# Recurrent Neural Network for Sequence Processing

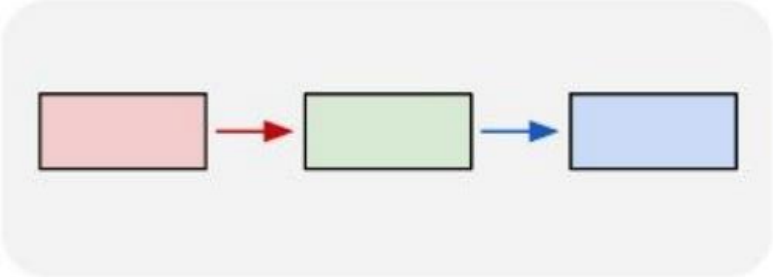
CS 281b

02/28/2018

**Da Zhang**

# So far: static images

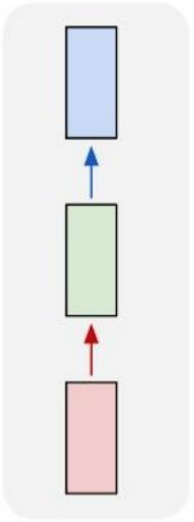
- Classification, localization, detection



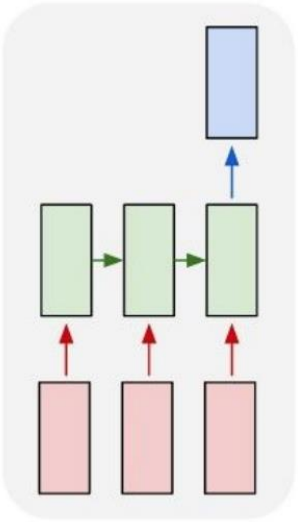

one to one

# Sequence processing

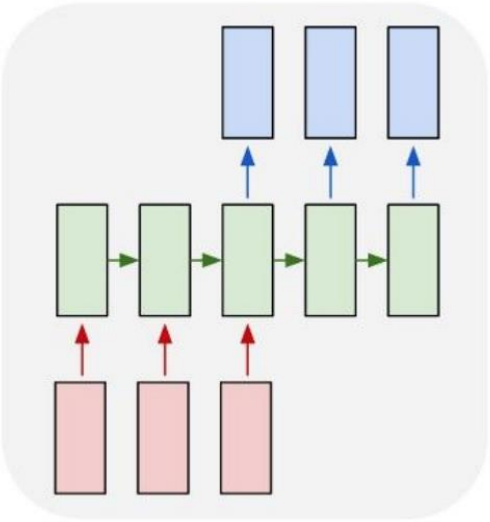
one to one



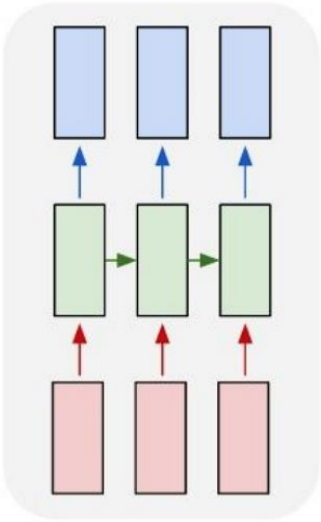
many to one



many to many



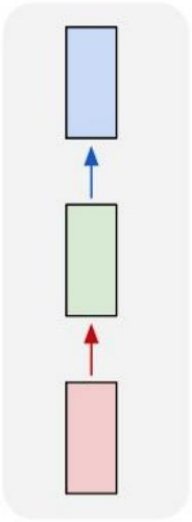
many to many



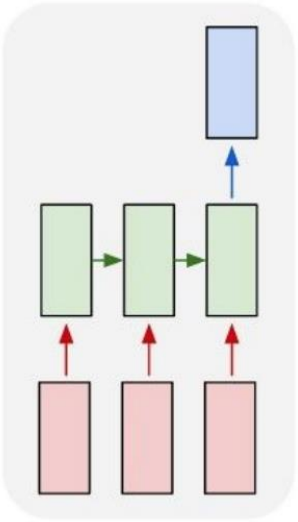
Sentiment analysis
Video classification

# Sequence processing

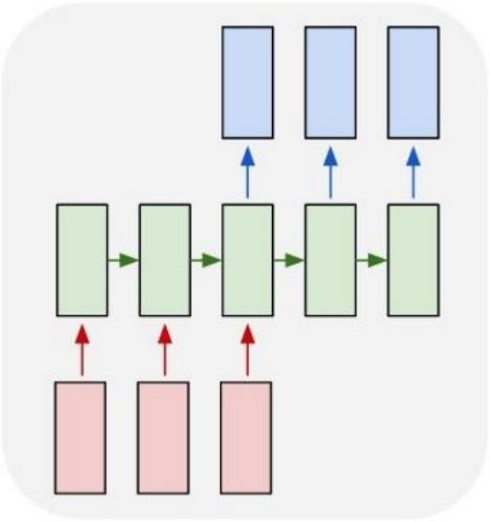
one to one



many to one

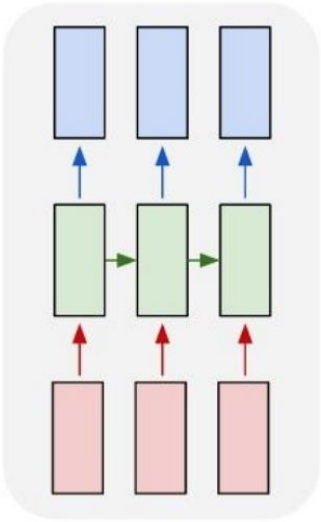


many to many



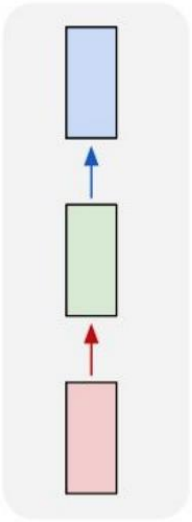
Machine translation

many to many

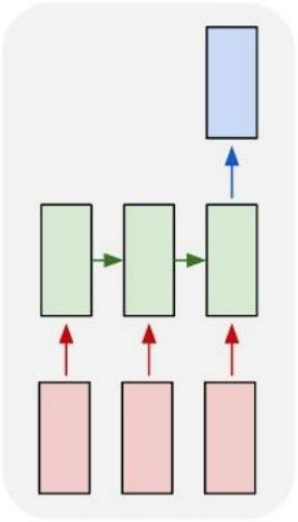


# Sequence processing

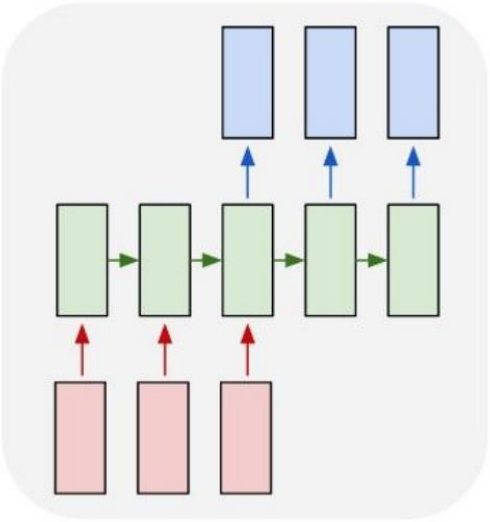
one to one



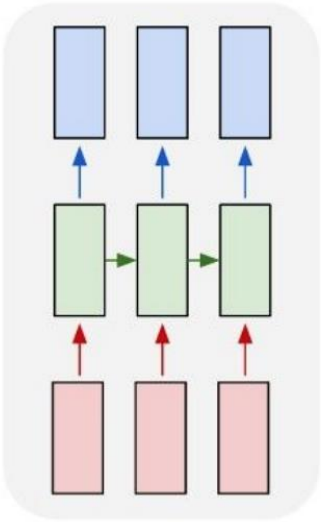
many to one



many to many



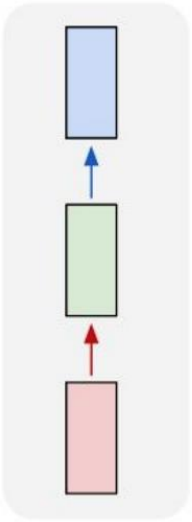
many to many



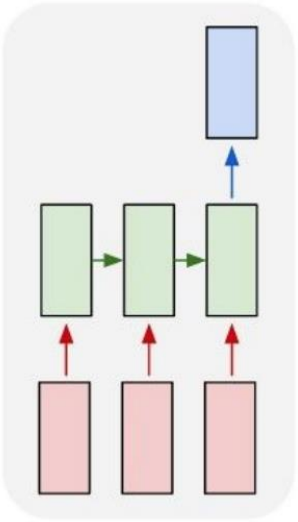
Frame-level prediction

# Sequence processing

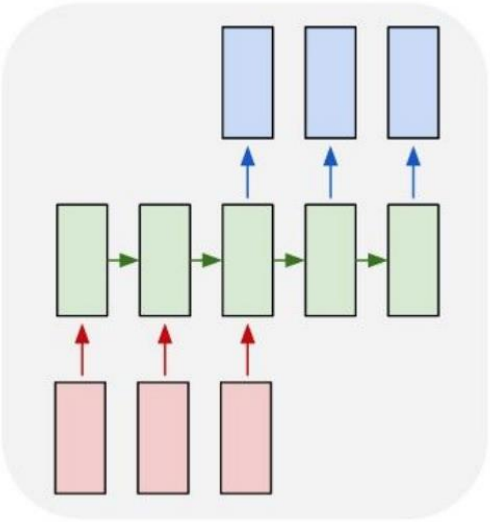
one to one



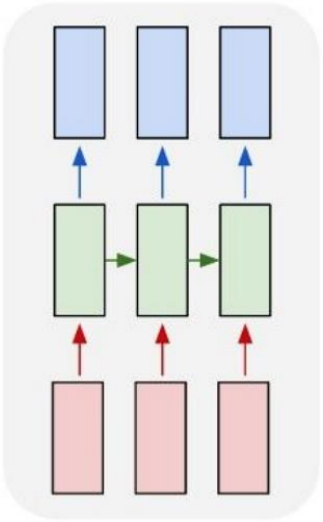
many to one



many to many



many to many



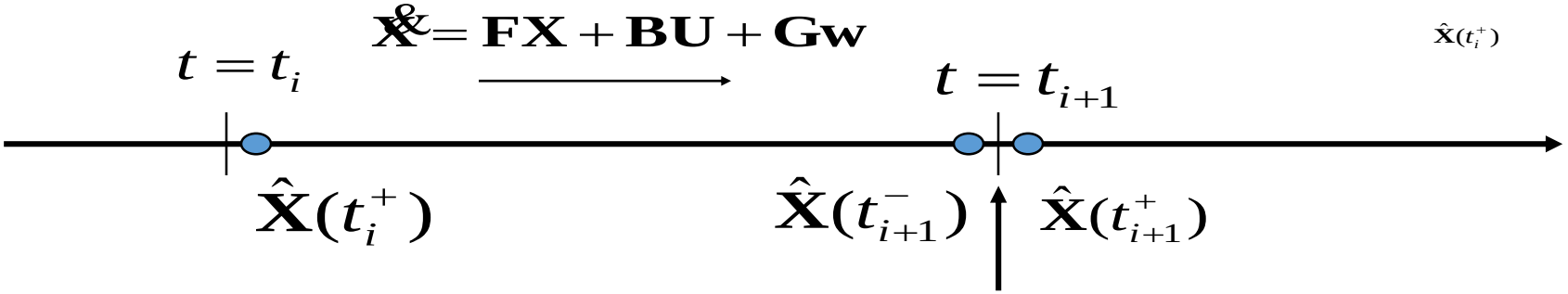
Recurrent neural network: very useful to model sequential data such as video and language.

# Artificial Example

- State: {Hungry, Full, Sleepy, Relieving}
- Action: {Go to Kitchen, Lie down on pillow, Go to potty, Being playful}
