long-term dependencies and gradient vanishin**
 - A RNN composed of multiple copies of the same network
 - A RNN can be unrolled as:



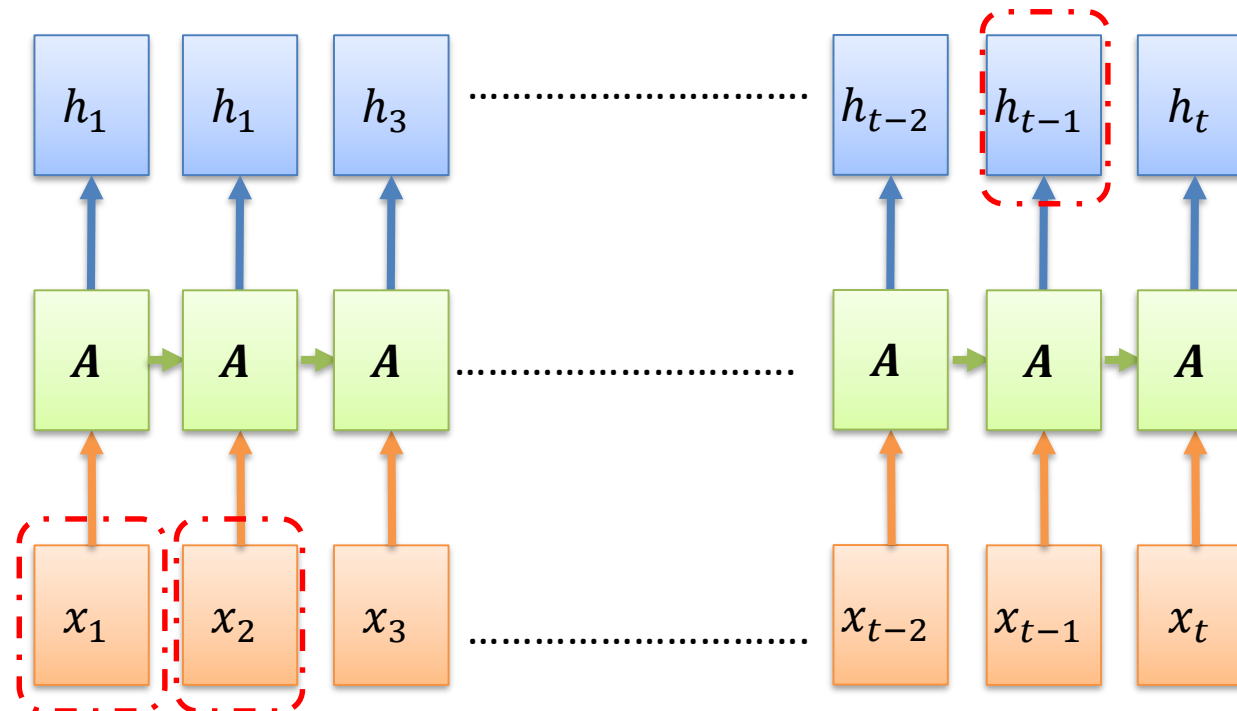
- **Long-term Dependencies**

- RNN connects retrieves previous information for the current states
 - Small gap between the relevant information and the place that it's needed
 - e.g. text prediction: 'the clouds are in the *sky*.'



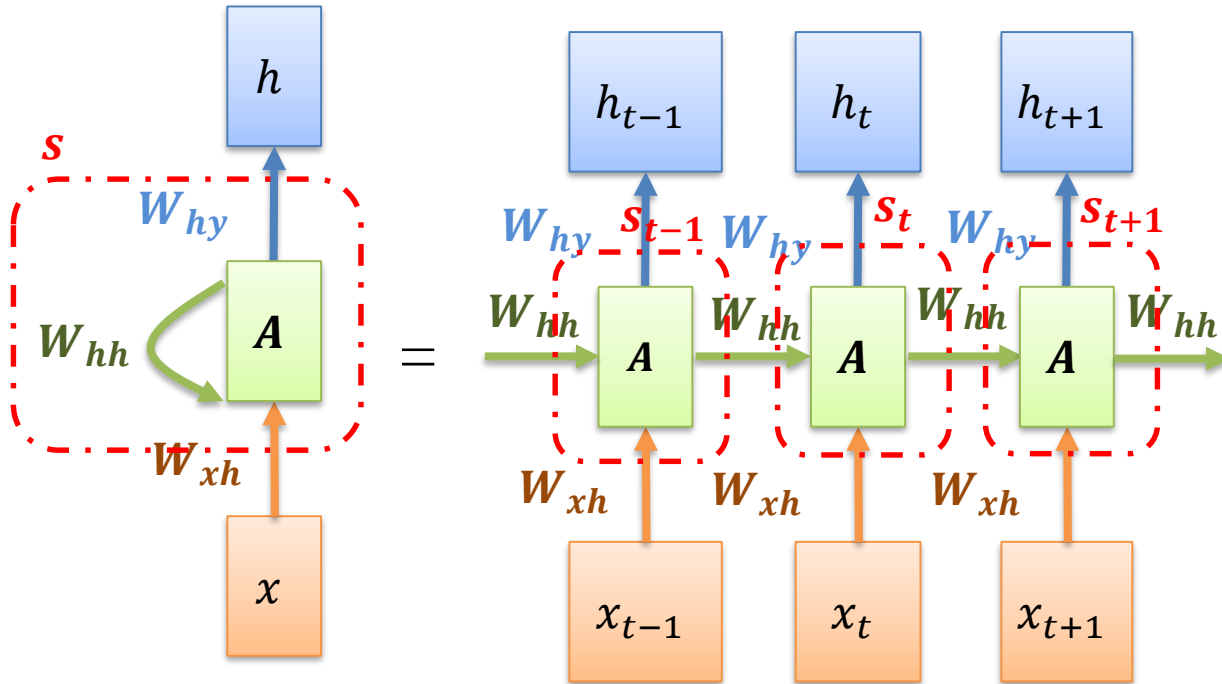
- **Long-term Dependencies**

- RNN connects retrieves previous information for the current states
 - Large gap between the relevant information and the place that it's needed
 - e.g. text prediction: 'My bachelor degree is Computer science.. My job is *programming*.'



- **Long-term Dependencies**
 - In theory, RNNs carefully picks up the parameters and seems to be able to connect information in a large gap.
 - In practice, RNNs are not able to learn in large gaps.
 - **Solution:** LSTM

- Backpropagation through time (BPTT)



Suppose:

