Recurrent Neural Networks

CS7150, Spring 2025

Prof. Huaizu Jiang

Northeastern University

Administrative stuff

- Anonymous Google form to solicit feedback of working on PA1
  - Your feedback is important
  - You can share your comments of this course in general
  - I’ll share responses on Friday
- Grades of Quiz 2 will be released on Friday
- Grades of PA1 will be released by Feb 19

Recap

Training Convolutional Networks

1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

If the model is big, won't we overfit?

Slide credit: D Fouhey & J Johnson

Regularizing CNNs: Weight Decay

$$L_{reg} = \frac{1}{2}\sum_{\ell} \|W_\ell\|^2 \qquad \frac{\partial L_{reg}}{\partial W_\ell} = W_\ell$$

Add L2 regularization term L_{reg} to the loss penalizing large weight matrices

Usually don’t regularize bias terms, or BatchNorm scale / shift params

*Technical note: Adding an explicit term to the loss is “L2 Regularization”; “Weight decay” adds a term to the gradient. Theyare equivalent for SGD, but not quite the same for other optimizers like Adam

Slide credit: D Fouhey & J Johnson

Regularizing CNNs: Data Augmentation

Hippo

Hippo?

Horizontal Flip

Hippo?

Color Jitter

Hippo?

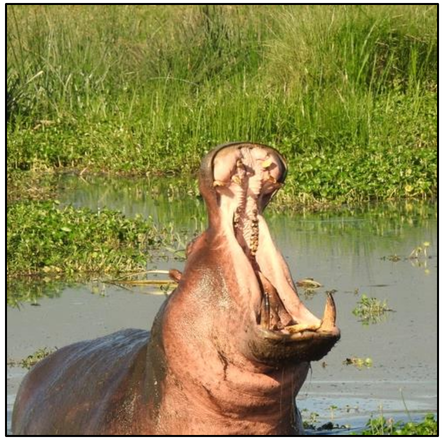
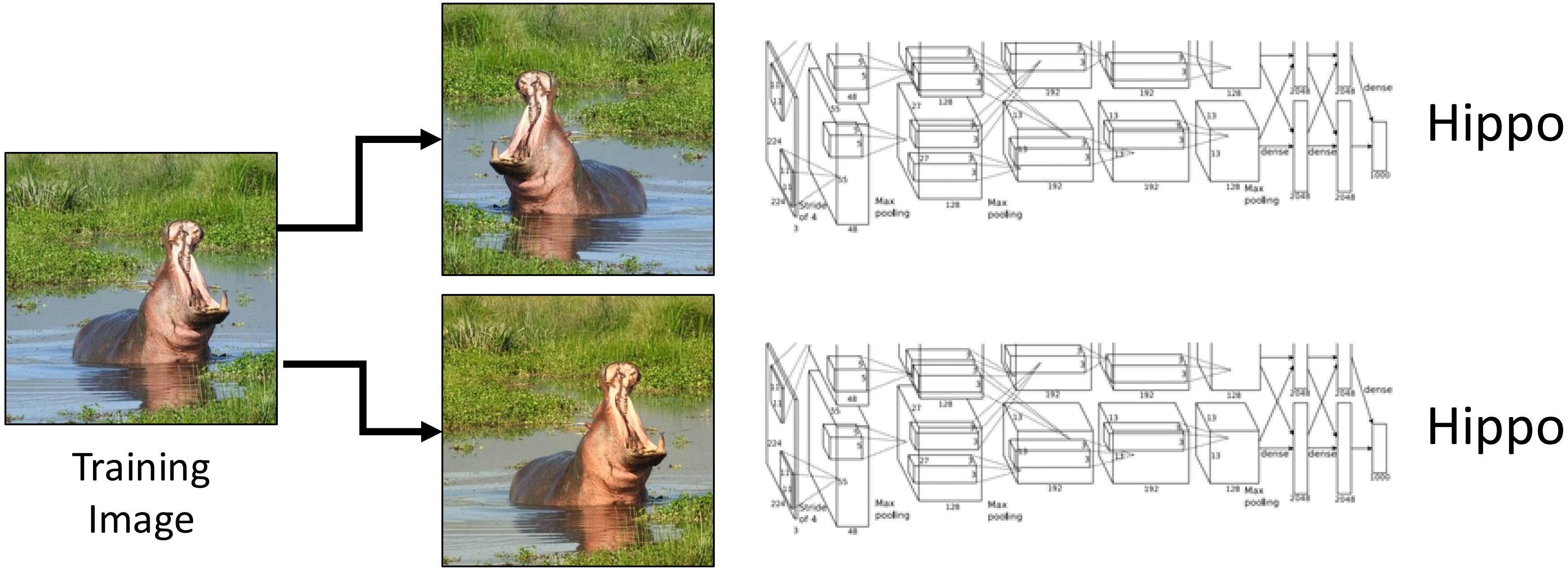


Image Cropping

Slide credit: D Fouhey & J Johnson

Regularizing CNNs: Data Augmentation

Apply random transformations to input images during training
Artificially “inflate” the size of your dataset



Training Image

Augmented images

Slide credit: D Fouhey & J Johnson

Training Convolutional Networks

1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

If the model is big, won't we overfit?

Slide credit: D Fouhey & J Johnson

Training Convolutional Networks

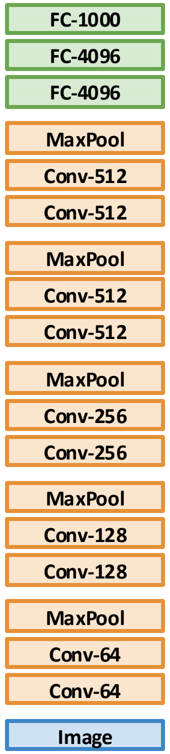
1. Download big datasets
2. Design CNN architecture
3. Initialize Weights
4. For t = 1 to T:
    1. Form minibatch
    2. Compute loss + gradient
    3. Update Weights
5. Apply trained model to task

What if we can't find one?

Slide credit: D Fouhey & J Johnson

Transfer Learning: Feature Extraction

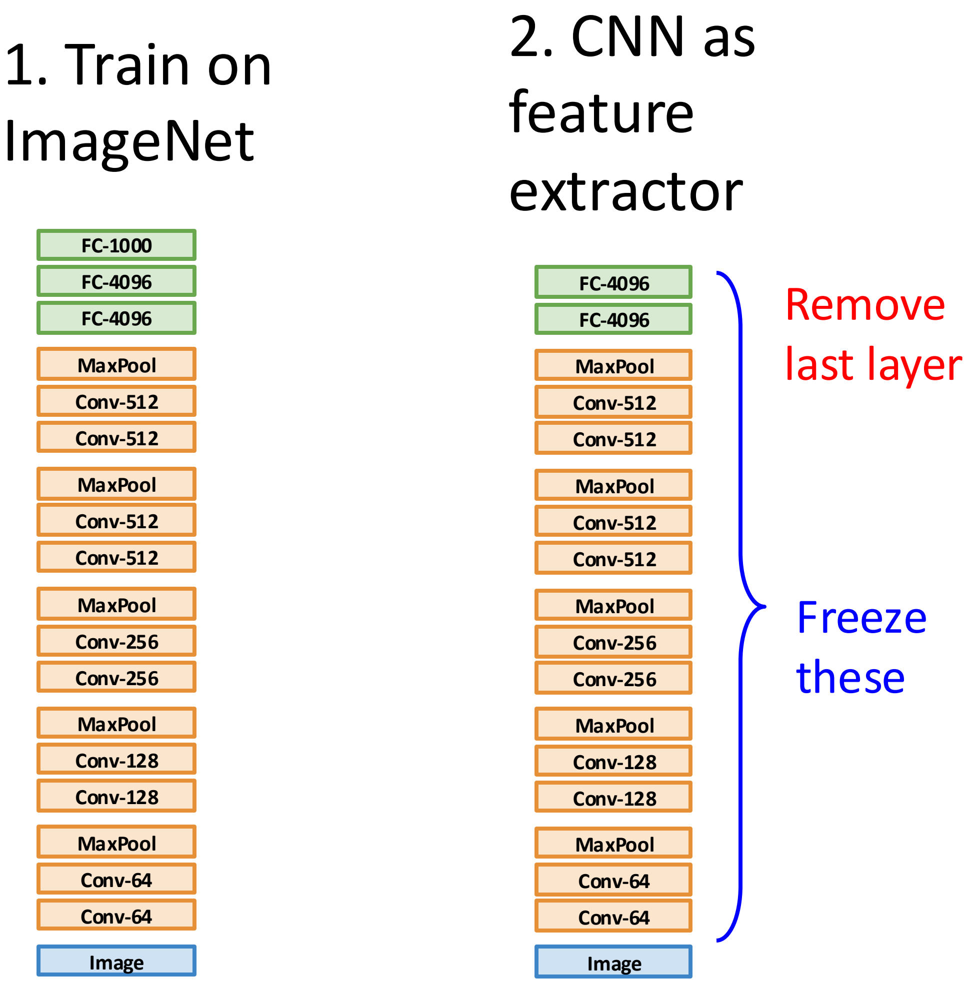
1. Train on ImageNet



Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

Slide credit: D Fouhey & J Johnson

Transfer Learning: Feature Extraction



Use your small dataset to train a **linear classifier** on top of pretrained CNN features