- Internal states are never observed directly
- Track internal states over time
- Two important aspects for state propagation:
  - Internal state transition (a transition or propagation equation)
  - Internal state x observed action (a state prediction equation)

# State Transition Equation

- State: {Hungry, Full, Sleepy, Relieving}
- Full -> sleeping
- Full -> relieving
- Sleepy -> relieving
- Sleepy -> hungry, ….



# State Prediction Equation

- State: {Hungry, Full, Sleepy, Relieving}
- Action: {Go to kitchen, Lie down on pillow, Go to potty, Being playful}
- Hungry x Go to kitchen -> Full
- Sleepy x Lie down on pillow -> Relieving
- Full x being playful -> Hungry
- Etc.

$$\mathbf{X}_i = \mathbf{X}_{i-1}^{+} + \mathbf{K}_i (\mathbf{Z}_i - \mathbf{H}_i \mathbf{X}_i^{-})$$

# Kalman Filter

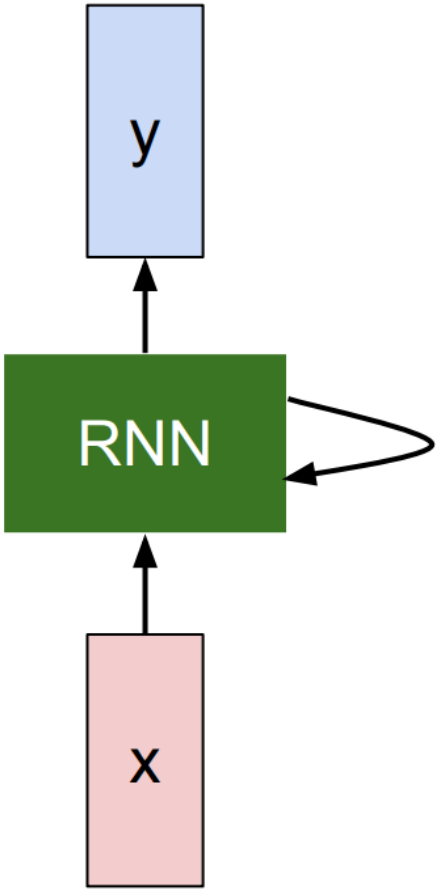
- The best linear prediction mechanism
- Assuming
  - Transition and prediction equations are linear
  - All components known
  - All noise processes are white and Gaussian
  - Etc. etc.

# RNN

- A glorified Kalman Filter
- Don’t know
  - State descriptions
  - Transition and prediction mechanisms
- Do assume
  - The same equation and parameters apply everywhere
- Learn everything from data!
- For learning to be possible, some fancy re-formulation is important

# Recurrent Neural Network

Recurrent neural network: Processing a sequence of vectors by applying a recurrence formula at every time step.



# Recurrent Neural Network

Recurrent neural network: Processing a sequence of vectors by applying a recurrence formula at every time step.