- $W_{xh} = U$
- $W_{xh} = W$
- $W_{hy} = V$
- $h = s$
- $Loss = E$

$$s_t = \tanh(Ux_t + Ws_{t-1})$$

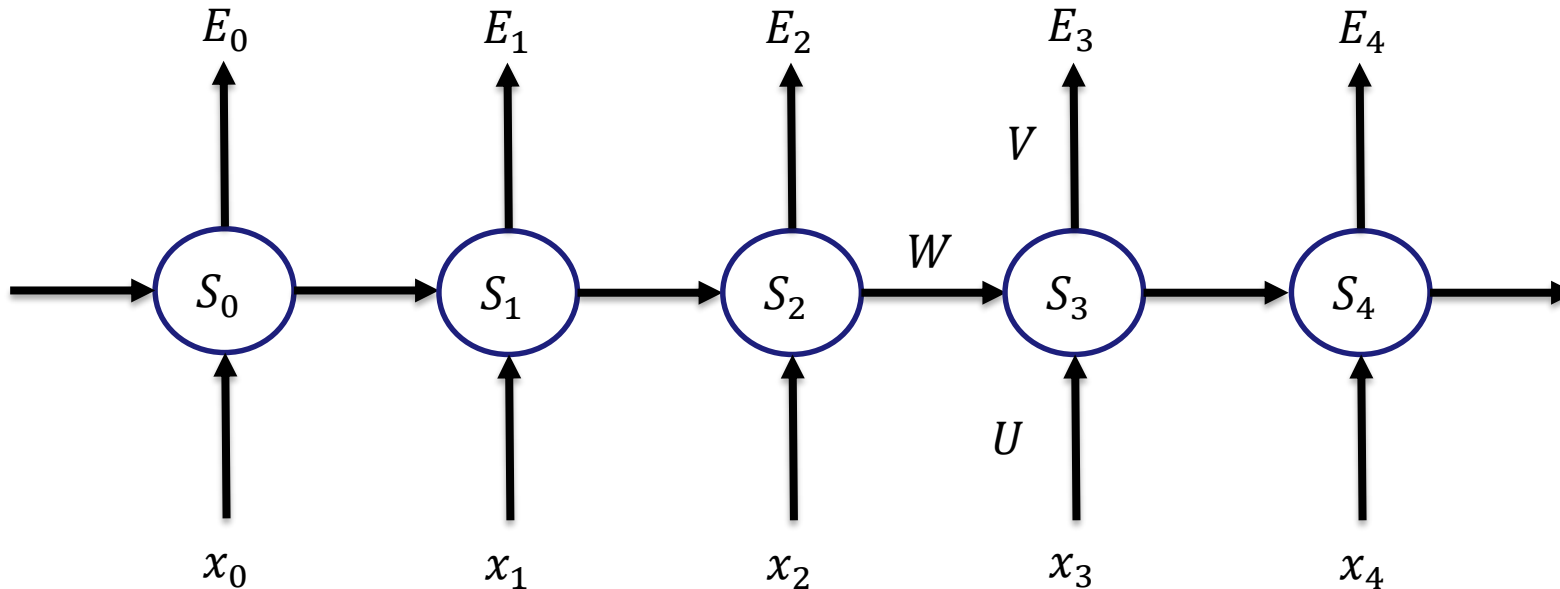
$$\hat{y}_t = \text{softmax}(Vs_t)$$

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$

$$E(y, \hat{y}_t) = \sum_t E_t(y_t, \hat{y}_t)$$

$$= - \sum_t y_t \log \hat{y}_t$$

- **Example:**
 - Full sequence (sentence) as one training example
 - The total error is just the same as the errors at each time step (word)
 - $\frac{\partial E_3}{\partial V}$ only depends on the values at the current time step.



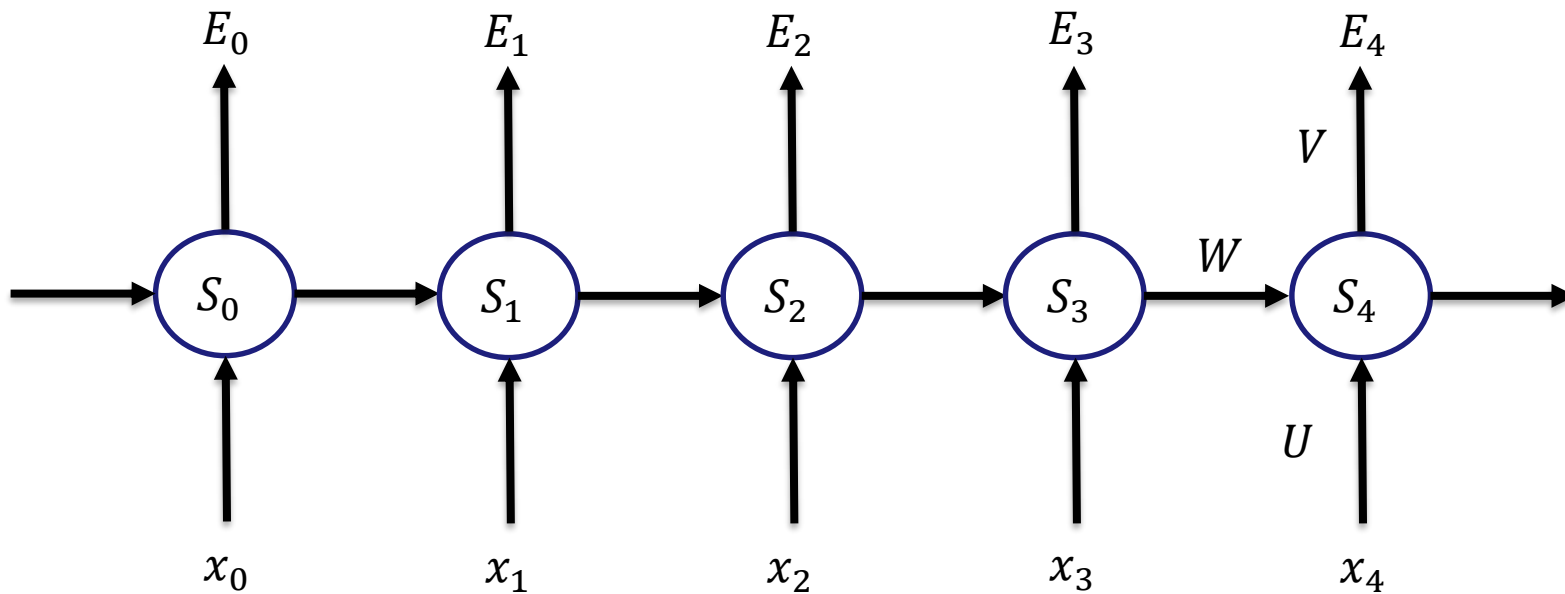
$$\frac{\partial E_3}{\partial V} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial V}$$
$$= \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial z_3} \frac{\partial z_3}{\partial V}$$

$$z_3 = V s_3$$

Gradient Vanishing



- The story is different for $\frac{\partial E_3}{\partial W}$ and $\frac{\partial E_3}{\partial U}$
- s_3 depends on s_2 , which depends on W and s_1 and so on.



$$\frac{\partial E_3}{\partial W} = \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial W}$$