Donahue et al, "DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition", ICML 2014

Slide credit: D Fouhey & J Johnson

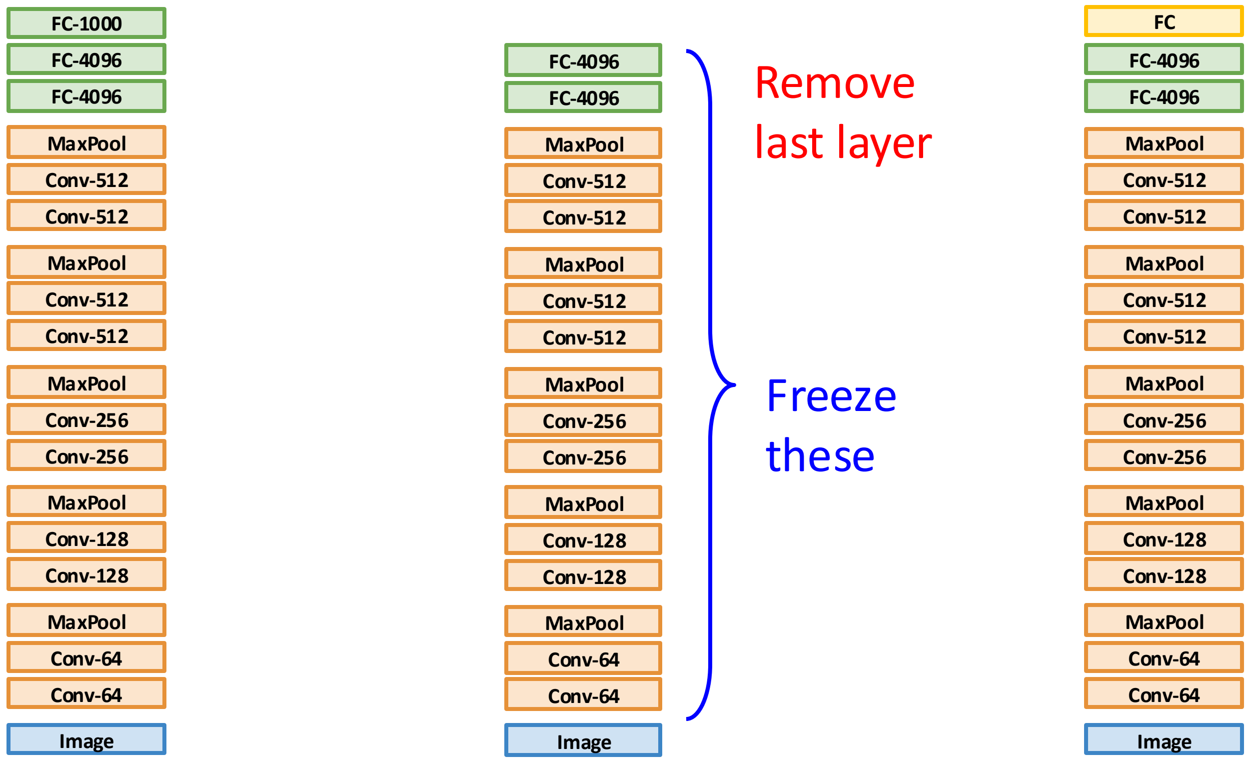
Transfer Learning: Fine-Tuning

1. Train on ImageNet

2. CNN as feature extractor

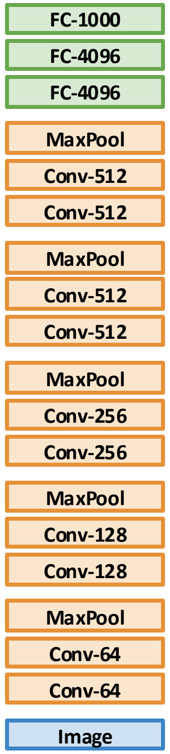
3. Bigger dataset: **Fine-Tuning**

Reinitialize last layer and continue training whole network on your dataset

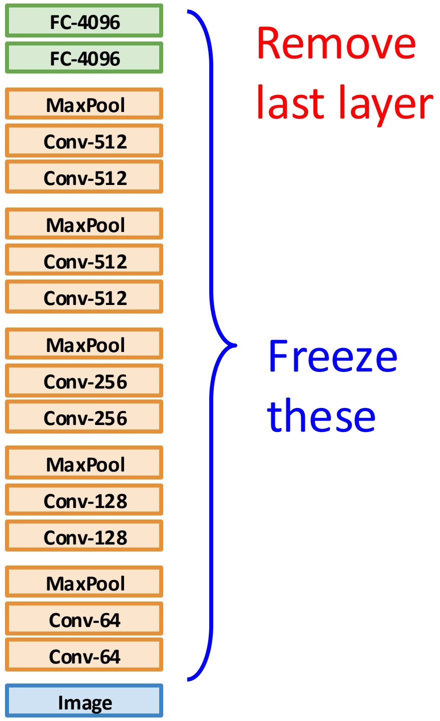


Transfer Learning: Fine-Tuning

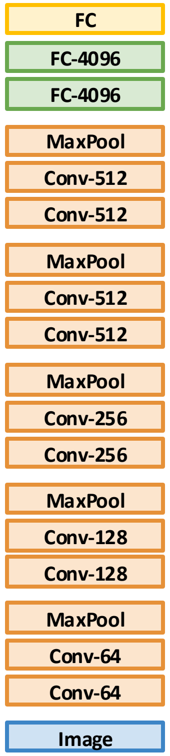
1. Train on ImageNet



2. CNN as feature extractor



3. Bigger dataset: **Fine-Tuning**



Reinitialize last layer and continue training whole network on your dataset

Some tricks:
- Train with feature extraction first before fine-tuning
- Lower the learning rate: use ~1/10 of LR used in original training
- Sometimes freeze lower layers to save computation

Slide credit: D Fouhey & J Johnson

Recurrent Neural Networks

Motivation Example: Sentiment classification

- Goal: classify a text sequence (e.g., restaurant, movie or product review, Tweet) as having positive or negative sentiment
  - “The food was really good”
  - “The vacuum cleaner broke within two weeks”
  - “The movie had slow parts, but overall was worth watching”
- What makes this problem challenging?
- What feature representation or predictor structure can we use for this problem?

Slide credit: S. Lazebnik

Encoding and decoding words

Input

“The cat sat on the mat.”
“The mat is under the cat.”

Tokenization

[“The”, “cat”, “sat”, “on”, “the”, “mat”]
[“The”, “mat”, “is”, “under”, “the”, “cat”]

Vocabulary (8 tokens)

[“The”, “cat”, “sat”, “on”, “the”, “mat”,
“is”, “under”]

One-hot embeddings

“the” -> 0 -> [1, 0, 0, …, 0] (length of 8)

“cat” -> 1 -> [0, 1, 0, …, 0]

“sat” -> 2 -> [0, 0, 1, …, 0]

Word embeddings

embed = Wx (x is the one-hot embedding)