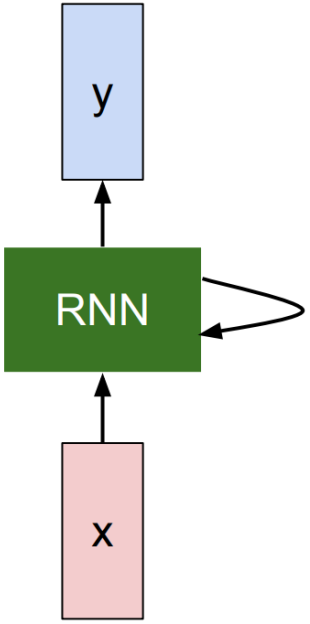
$$h_t = f_W(h_{t-1}, x_t)$$

new state

some function with parameters W

old state

input vector at some time step



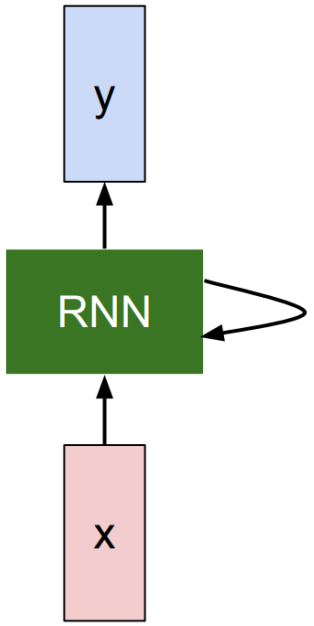

**The same set of parameters are used at very time step.**

# Recurrent Neural Network

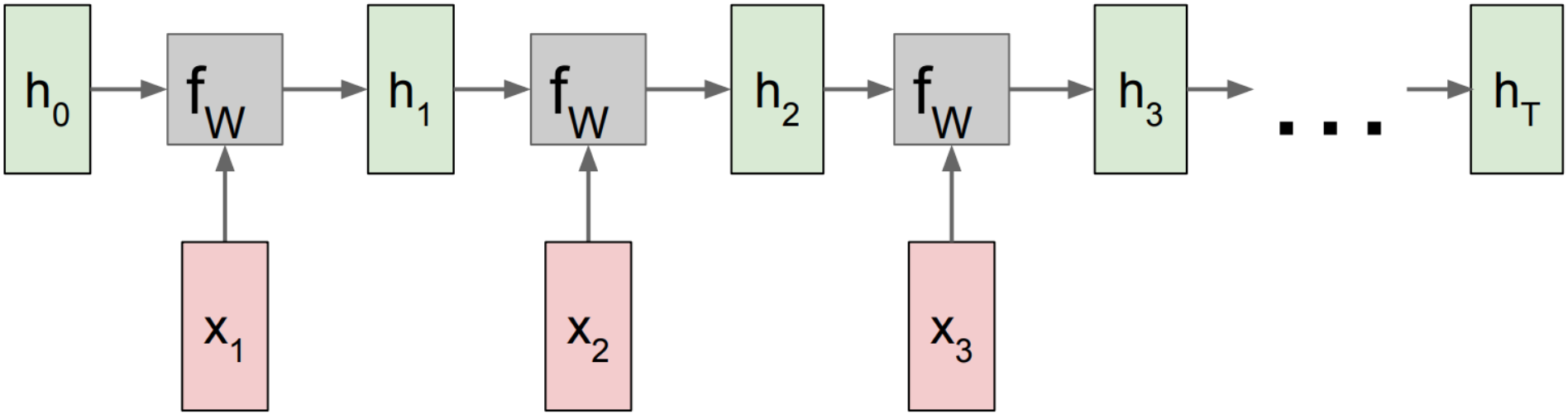
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = \tanh(W_{hh} h_{t-1} + W_{xh} x_t)$$