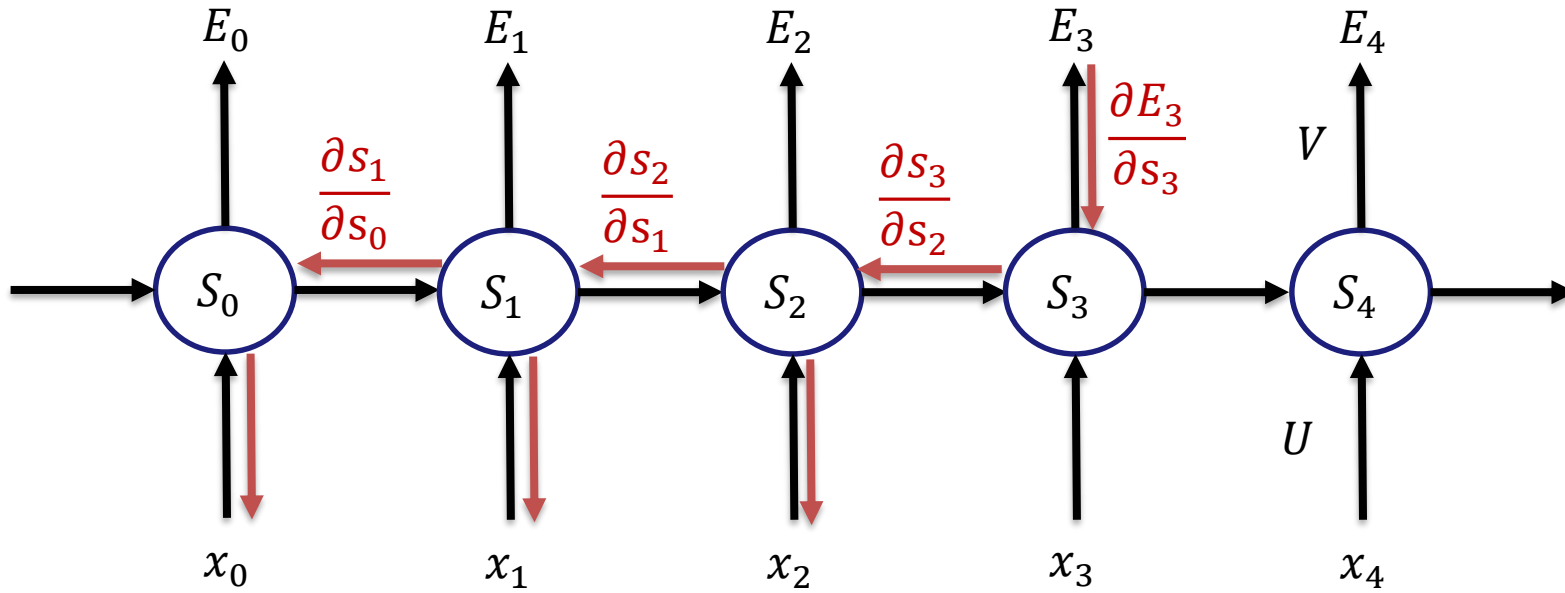
$$s_3 = \tanh(Ux_t + Ws_2)$$

Gradient Vanishing



- We need to apply the chain rule again
 - This is exactly the same as the standard backpropagation algorithm
 - The key difference is parameter sharing

$$\frac{\partial E_3}{\partial W} = \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W}$$



Note: $\frac{\partial s_3}{\partial s_1}$ is a chain rule itself.

$$\frac{\partial s_3}{\partial s_1} = \frac{\partial s_3}{\partial s_2} \frac{\partial s_2}{\partial s_1}$$

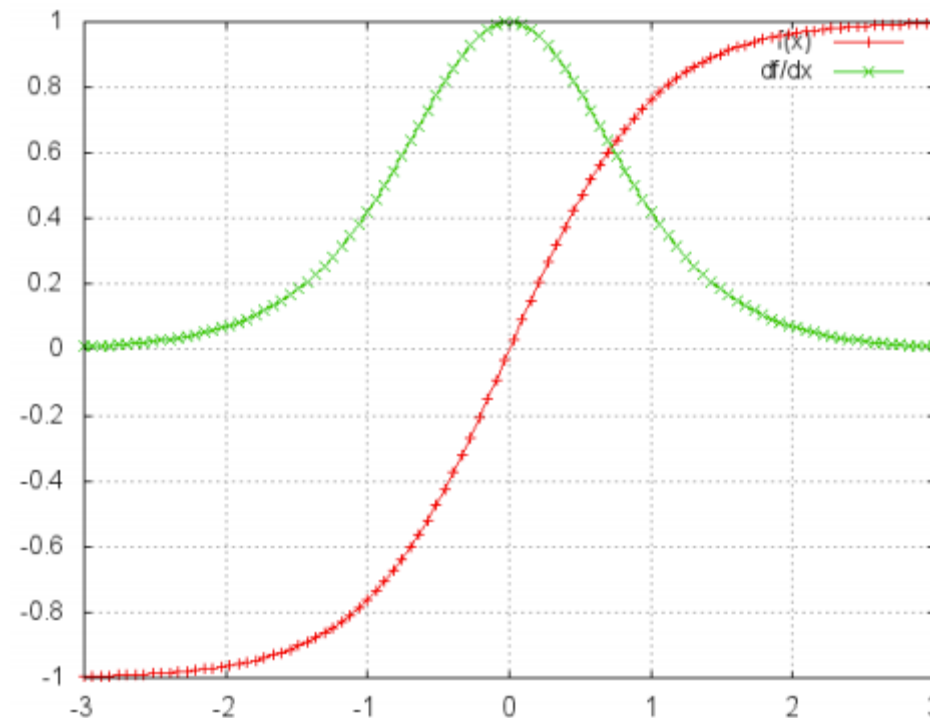
Gradient Vanishing



- Gradient

$$\begin{aligned}\frac{\partial E_3}{\partial W} &= \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \frac{\partial s_3}{\partial s_k} \frac{\partial s_k}{\partial W} \\ &= \sum_{k=0}^3 \frac{\partial E_3}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial s_3} \left(\prod_{j=k+1}^3 \frac{\partial s_j}{\partial s_{j-1}} \right) \frac{\partial s_k}{\partial W}\end{aligned}$$