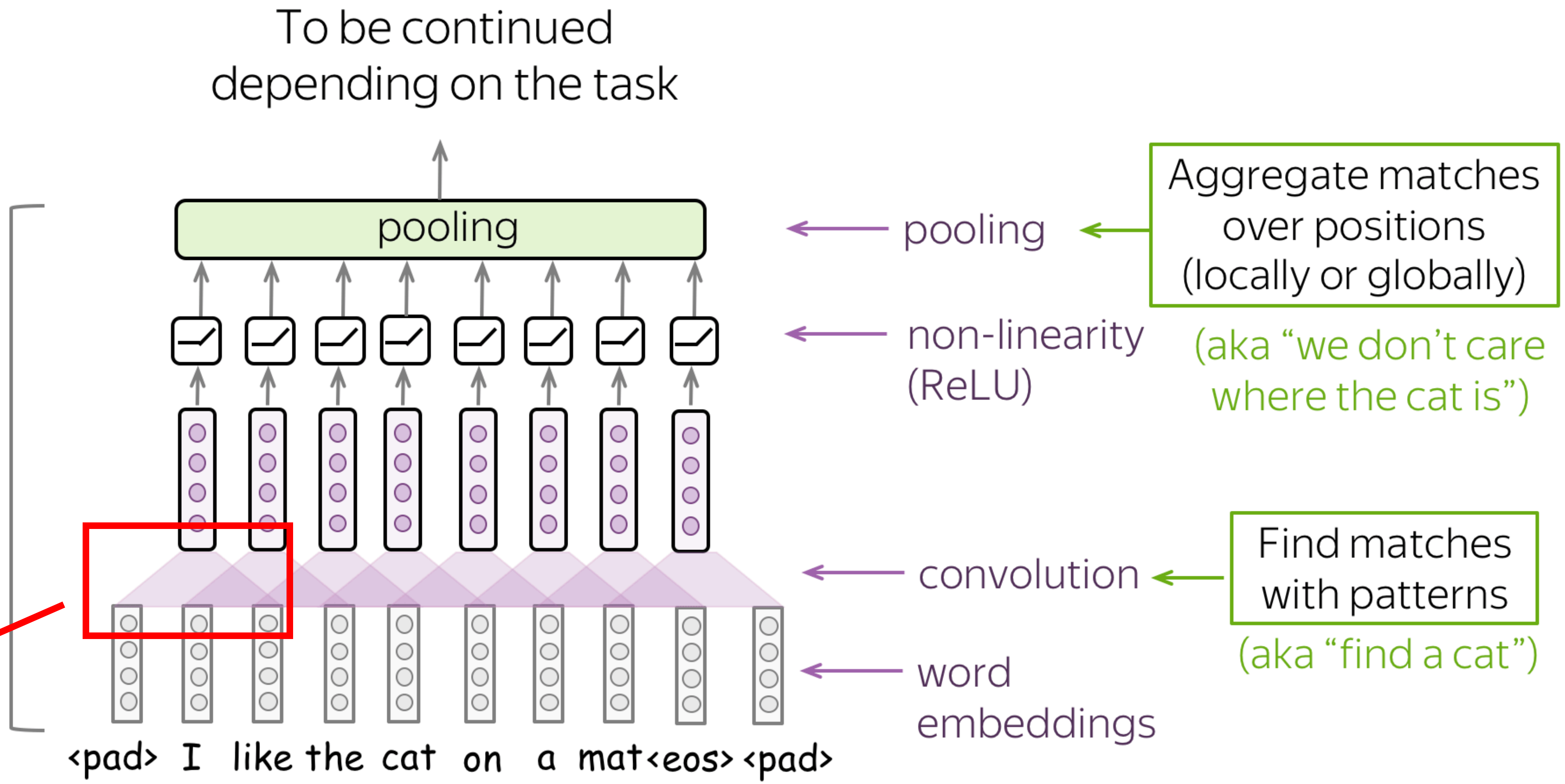
(See torch.nn.Embedding)

Word generation

8-category classification

Text Classification with CNNs

Typical usage CNNs for texts



How large should the convolution kernel be?

https://lena-voita.github.io/nlp_course/models/convolutional.html

Motivation of using RNNs

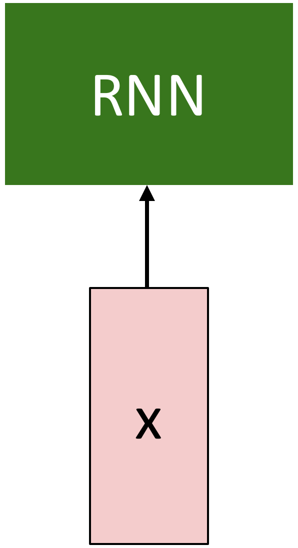
- Not all problems can be converted into one with fixed-length inputs and outputs

- Problems such as Speech Recognition or Time-series Prediction require a system to store and use context information
  - Simple case: Output YES if the number of 1s is even, else NO
    1000010101 – YES, 100011 – NO, …

- Hard/Impossible to choose a fixed context window
  - There can always be a new sample longer than anything seen

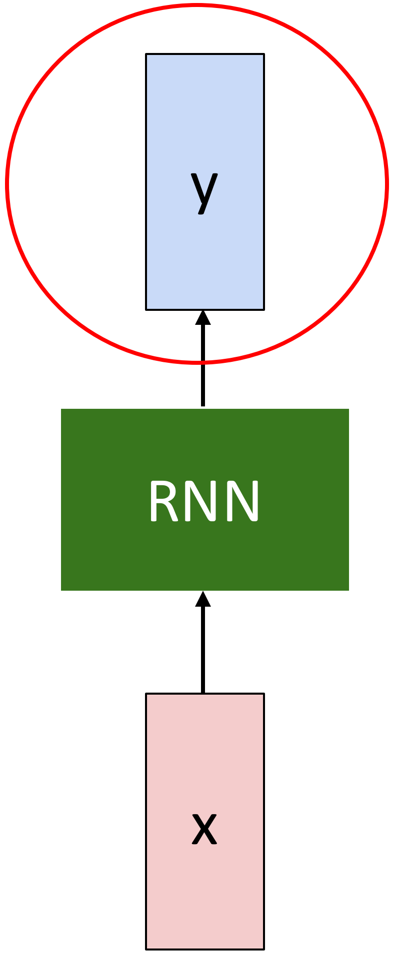
Slide credit: A. Mallya

Recurrent Neural Network



Slide credit: E. Learned-Miller, originally developed by Andrej Karpathy, Justin Johnson, Fei-Fei Li

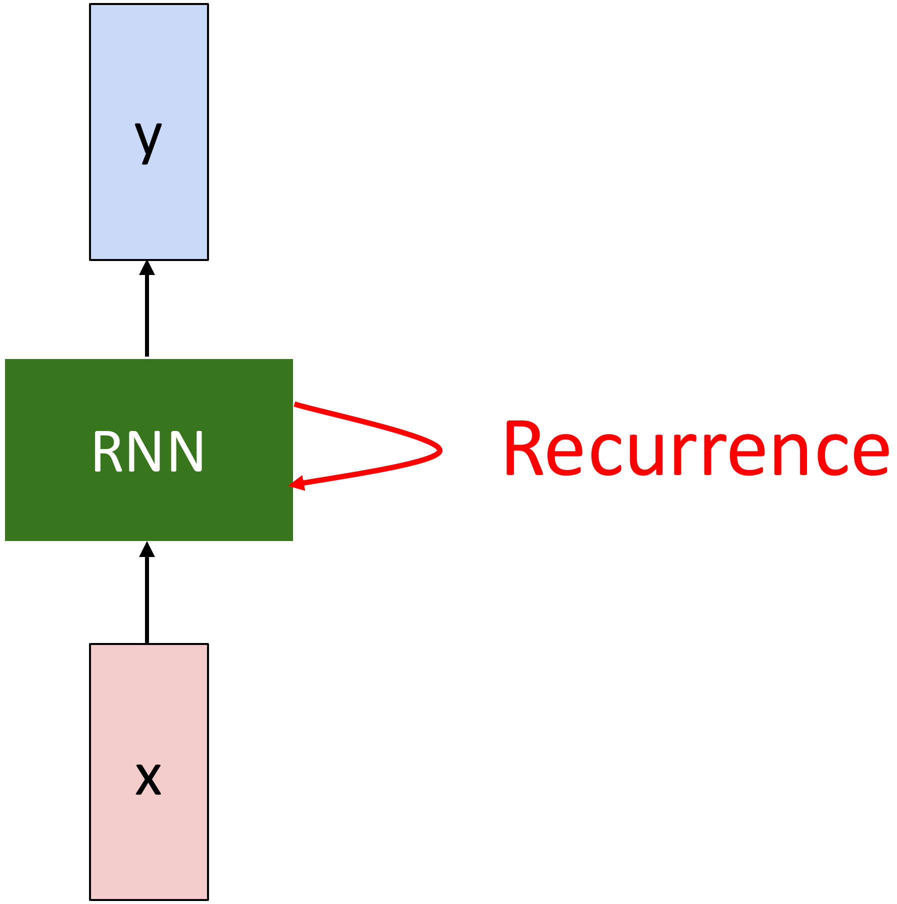
Recurrent Neural Network



usually want to predict a vector at some time steps

Slide credit: E. Learned-Miller, originally developed by Andrej Karpathy, Justin Johnson, Fei-Fei Li