$$y_t = W_{hy} h_t$$

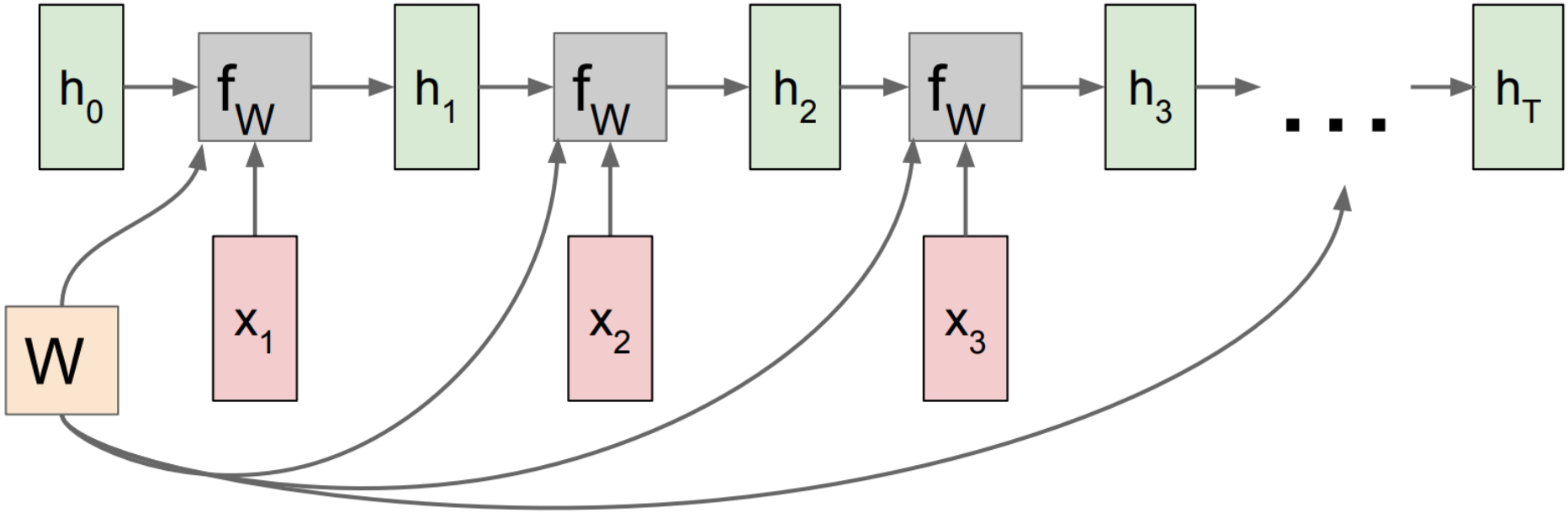


# RNN: unrolling

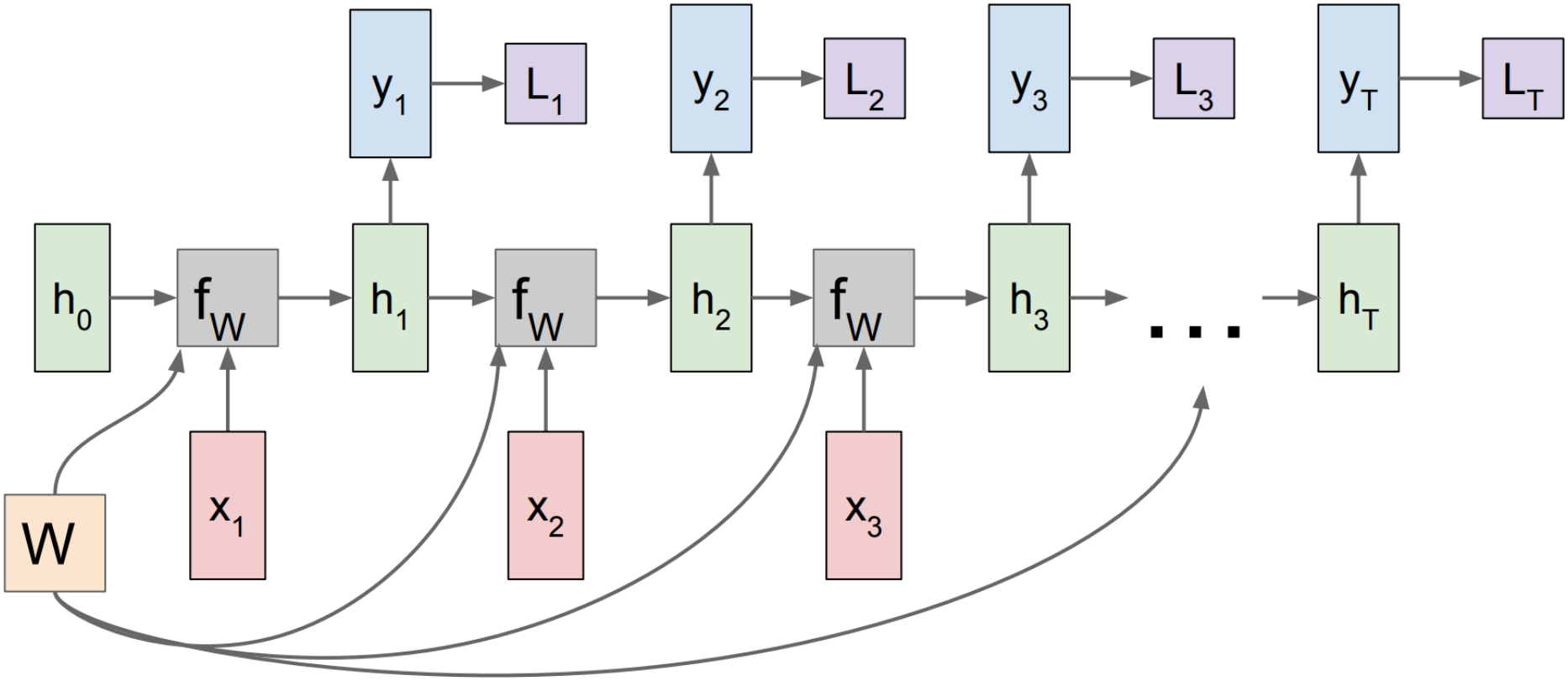


# RNN: unrolling

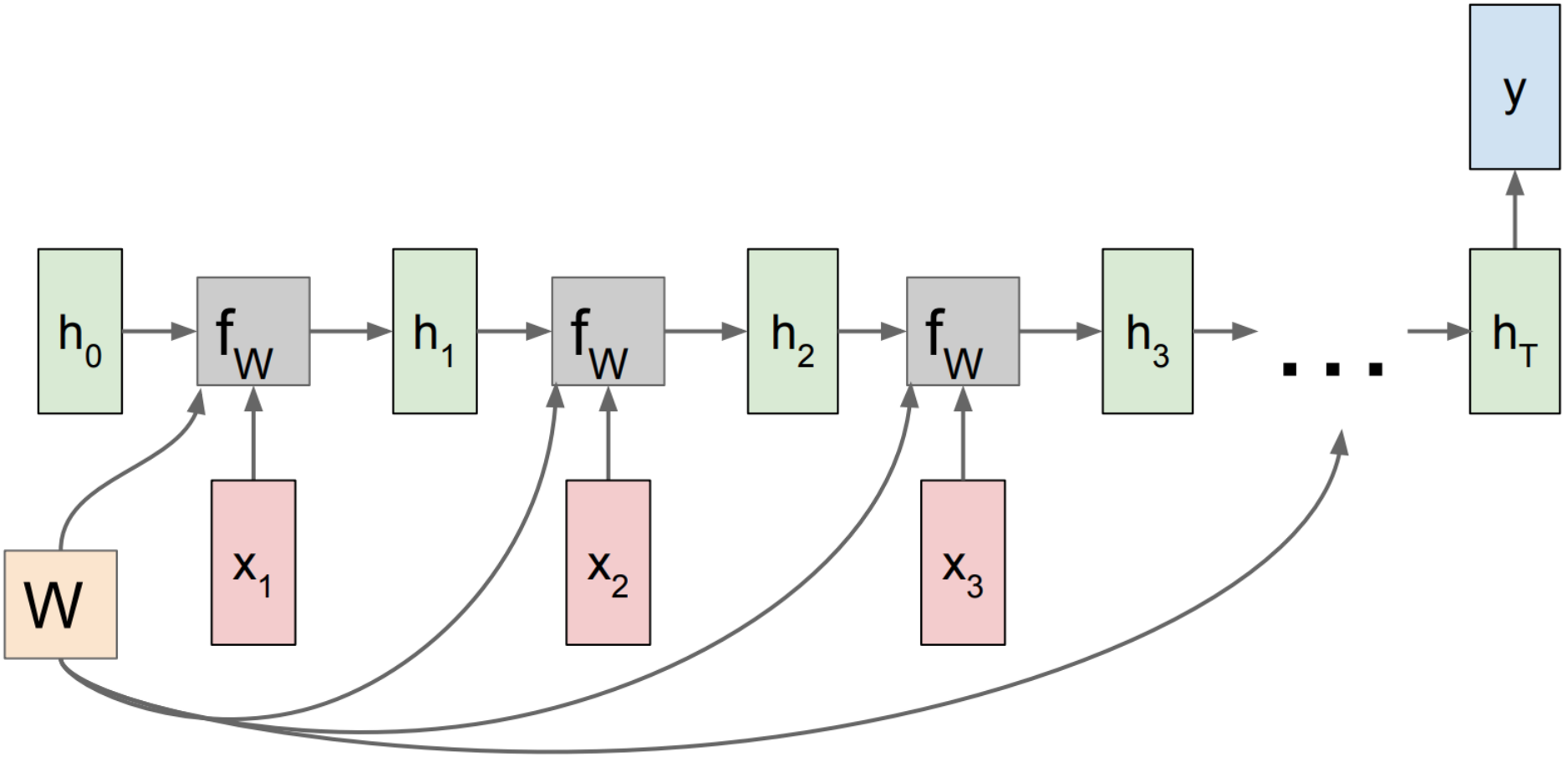
Reuse the same weight matrix at every time-step