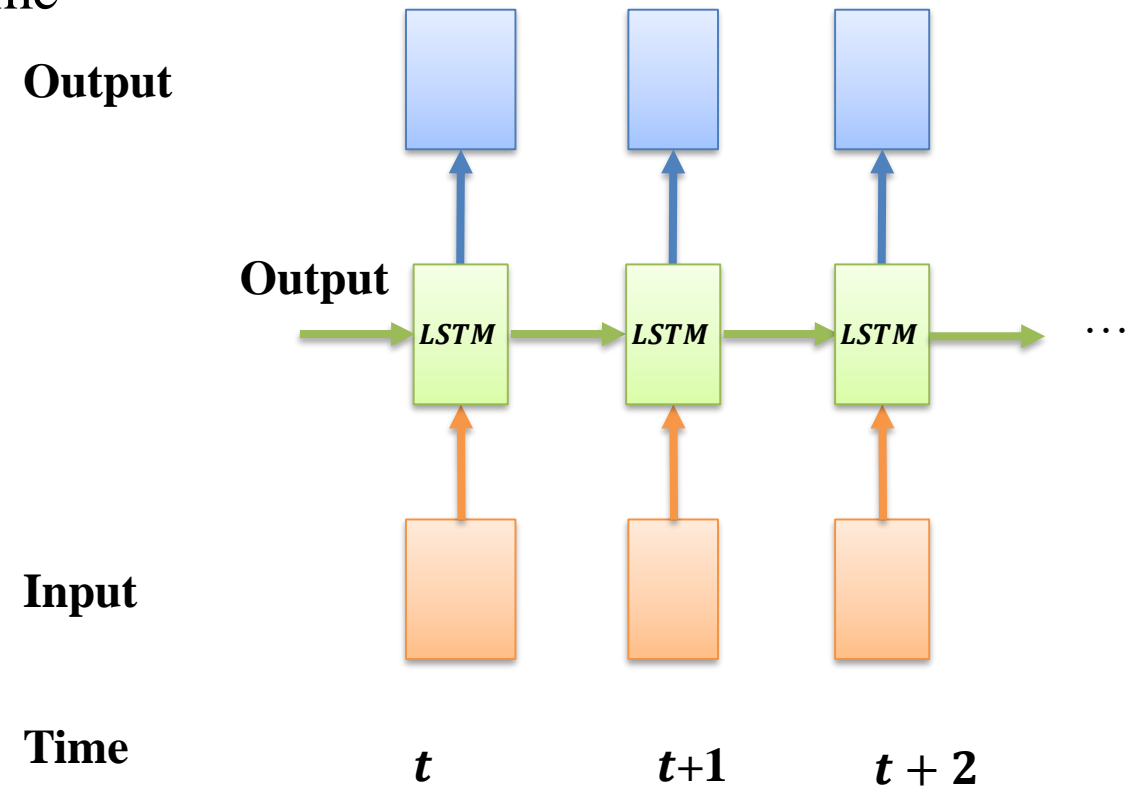
$$s_t = \tanh(Ux_t + Ws_{t-1})$$



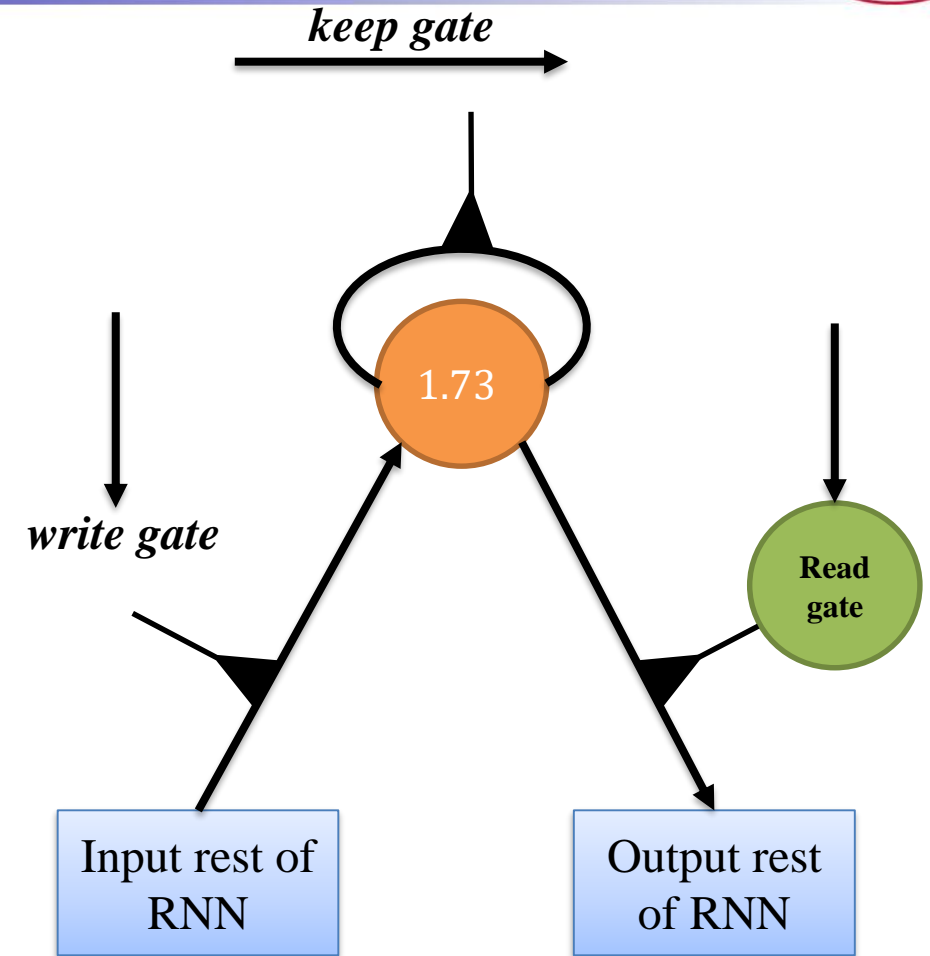
$$\frac{\partial \tanh(x)}{\partial x} = 1 - \tanh^2(x)$$

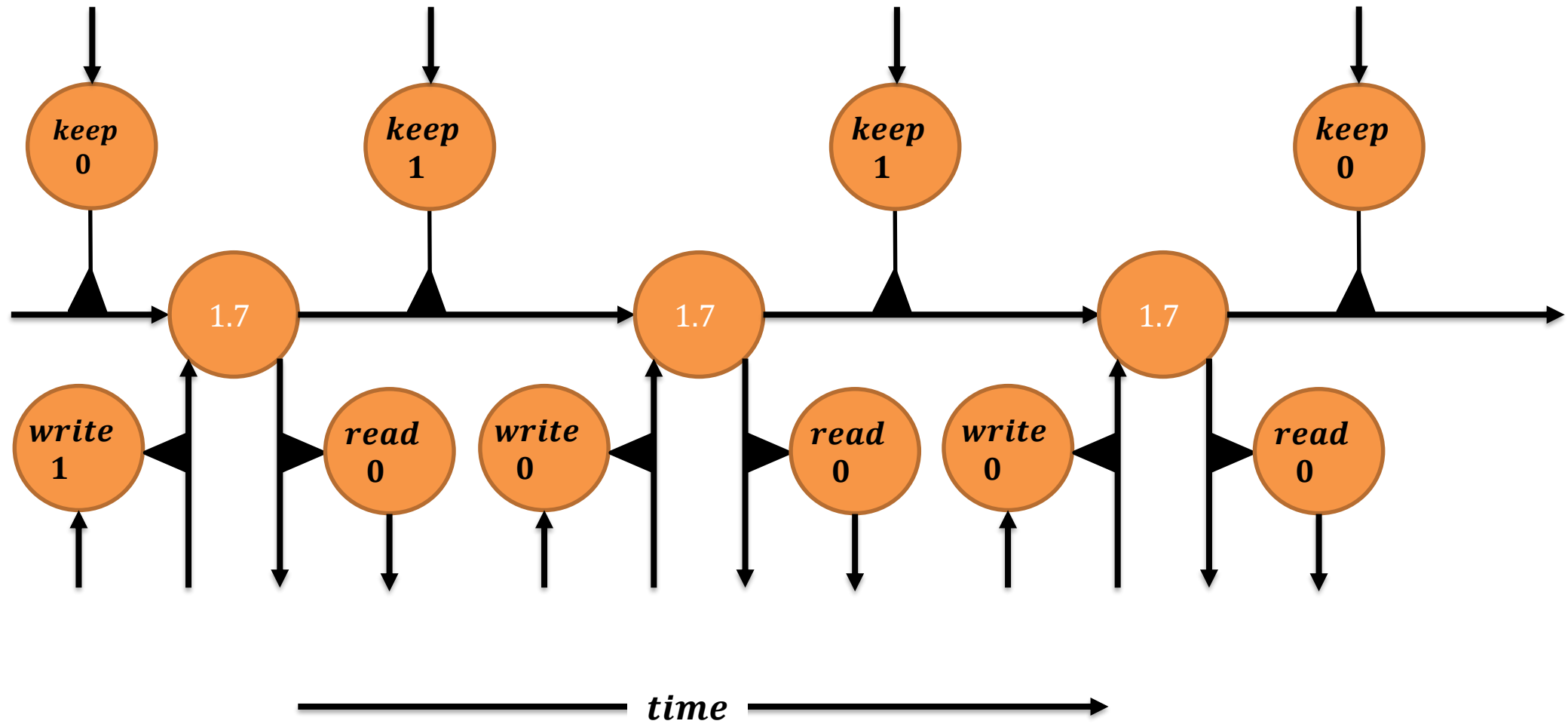
- Computing gradient of s_0 involves many factors of W (and repeated \tanh)
- Exploding gradients
 - **Gradient clipping**: scale gradient if its norm is too big
- Vanishing gradient:
 - Proper **initialization** of W
 - Using **ReLU** instead of \tanh or sigmoid
 - More popular solution is to use **LSTM** or **Gated Recurrent Unit (GRU)**

- A special type of RNN
- Introduced by S. Hochriter and J. Schmidhuber in 1997:
<http://www.bioinf.jku.at/publications/older/2604.pdf>
- Learn **long-term dependencies**
 - e.g. remember information for a long period of time