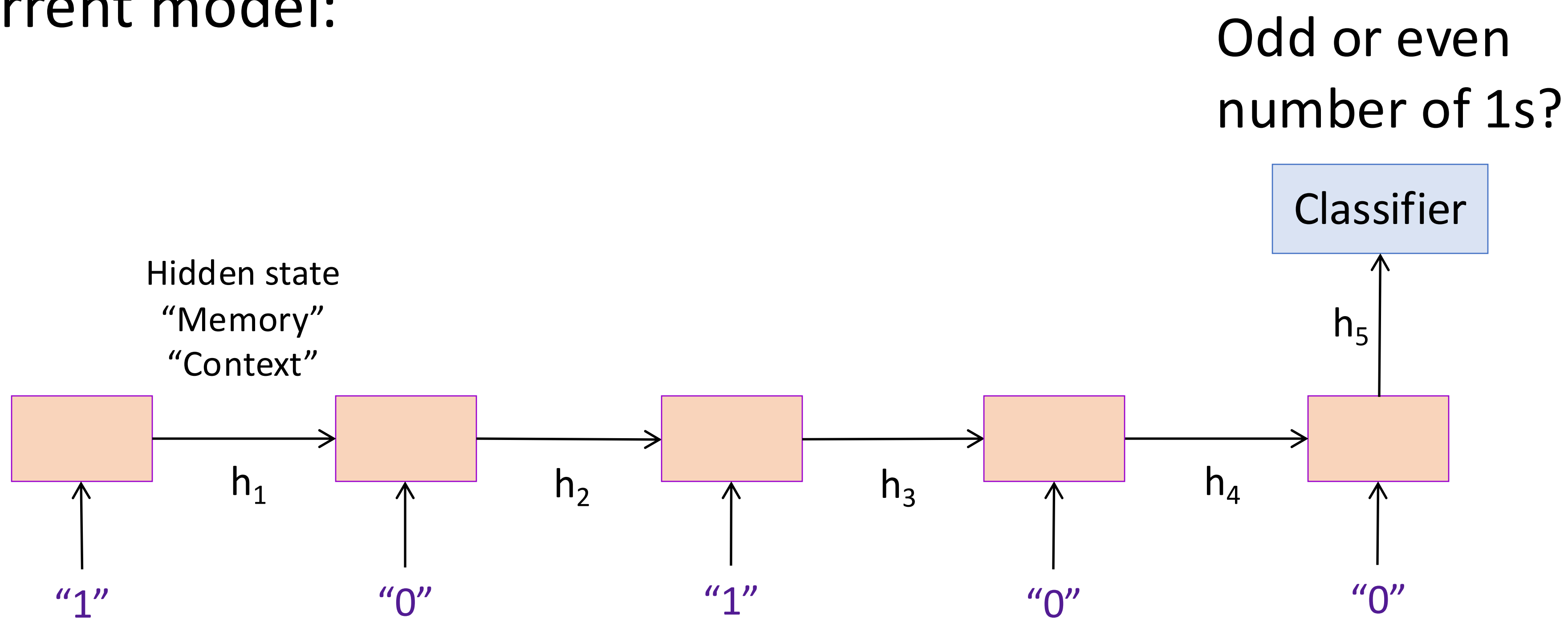
Recurrent Neural Network



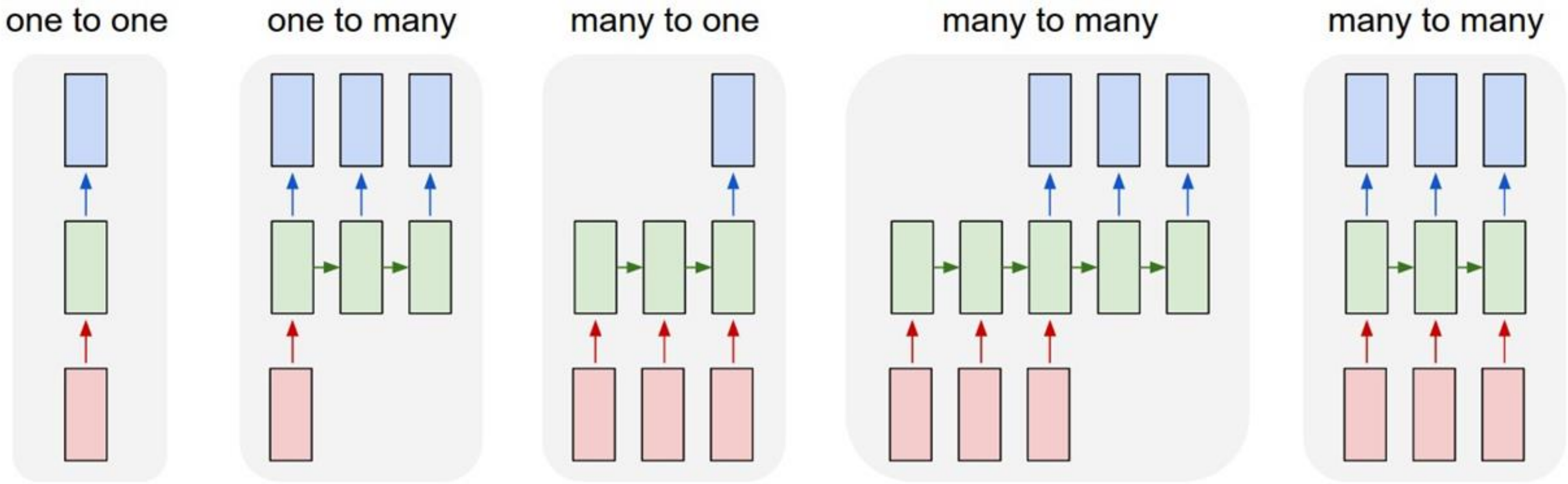
Slide adapted from: E. Learned-Miller, originally developed by Andrej Karpathy, Justin Johnson, Fei-Fei Li

Sequence classification using a RNN

- Recurrent model:



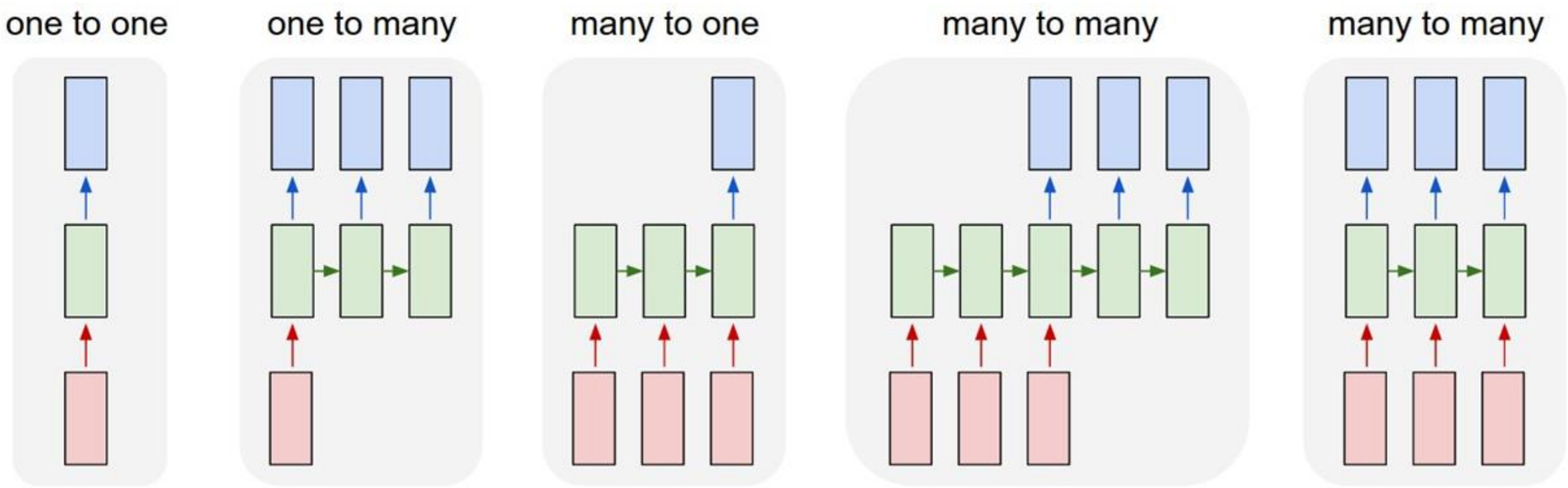
Recurrent Networks offer a lot of flexibility:



Vanilla Neural Networks

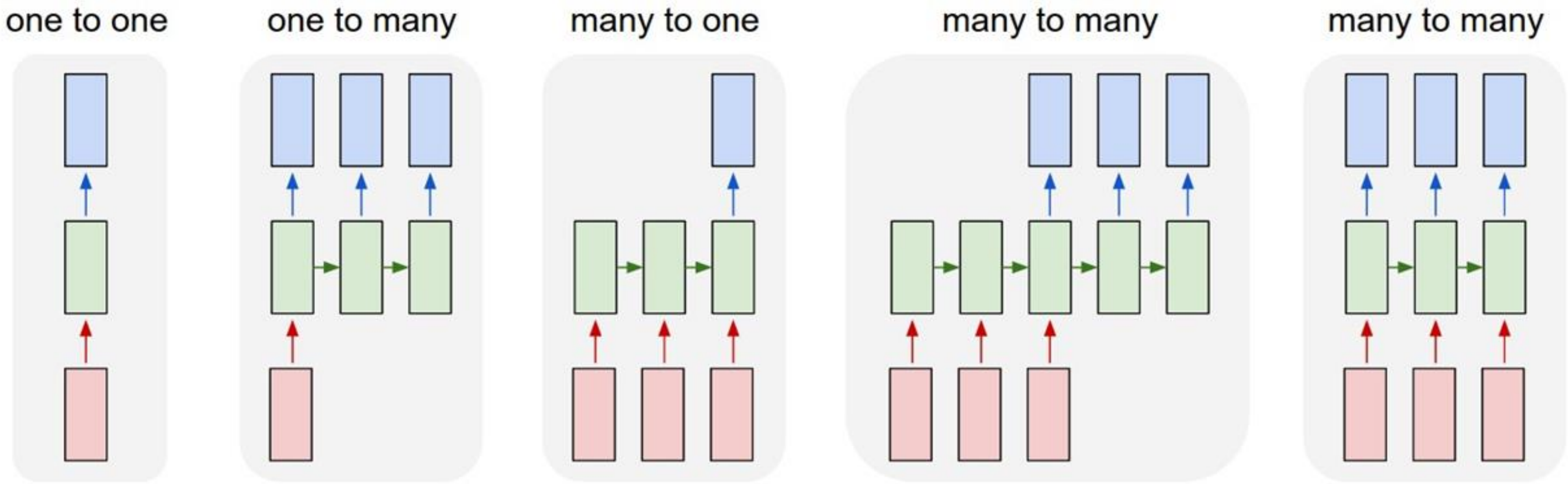
Slide credit: E. Learned-Miller, originally developed by Andrej Karpathy, Justin Johnson, Fei-Fei Li

Recurrent Networks offer a lot of flexibility:



e.g. **Image Captioning**
image -> sequence of words

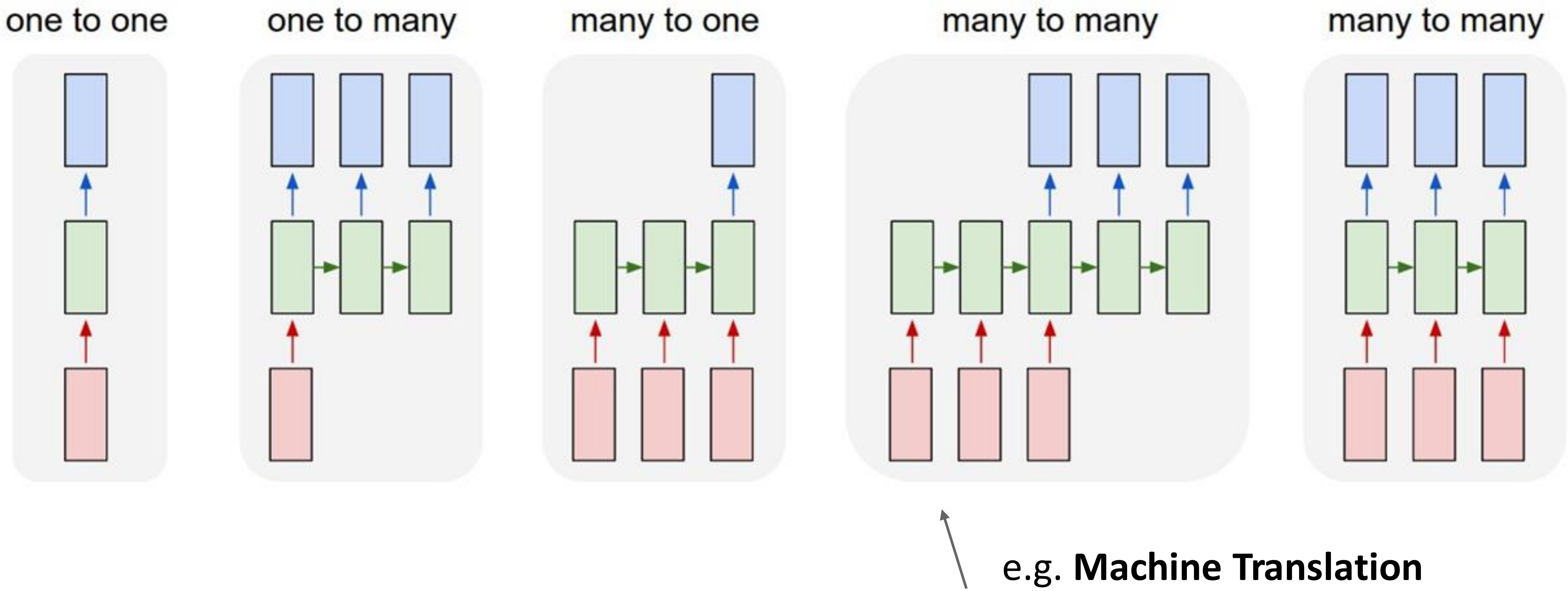
Recurrent Networks offer a lot of flexibility:



e.g. **Sentiment Classification**
sequence of words -> sentiment

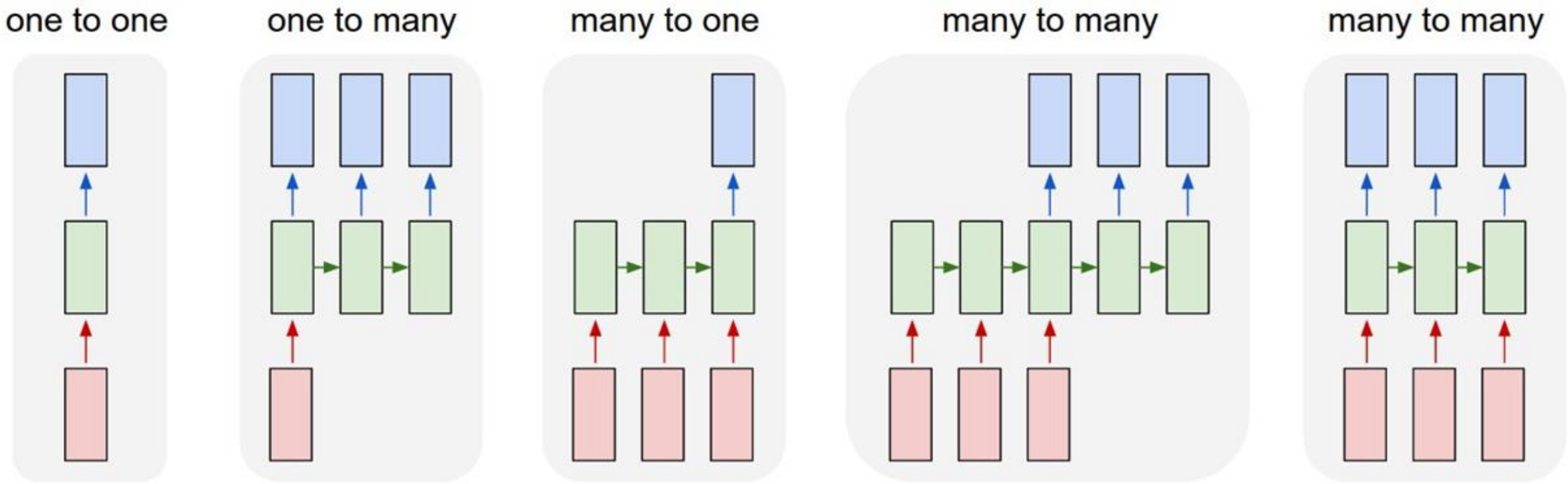
Slide credit: E. Learned-Miller, originally developed by Andrej Karpathy, Justin Johnson, Fei-Fei Li