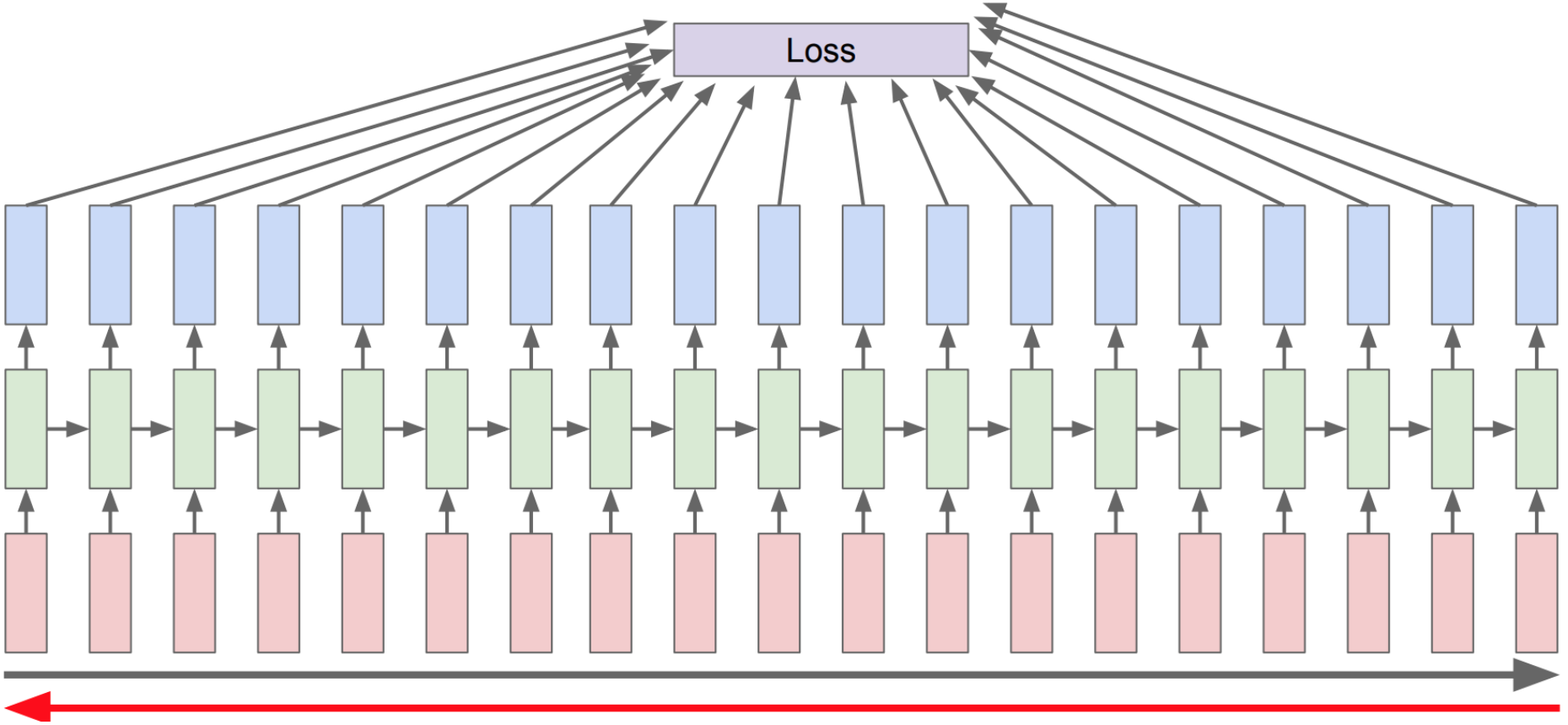
# RNN: many to many



# RNN: many to one

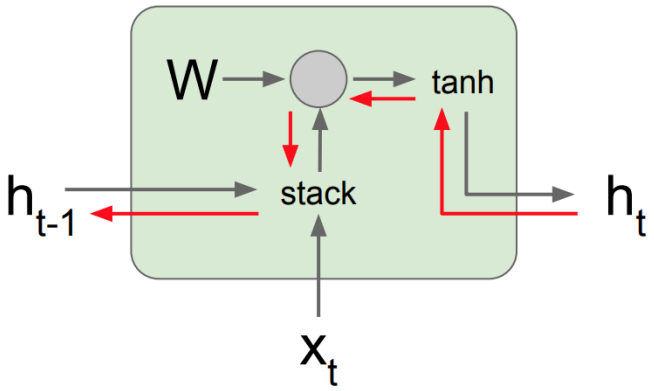


# RNN: truncated backpropagation



# Problem with RNN

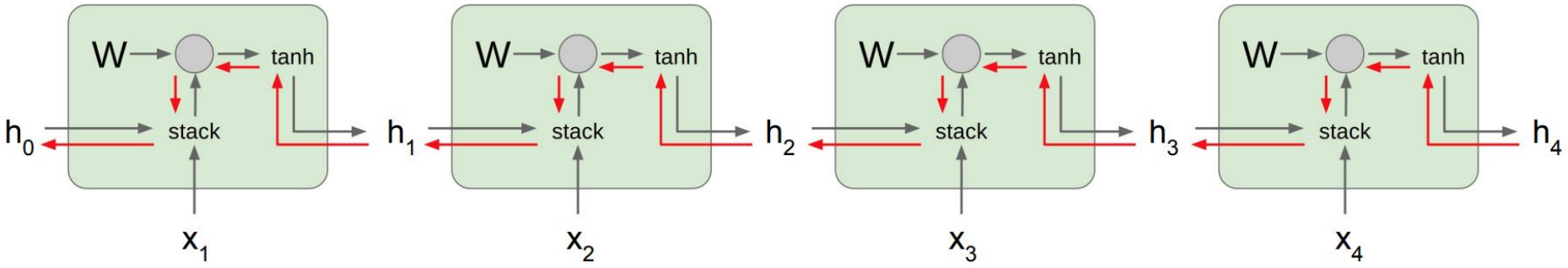
- Gradient flow in RNN




$$
\begin{aligned}
h_t &= \tanh(W_{hh}h_{t-1} + W_{xh}x_t) \\
&= \tanh\left( \begin{pmatrix} W_{hh} & W_{hx} \end{pmatrix} \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right) \\
&= \tanh\left( W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix} \right)
\end{aligned}
$$

# Problem with RNN

- Gradient flow in RNN




Computing gradient involves many factors of W

**Exploding gradients & Vanishing gradients**

# What do you need? (4 important pieces)

- State
  - Need to know how much prior state to keep or forget
  - Forget (remember) vector
    - 0 -> forget
    - 1 -> remember completely
- How to generate state update (innovation in KF)
  - Innovation vector
    - From prior state and observation
  - Weight vector
- How to generate current output
  - Output vector
    - Reveal current state to outside world

# Long Short Term Memory (LSTM)

RNN

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

LSTM

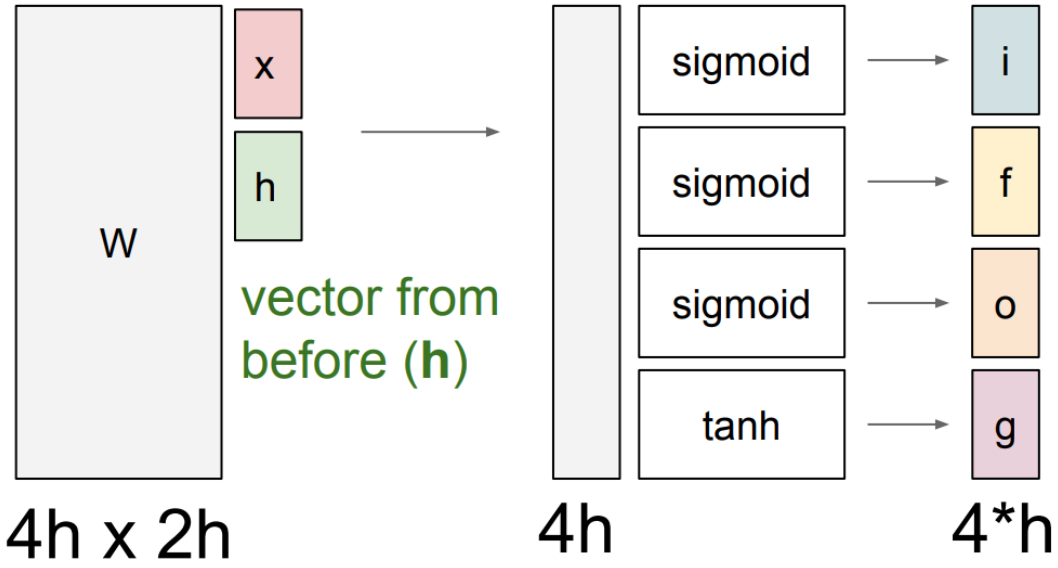
$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# Long Short Term Memory (LSTM)

f: whether to erase cell (forget)
i: whether to write to cell (input)
o: how much to reveal the cell (output)
g: how much to write to cell