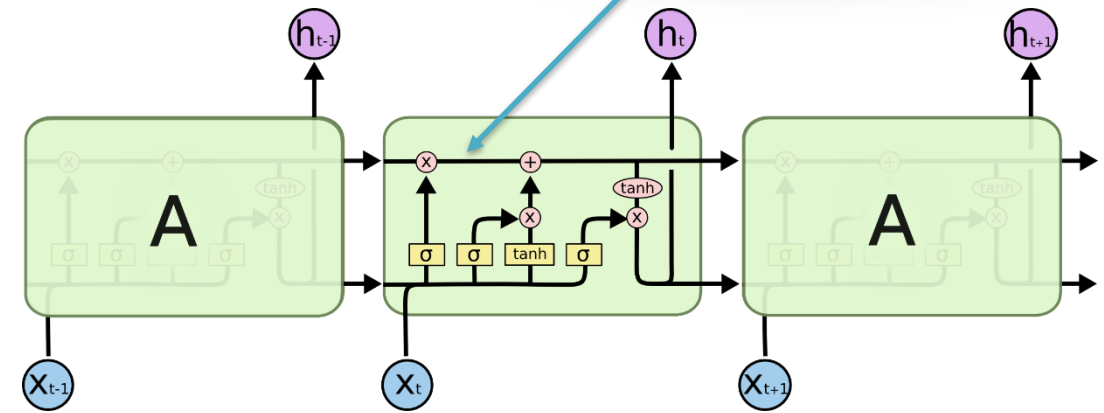
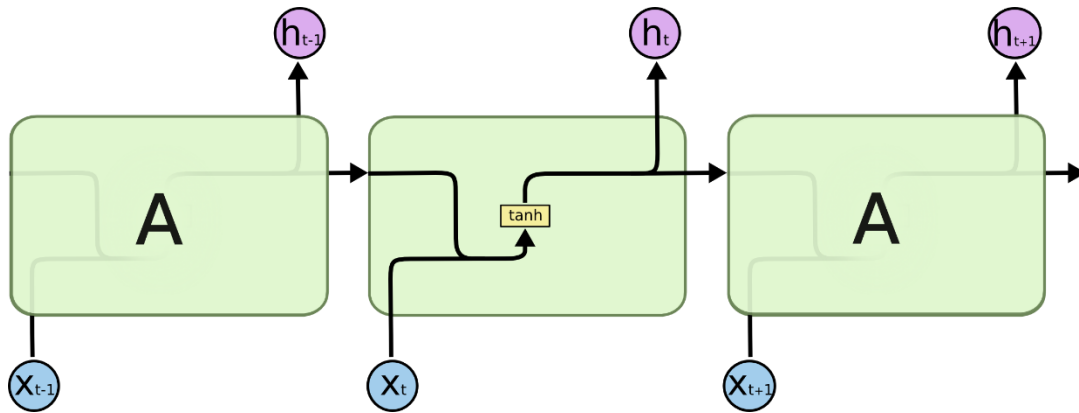


- To preserve information for a long time in the activities of an RNN,
- Use a circuit that implements an analog memory cell.
- A linear unit with a self-link and a weight of 1 will maintain its state.
- Information is stored in the cell by activating its write gate.
- Information is retrieved by activating the read gate.








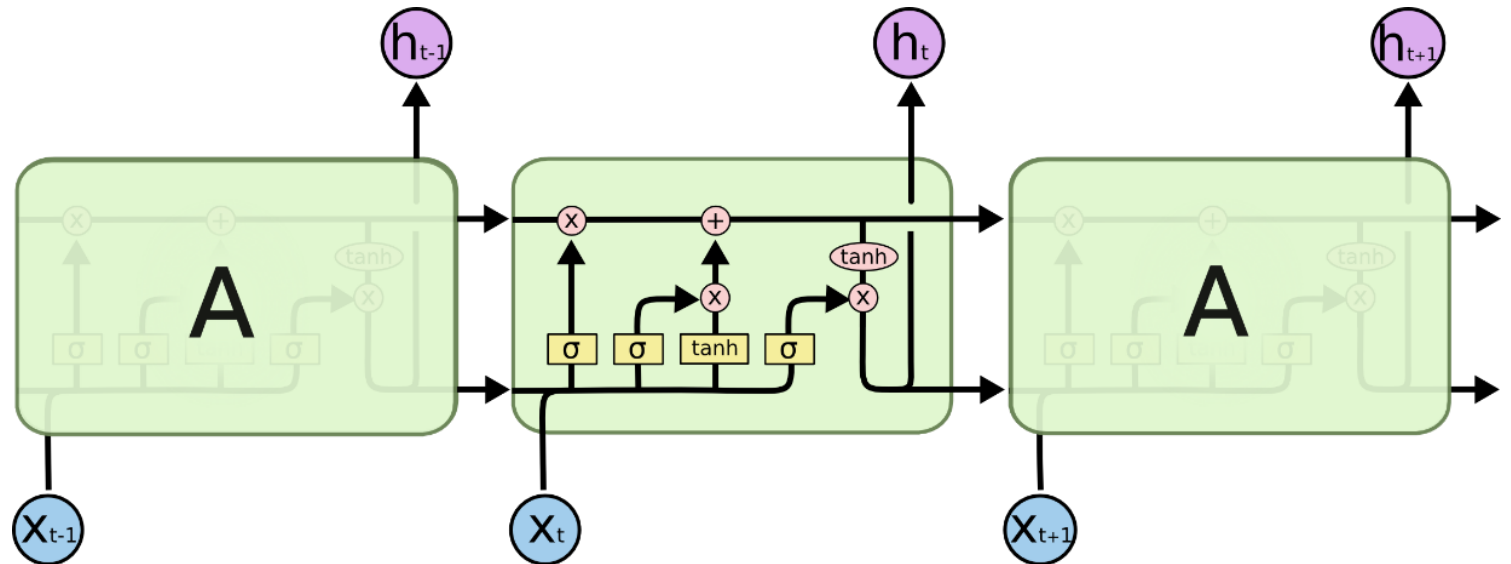


- LSTM vs. RNN
 - RNN: each module contains a single layer
 - LSTM: each module contains some new functionalities.