Recurrent Networks offer a lot of flexibility:



e.g. **Machine Translation**
seq of words -> seq of words

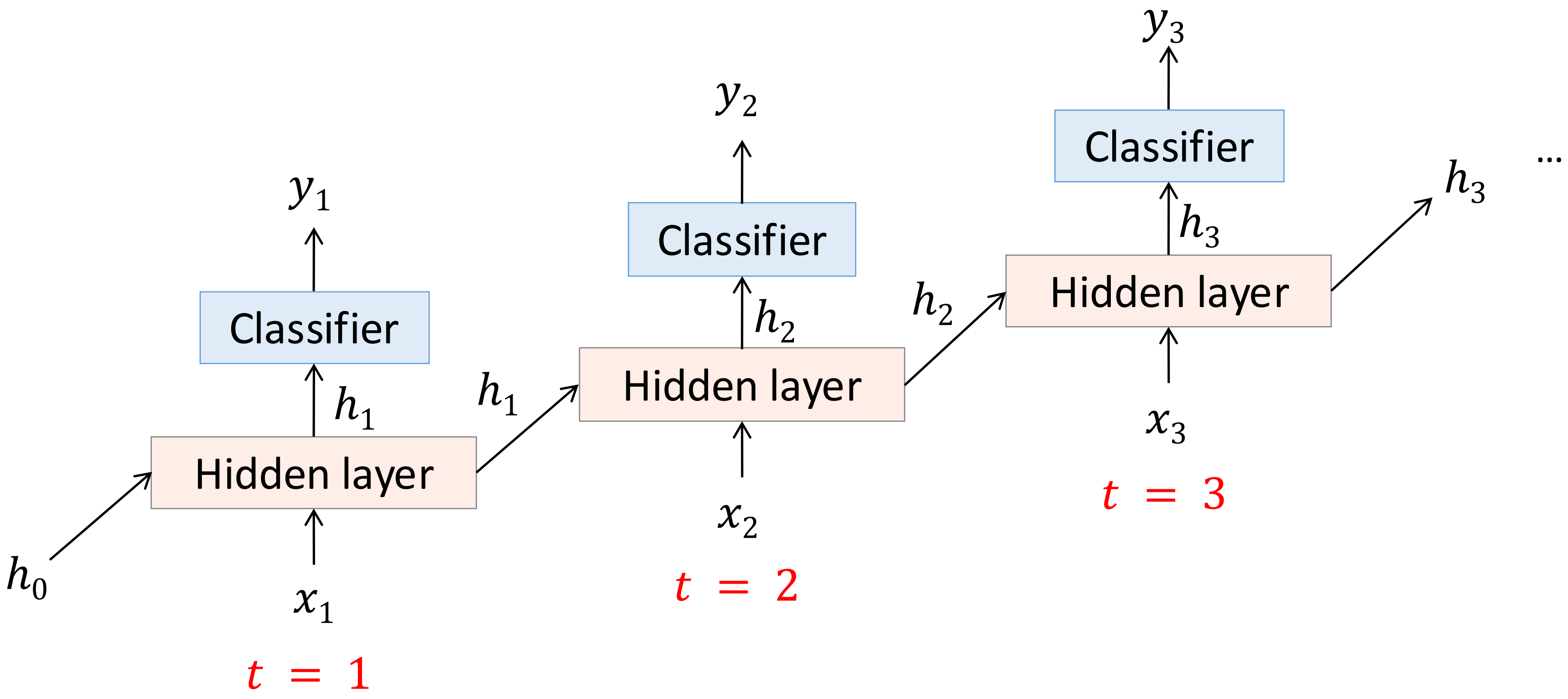
Recurrent Networks offer a lot of flexibility:



e.g. **Video classification on frame level**

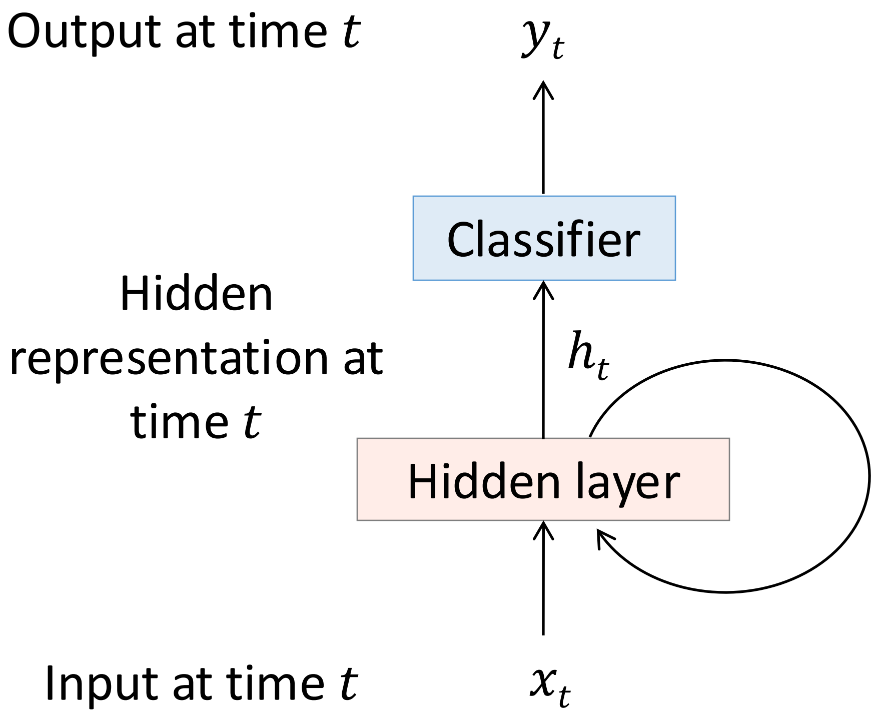
Slide credit: E. Learned-Miller, originally developed by Andrej Karpathy, Justin Johnson, Fei-Fei Li

Recurrent unit/cell



Slide credit: S. Lazebnik

Recurrent unit/cell

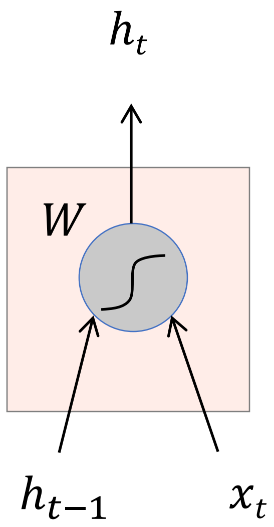


Recurrence:

$$\underbrace{h_t}_{\text{new state}} = \underbrace{f_W}_{\text{function of } W}(\underbrace{x_t}_{\text{input at time } t}, \underbrace{h_{t-1}}_{\text{old state}})$$

Slide credit: S. Lazebnik

Vanilla RNN cell



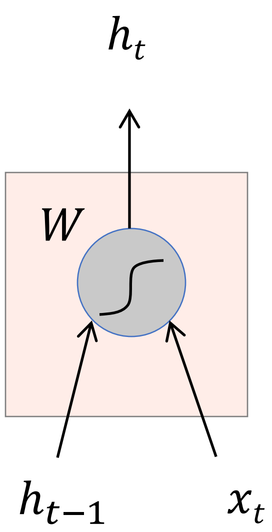
$$h_t = f_W(x_t, h_{t-1})$$

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

Could be ReLU [Nair and Hinton. ICML, 2010] as well

Slide adapted from: S. Lazebnik

Vanilla RNN cell

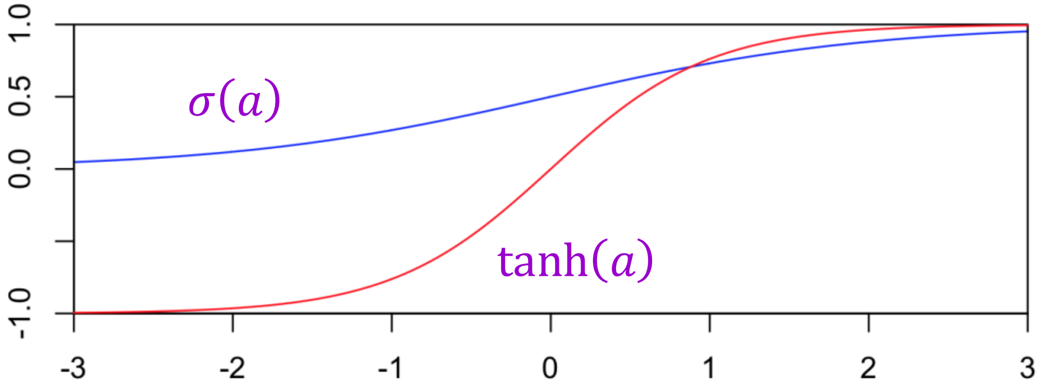


$$h_t = f_W(x_t, h_{t-1})$$

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

$$\sigma(a) = \frac{e^a}{e^a + 1}$$

Sigmoid, [0, 1]