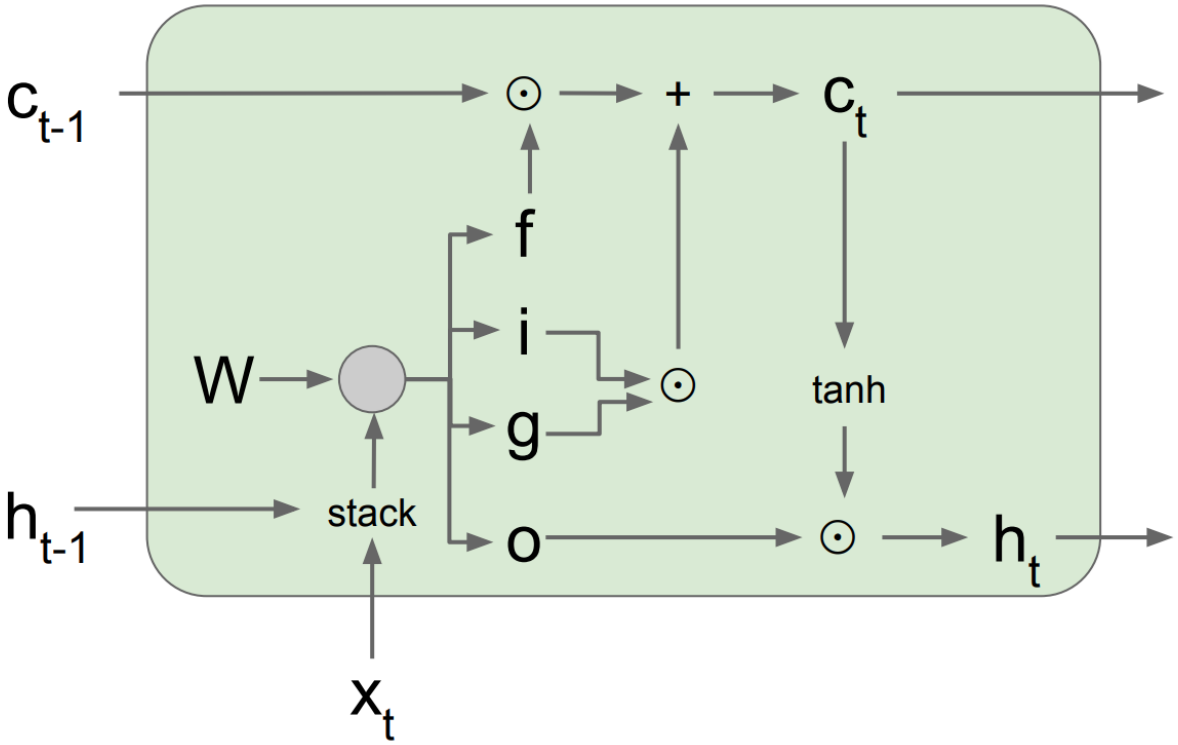



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

# Long Short Term Memory (LSTM)

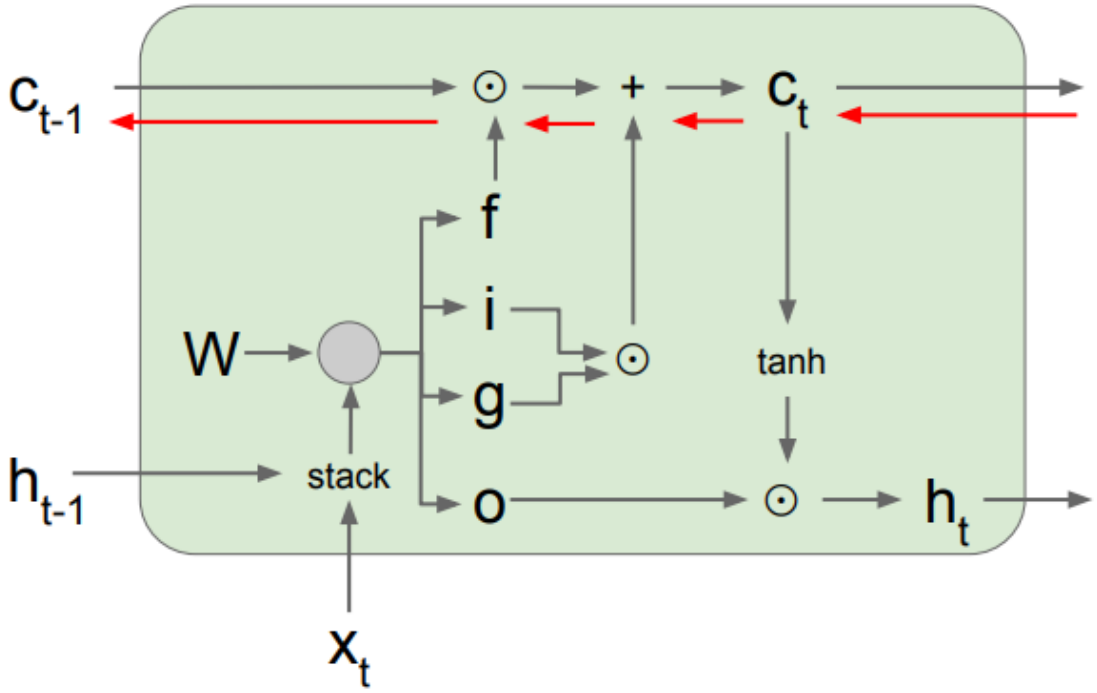



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

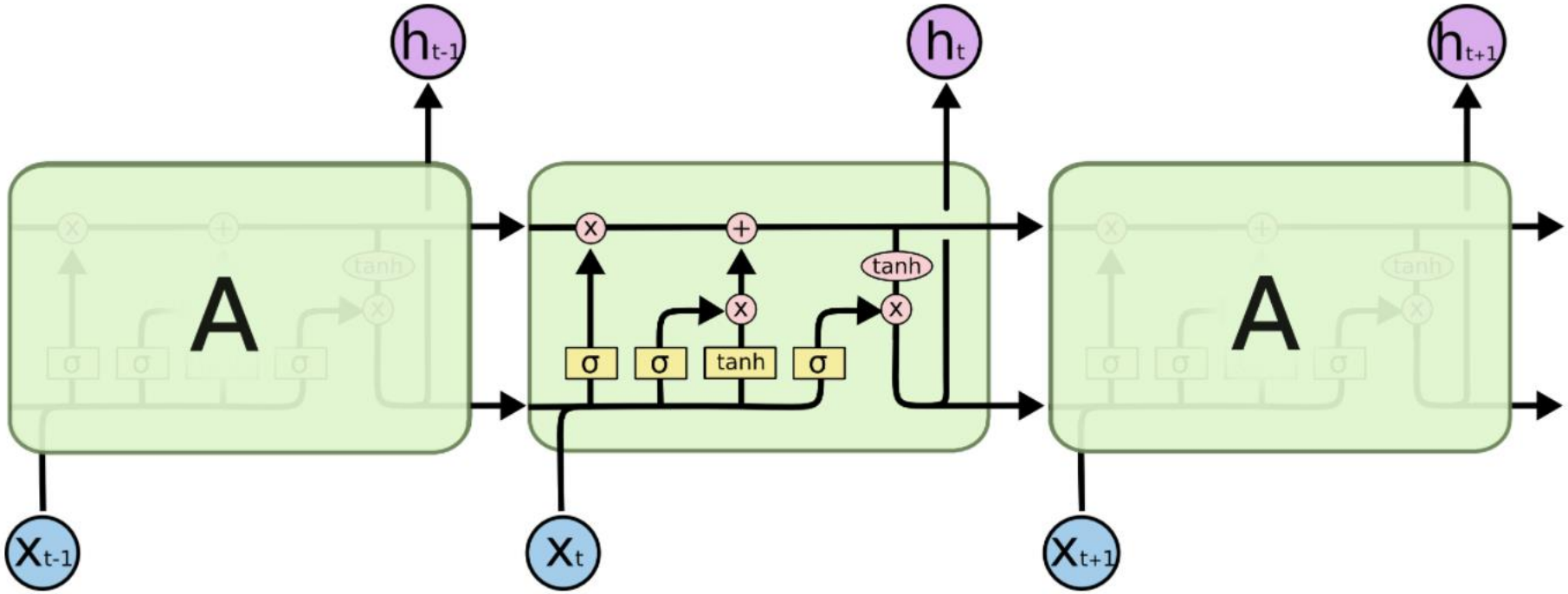
$$h_t = o \odot \tanh(c_t)$$

# Long Short Term Memory (LSTM)



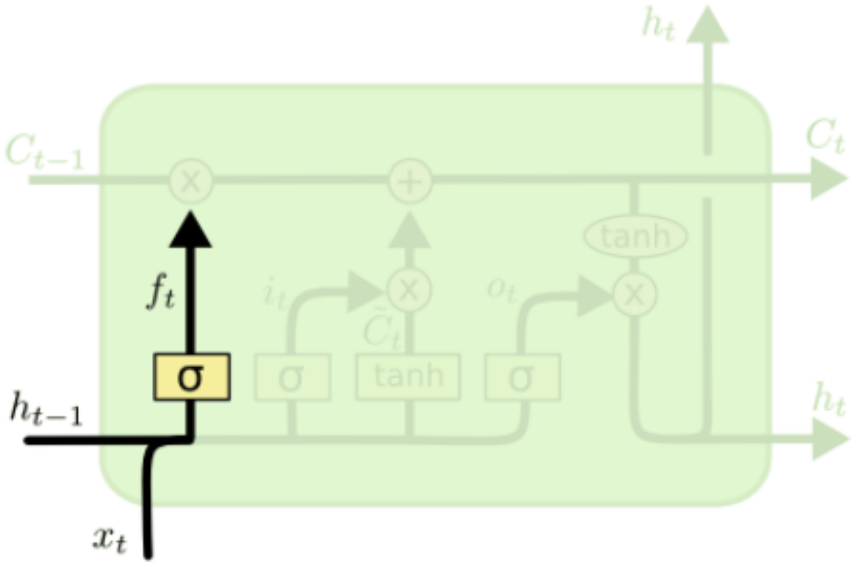

Back propagation from c_t to c_t-1 only elementwise multiplication by f, no matrix multiply by W

# Long Short Term Memory (LSTM)



# Forget or Remember

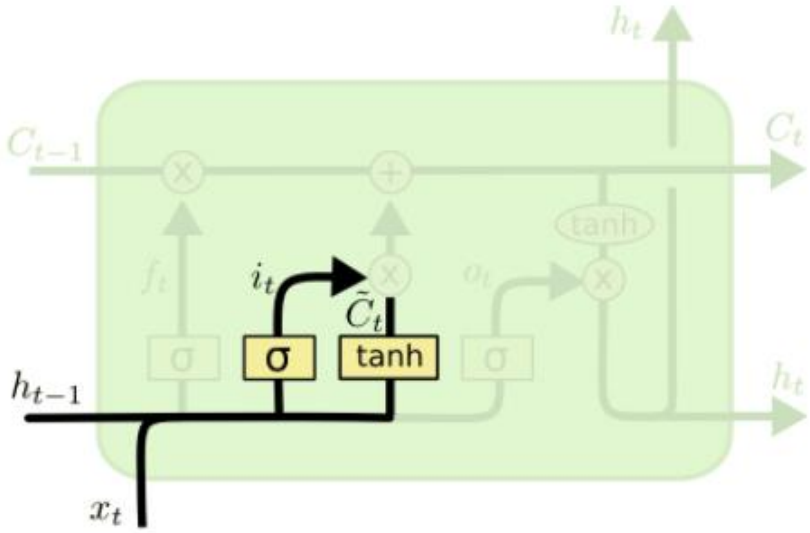
- How much prior memory matter
  - Depend on prior output and current input