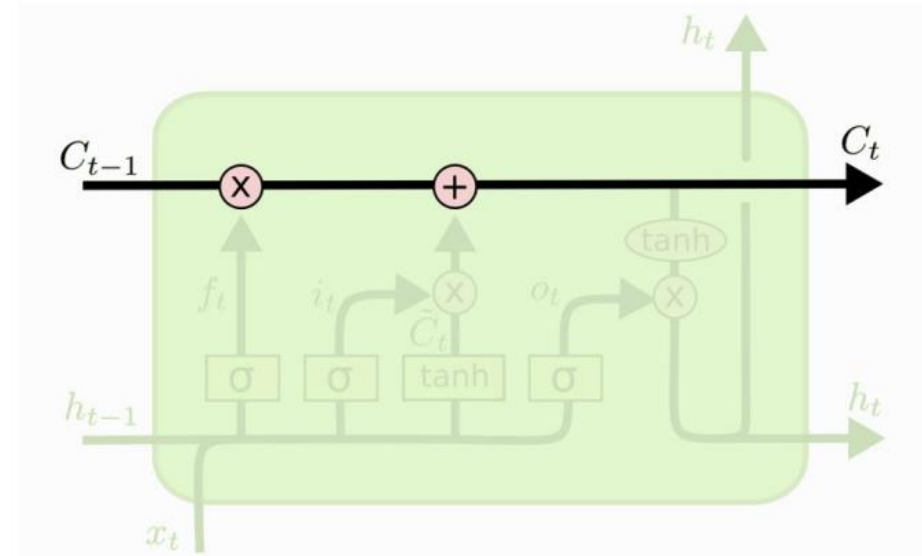


- Symbols

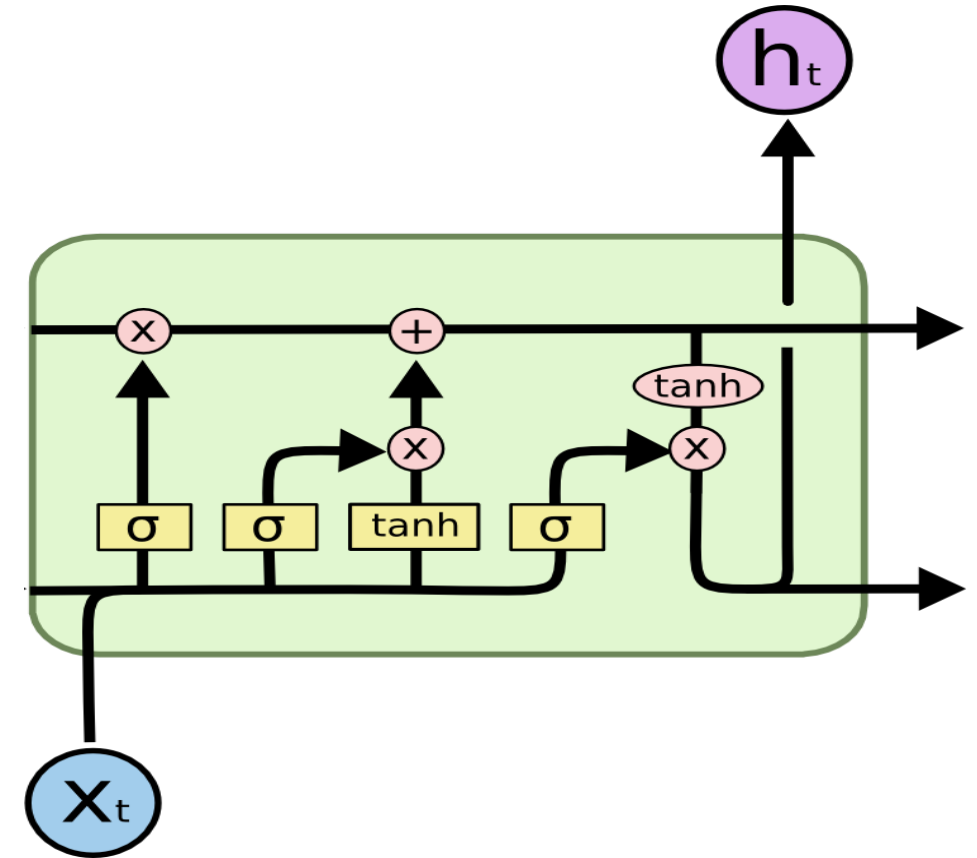
- Neural network layer 
- Pointwise operation 
- Vector transfer 
- Concatenate 
- Copy 



- **Cell state:** the novel functionality in LSTM
 - Like a conveyor belt
 - Runs straight down the entire chain
 - With only some minor linear interactions.
 - Can remove or add information to the cell state



- Components
 - **Hidden states**
 - Carry previous information
 - Build a direct path
 - **Gates**
 - A neural network σ
 - Output range $[0, 1]$
 - Pointwise multiplication
 - Filtering function
 - Input gate, forget gate, output gate

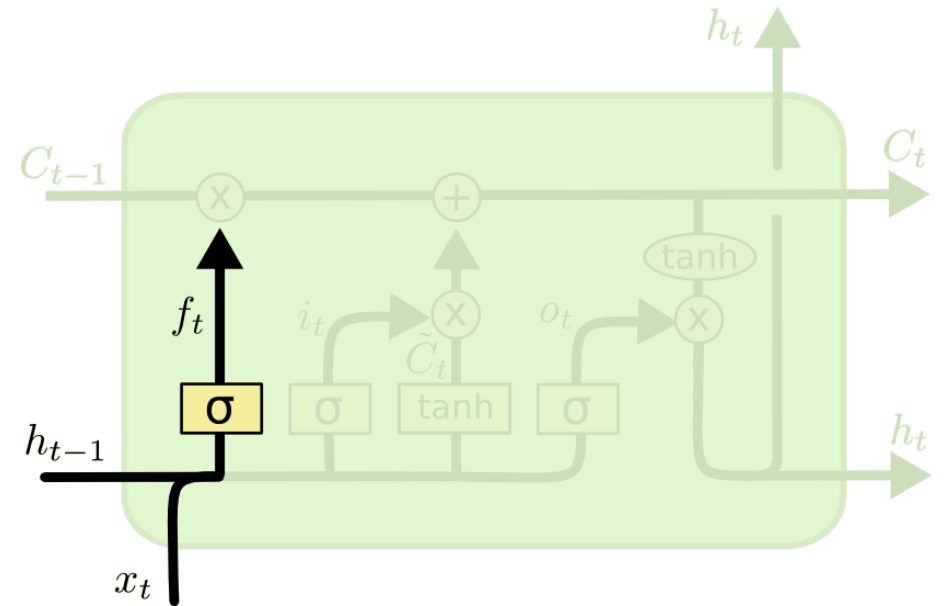


Gates

- **Forget Gate**

- A way to optionally let information pass
- Compose of a sigmoid neural networks layer and a pointwise multiplication operation.
- Sigmoid layer:
 - 0: *'let nothing pass'*
 - 1: *'let everything pass'*

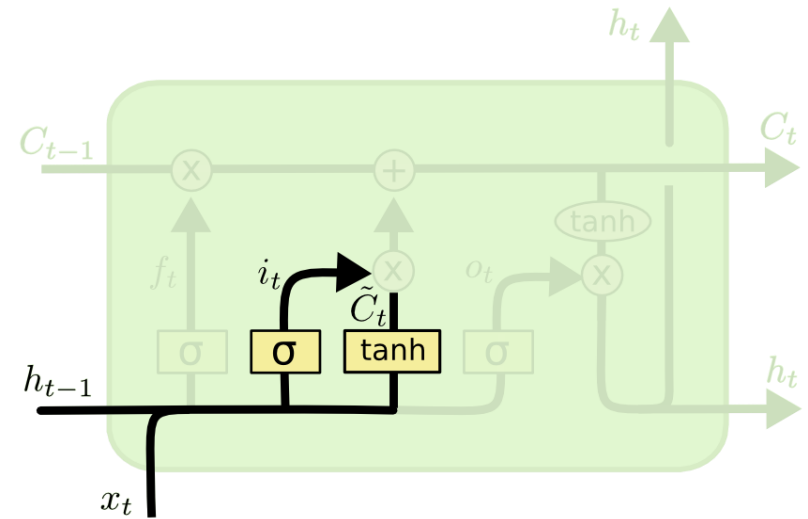
$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$



- **Input Gate**
 - Add new information
 - Decide what new information store in the cell state
 - Input gate layer (σ)
 - Decides which values to update
 - *tanh* layer
 - Creates a vector of new candidate values, \tilde{C}_t

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

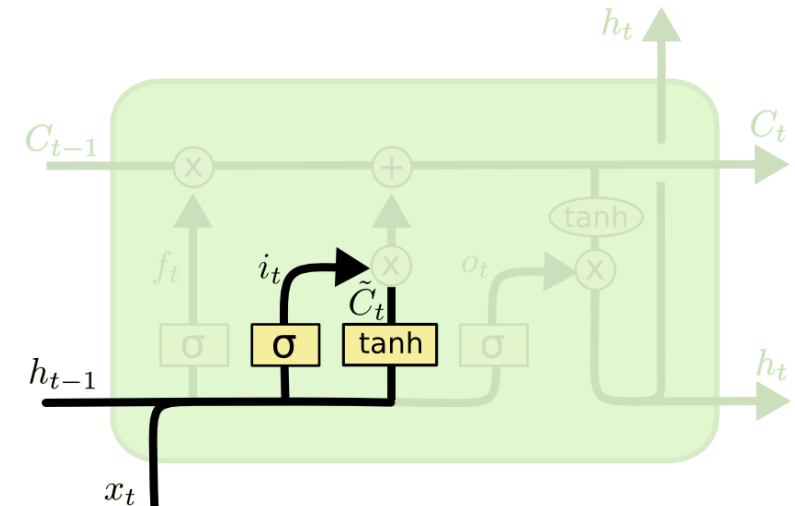
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$