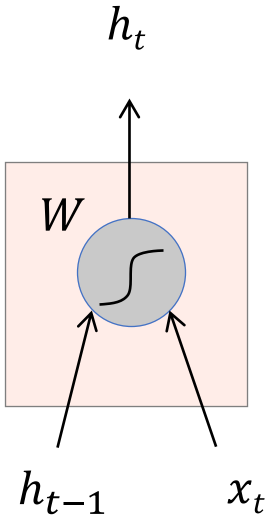


$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}}$$

$$= 2\sigma(2a) - 1$$

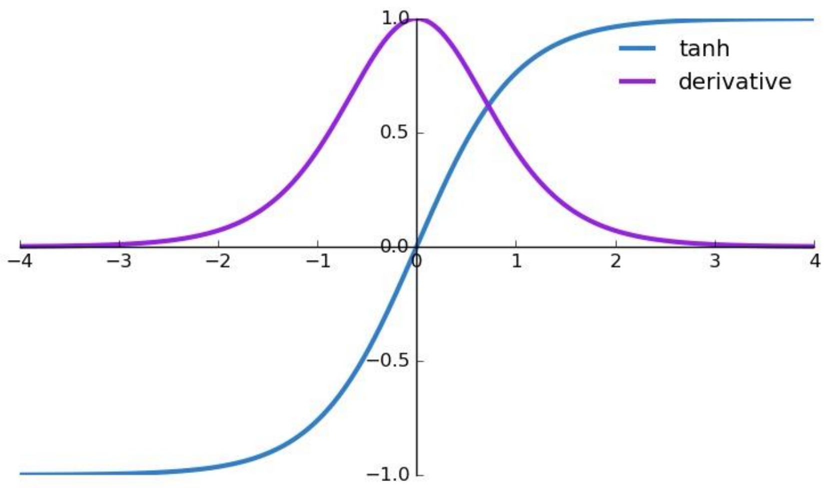
Image source

Vanilla RNN cell



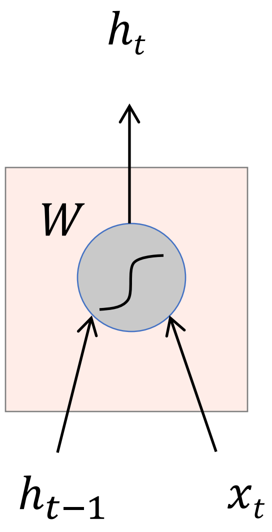
$$h_t = f_W(x_t, h_{t-1})$$

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

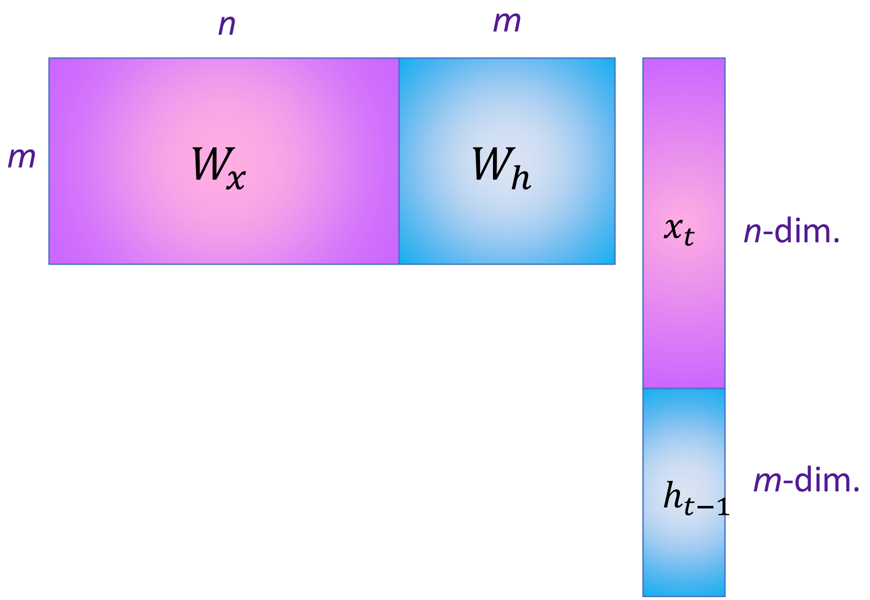


$$\frac{d}{da}\tanh(a) = 1 - \tanh^2(a)$$

Vanilla RNN cell

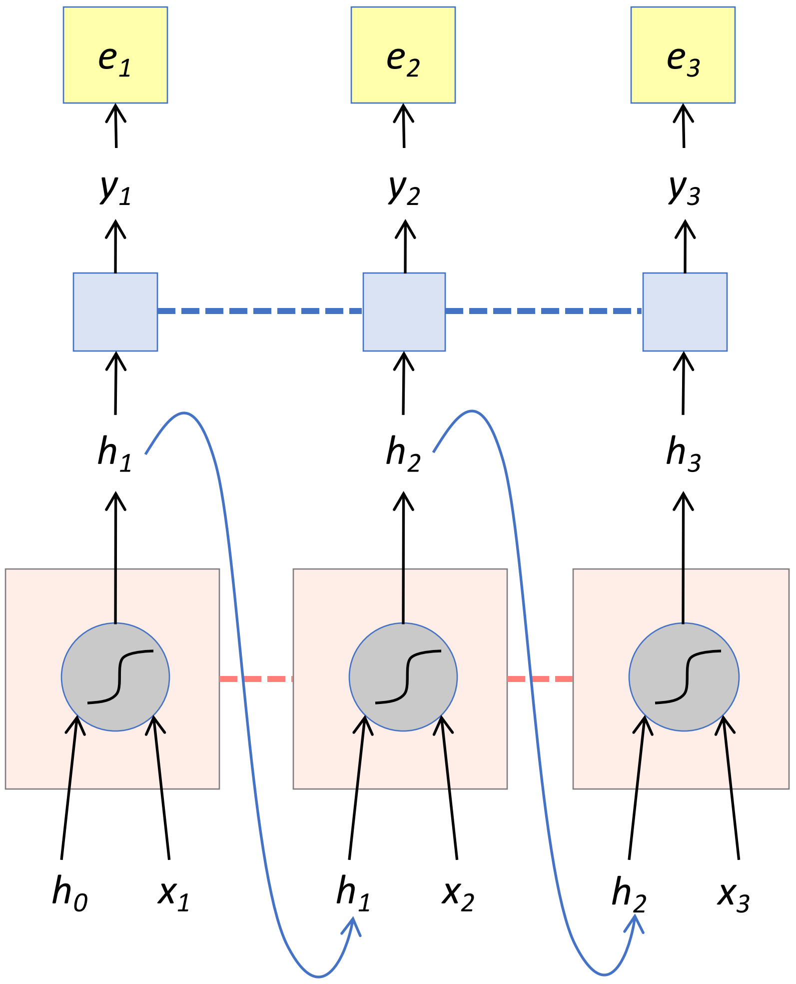


$$
\begin{aligned}
h_t &= f_W(x_t, h_{t-1}) \\
&= \tanh\left(W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}\right) \\
&= \tanh(W_x x_t + W_h h_{t-1})
\end{aligned}
$$