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

# New prediction and importance

- Again, based on state inference (hungry) and observation (go to kitchen) hungry x go to kitchen -> full

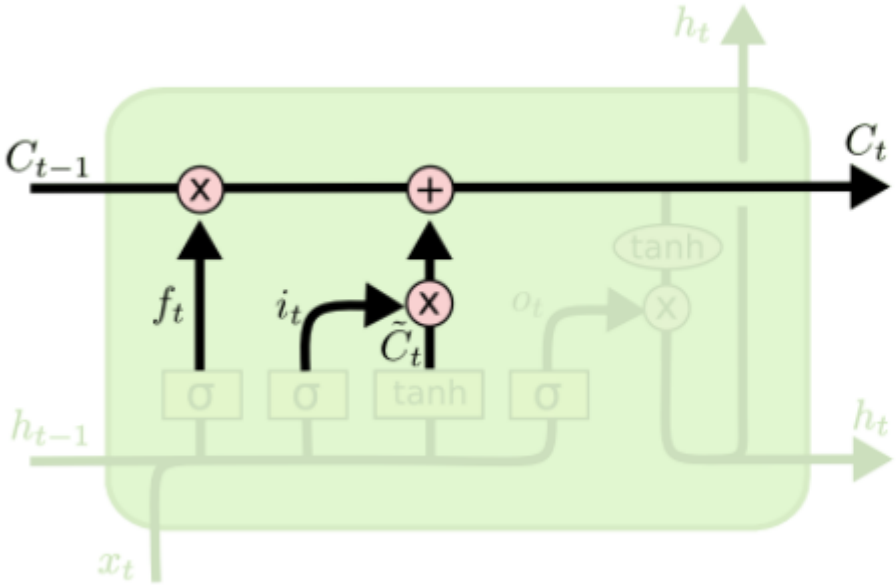



$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

# New State

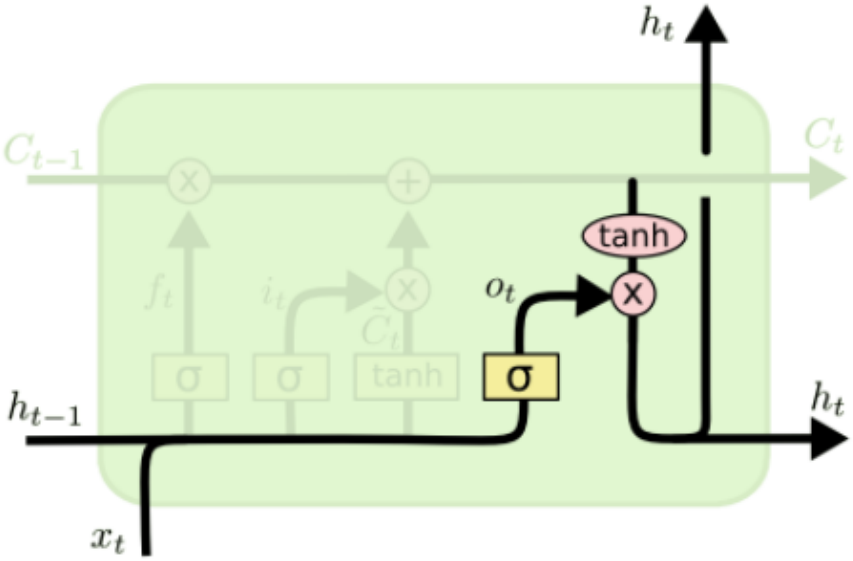
- Weighted sum of
  - Prior state with weight
  - New estimated state with weight




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

# New output

- Based on new state




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

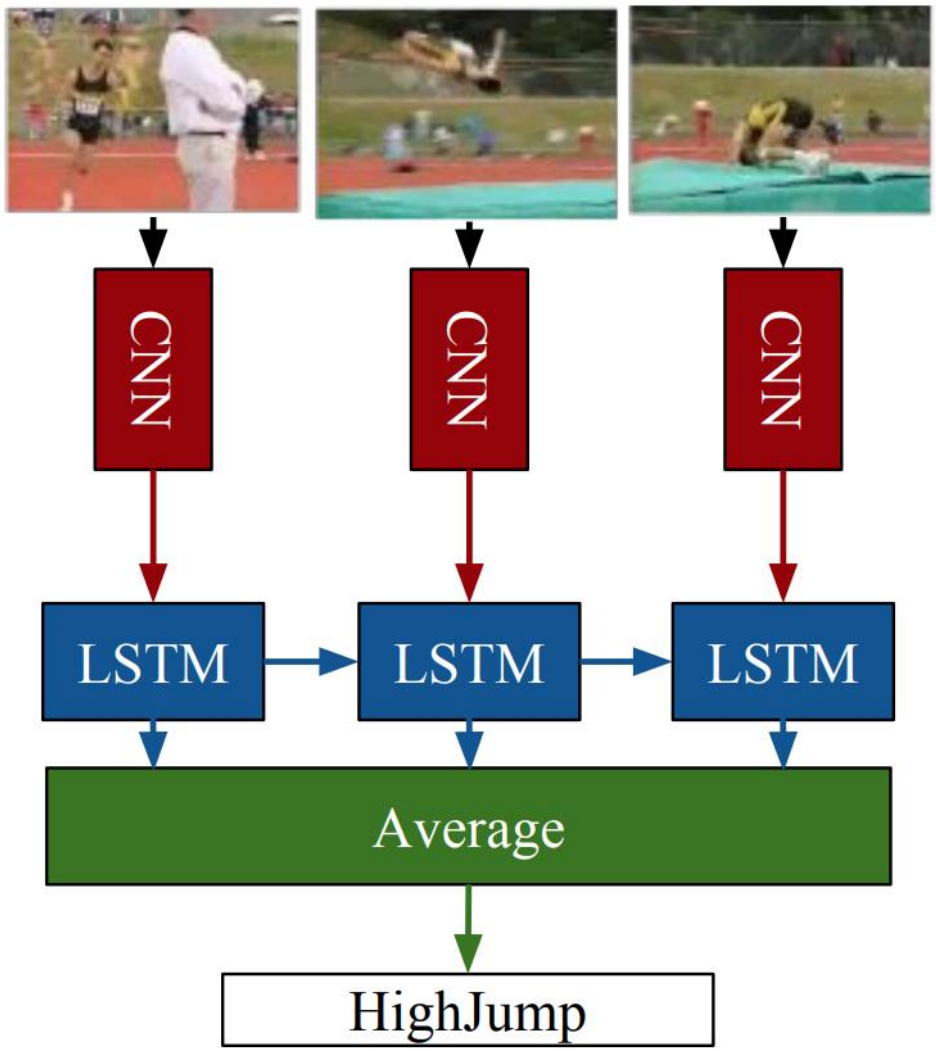
$$h_t = o_t * \tanh\left(C_t\right)$$

# Task 1: Activity Recognition

- The problem
  - Given a trimmed video, predict the activity label for this video

- Simple solution
  - Sample some frames in the video
  - Classify each image with a CNN
  - Aggregate the per-frame class scores (such as majority vote, etc.)