- Example
 - Predict the next word based on all the previous words
 - The cell state might include the degree and major of the present subject
 - By arriving new subject: add the major or degree of the new subject to the cell state

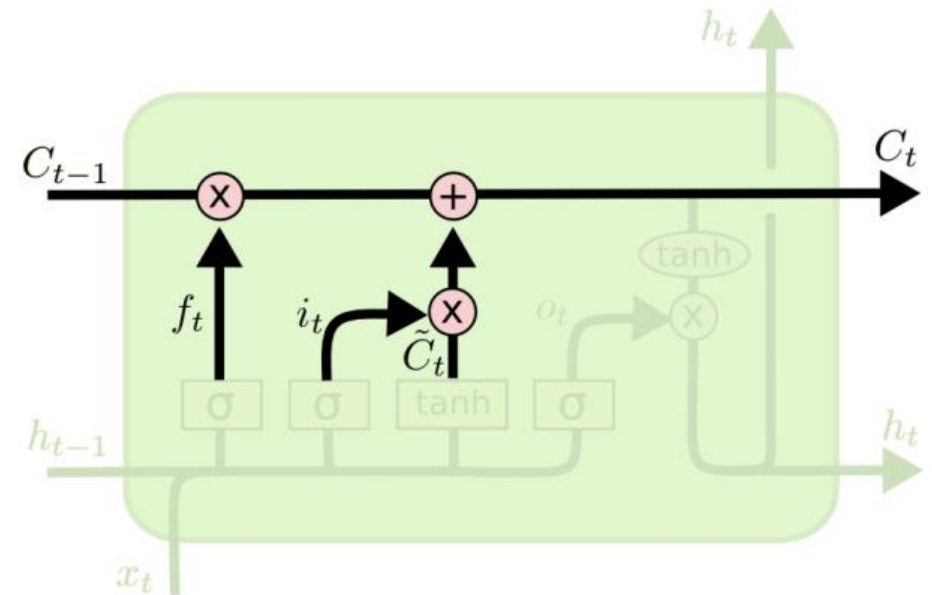
$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

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



- Update cell state
 - Discard unwanted information: $f_t * C_{t-1}$
 - Add the desired information: $i_t * \tilde{C}_t$

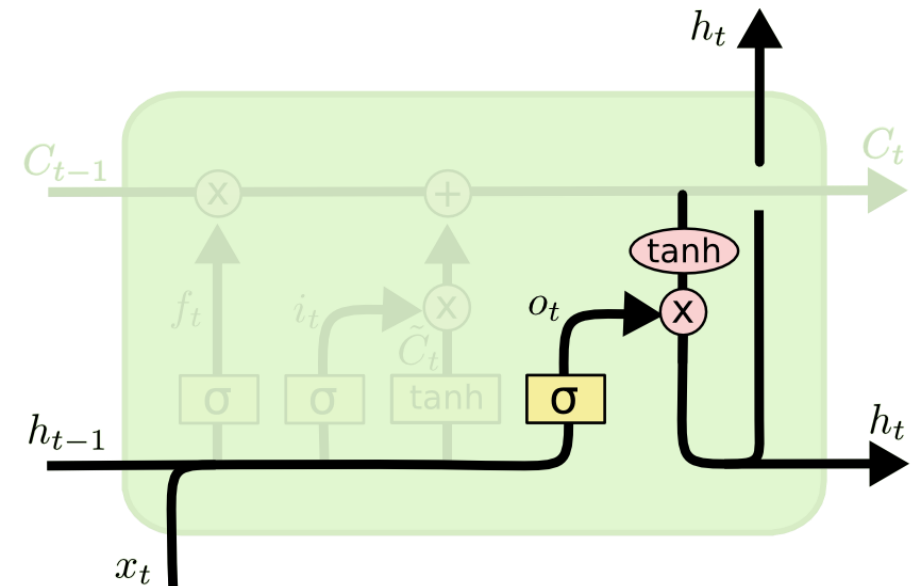
$$\tilde{C}_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



- **Output Gate**
 - Decides what parts of the cell state as output
 - *tanh* layer
 - Push the values to be between -1 and +1

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

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



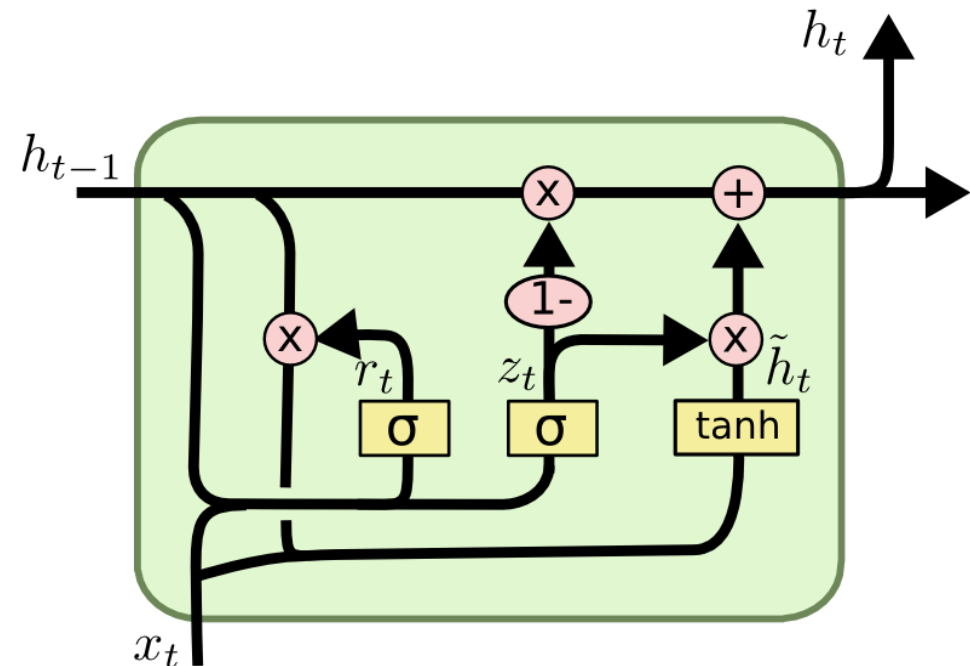
- **Gated recurrent unit (GRU)**
 - Combines the forget and input gates into a single *updated gate*
 - Also merges the cell state and hidden state

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

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

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

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



Conclusion



- RNN is used for sequence / time-series problems
- Gradient vanishing problem limit its effectiveness
- LSTM cell to solve the problem
- Practically LSTM works much better than RNN