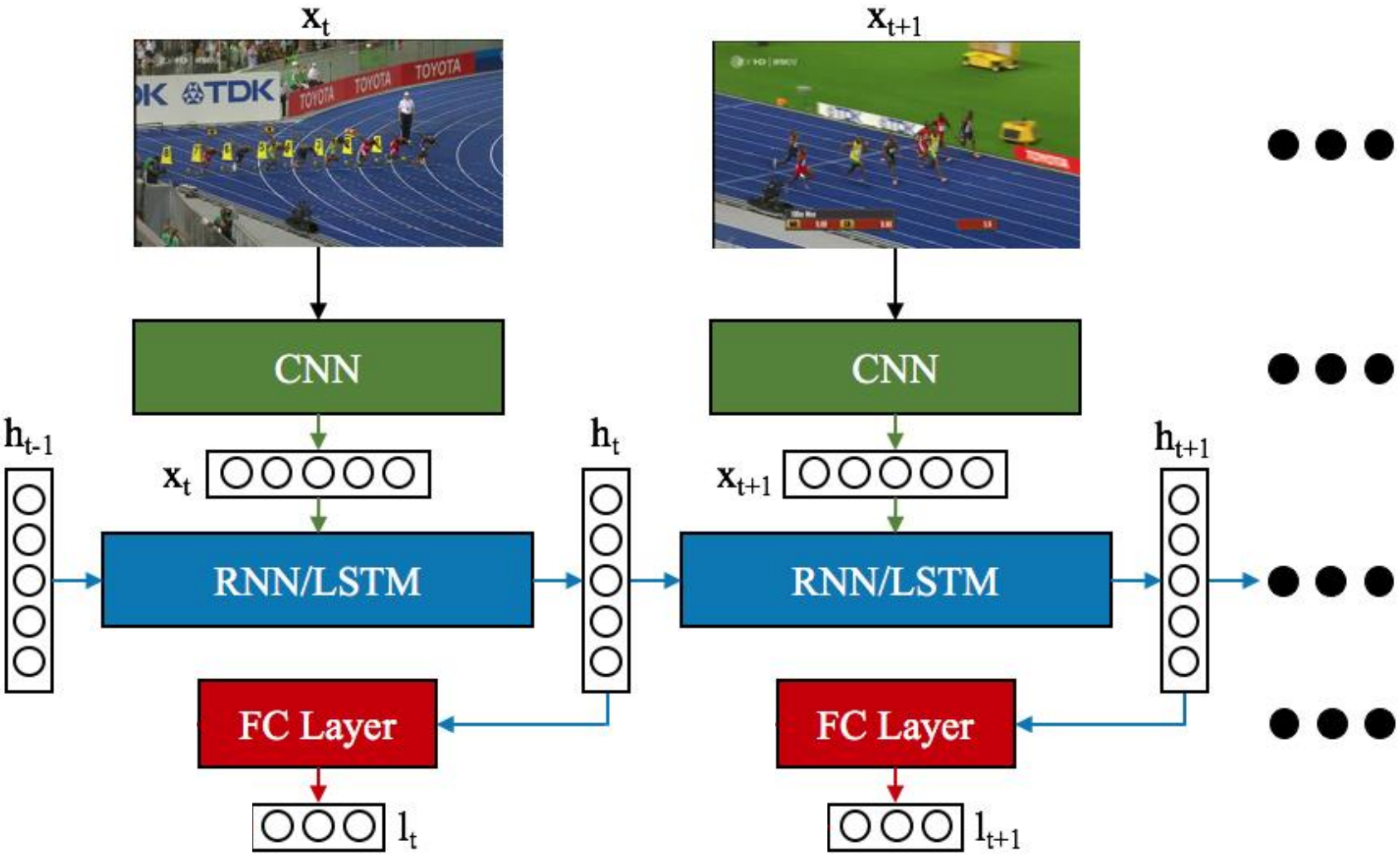
# Task 1: Activity Recognition




# Task 2: Object Tracking

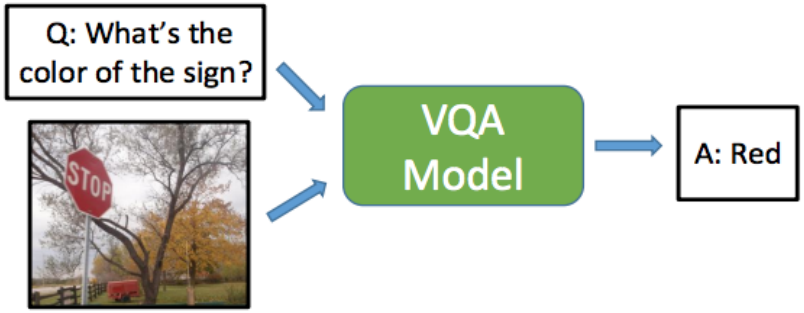
- The problem
  - Track one object in one video
  - GT is given in each frame in training set
  - GT is only given in the first frame in testing set


# Task 2: Object Tracking



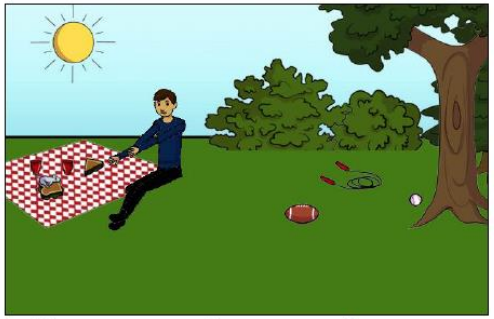
# Task 3: Visual Question Answering

- The problem
  - Given an image and a free question (in free text) about the image, output a textual answer.




What color are her eyes?
What is the mustache made of?

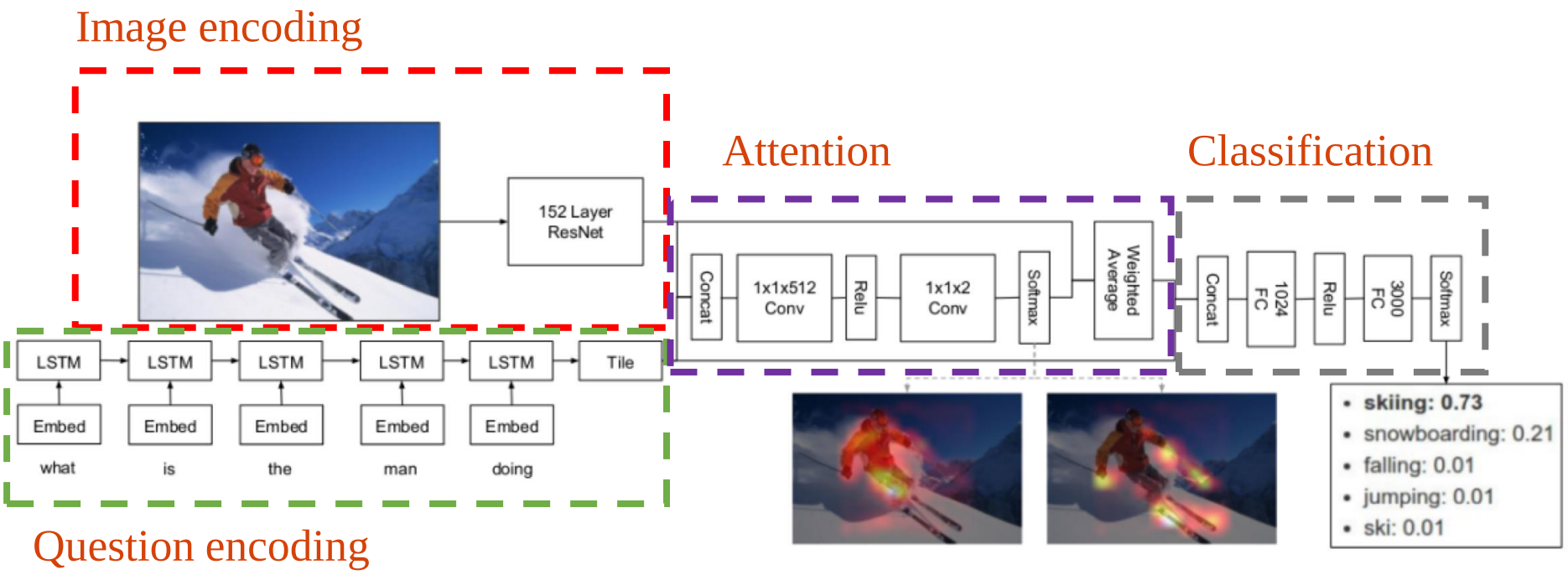
How many slices of pizza are there?
Is this a vegetarian pizza?



Is this person expecting company?
What is just under the tree?

Does it appear to be rainy?
Does this person have 20/20 vision?

# Task 3: Visual Question Answering



# Quick Summary

- RNN is better to process sequential data
- Simple RNNs are simple but don’t work very well
- Backward flow of gradients in RNN can explode or vanish. LSTM improves the gradient flow
- RNN/LSTMs are widely used in different vision problems.

# Thank You !