Image Caption Generation

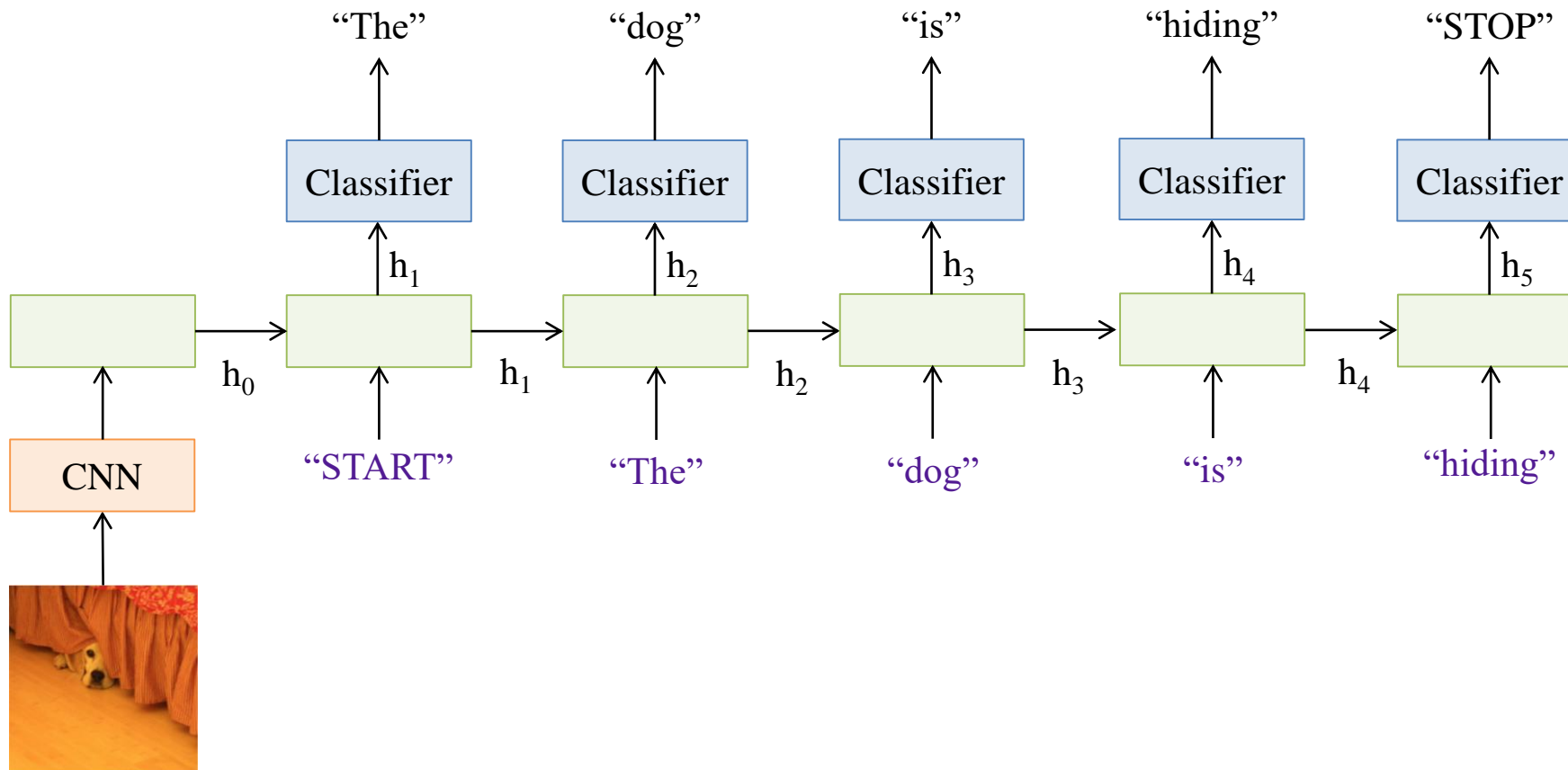


Image Caption Generation

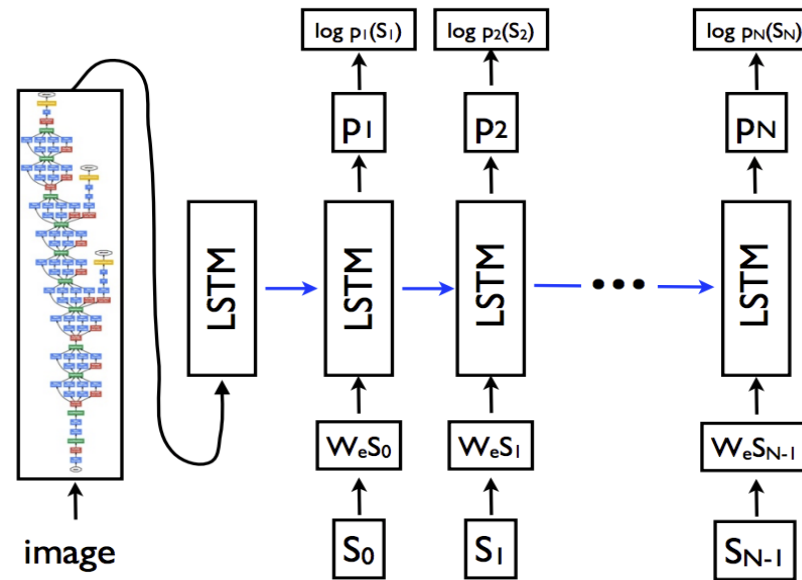
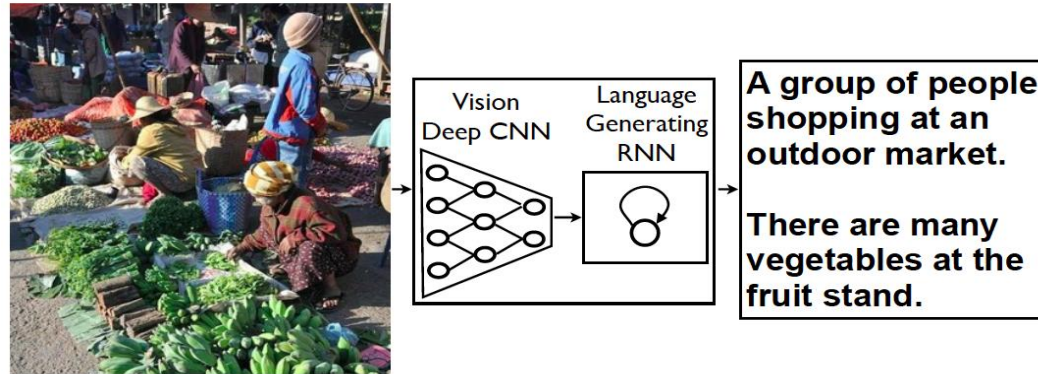


Image Caption Generation

A person riding a motorcycle on a dirt road.



Two dogs play in the grass.



A skateboarder does a trick on a ramp.



A dog is jumping to catch a frisbee.



A group of young people playing a game of frisbee.



Two hockey players are fighting over the puck.



A little girl in a pink hat is blowing bubbles.



A refrigerator filled with lots of food and drinks.



A herd of elephants walking across a dry grass field.



A close up of a cat laying on a couch.



A red motorcycle parked on the side of the road.



A yellow school bus parked in a parking lot.



Describes without errors

Describes with minor errors

Somewhat related to the image

Unrelated to the image

- R. Pascanu, T. Mikolov, and Y. Bengio, [On the difficulty of training recurrent neural networks](#), ICML 2013
- S. Hochreiter, and J. Schmidhuber, [Long short-term memory](#), Neural computation, 1997 9(8), pp.1735-1780
- F.A. Gers, and J. Schmidhuber, [Recurrent nets that time and count](#), IJCNN 2000
- K. Greff , R.K. Srivastava, J. Koutník, B.R. Steunebrink, and J. Schmidhuber, [LSTM: A search space odyssey](#), IEEE transactions on neural networks and learning systems, 2016
- K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, [Learning phrase representations using RNN encoder-decoder for statistical machine translation](#), ACL 2014
- R. Jozefowicz, W. Zaremba, and I. Sutskever, [An empirical exploration of recurrent network architectures](#), JMLR 2015