Slide credit: S. Lazebnik

RNN forward pass



$$e_t = -\log(y_t(GT_t))$$

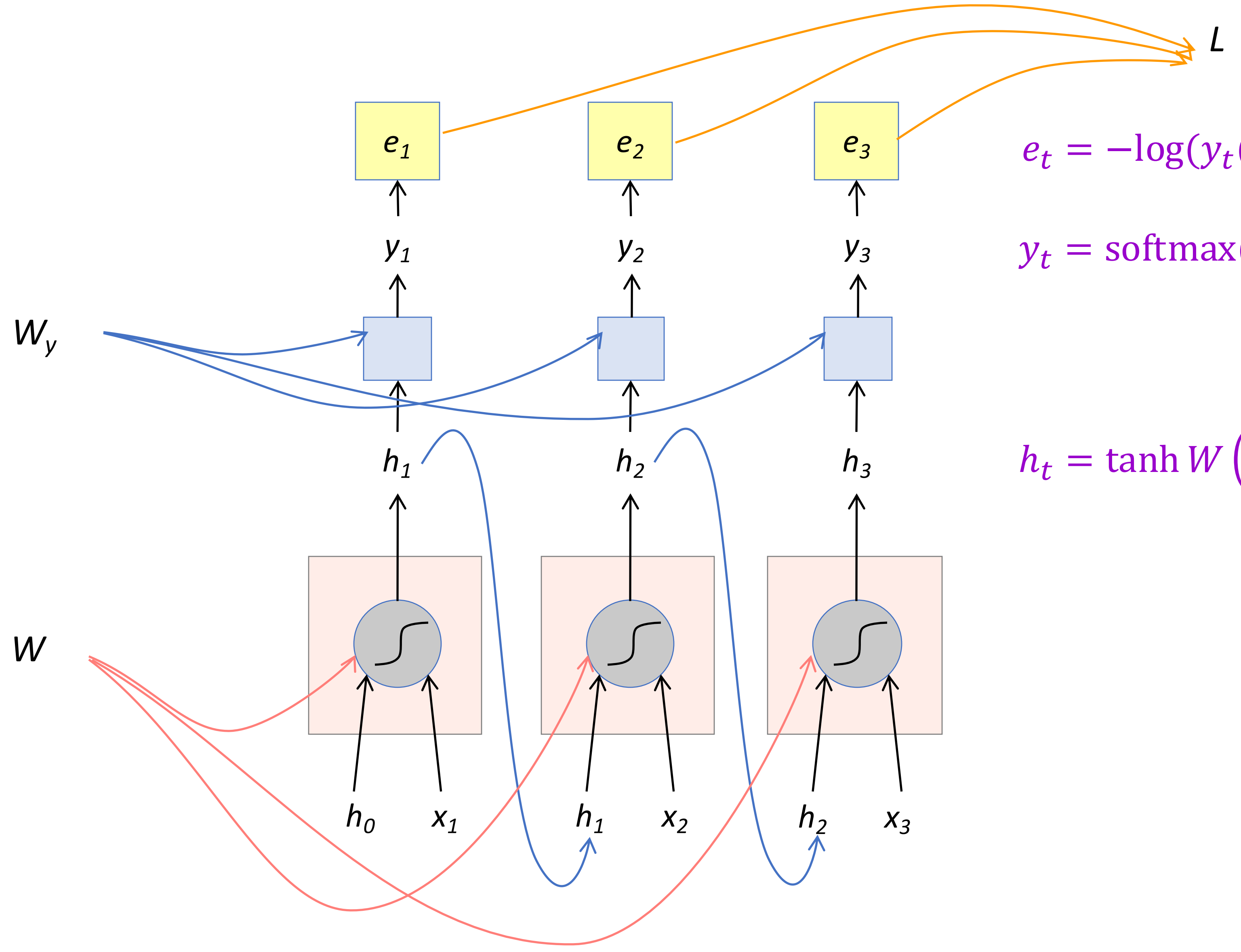
$$y_t = \mathrm{softmax}(W_y h_t)$$

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

------ shared weights

Slide credit: S. Lazebnik

RNN forward pass: Computation graph



$$e_t = -\log(y_t(GT_t))$$

$$y_t = \text{softmax}(W_y h_t)$$

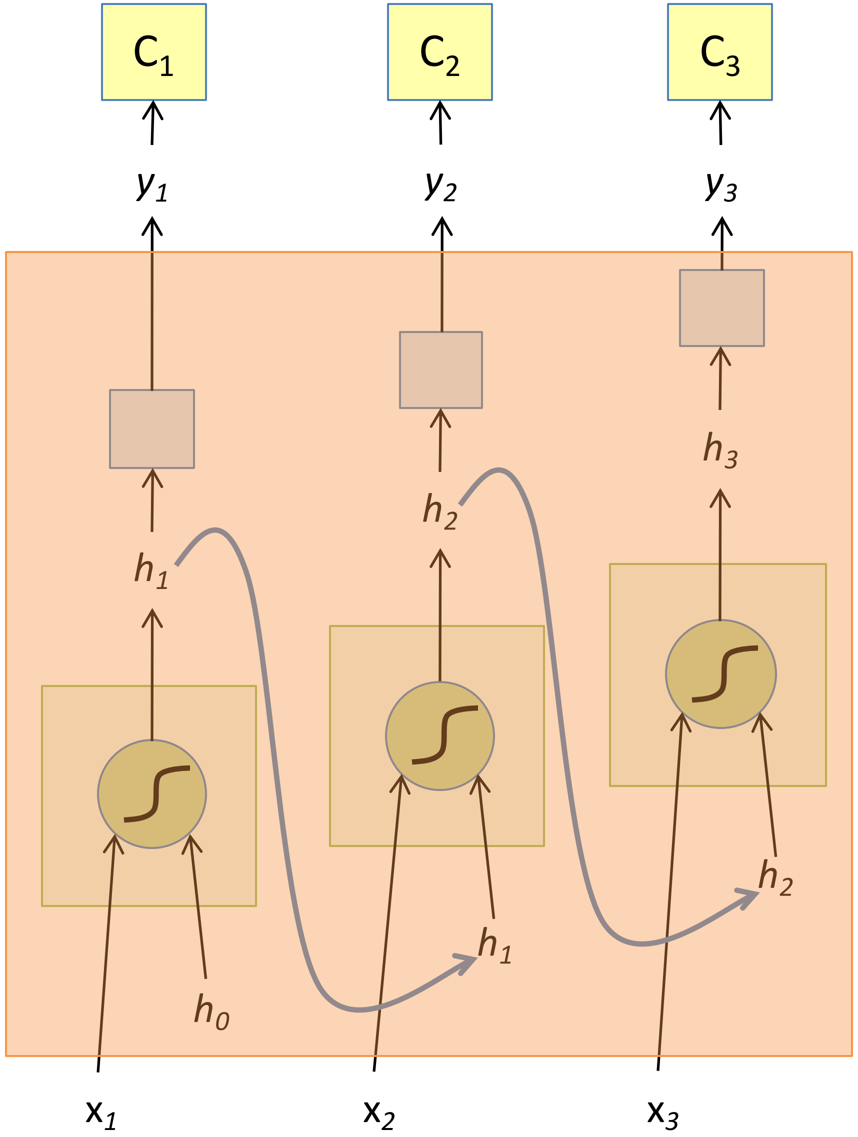
$$h_t = \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

Training: Backpropagation through time (BPTT)

- The unfolded network (used during forward pass) is treated as one big feed-forward network that accepts the whole time series as input
- The weight updates are computed for each copy in the unfolded network, then summed (or averaged) and applied to the RNN weights

Slide credit: S. Lazebnik

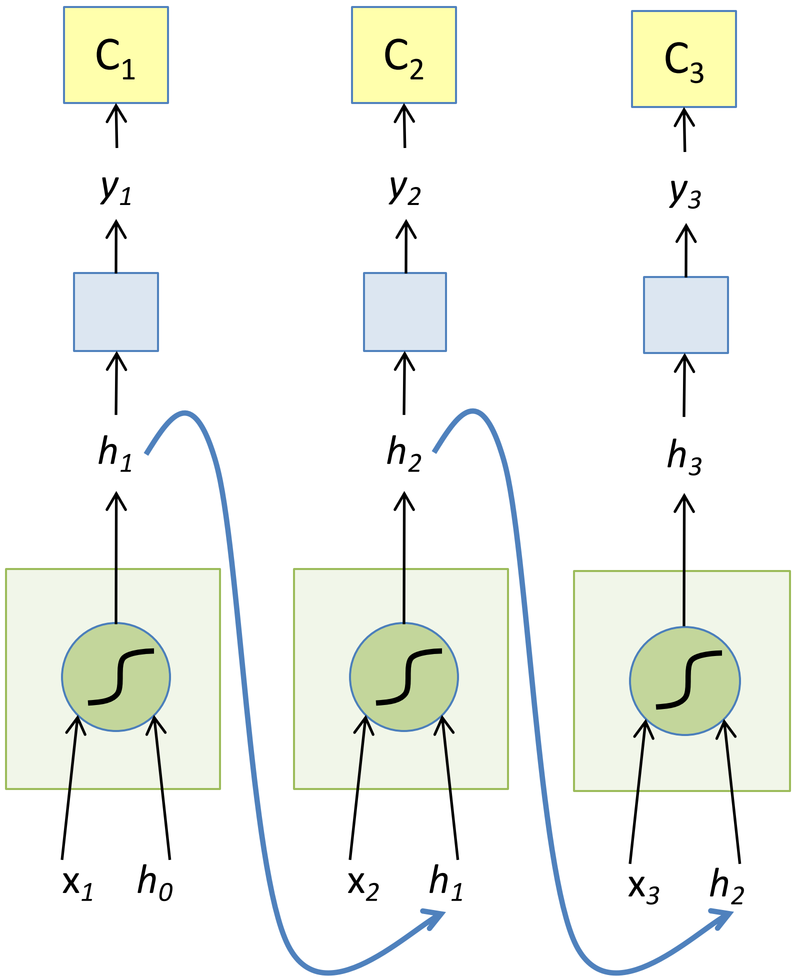
The Unfolded Vanilla RNN



- Treat the unfolded network as one big feed-forward network!
- This big network takes in entire sequence as an input
- Compute gradients through the usual backpropagation
- Update shared weights

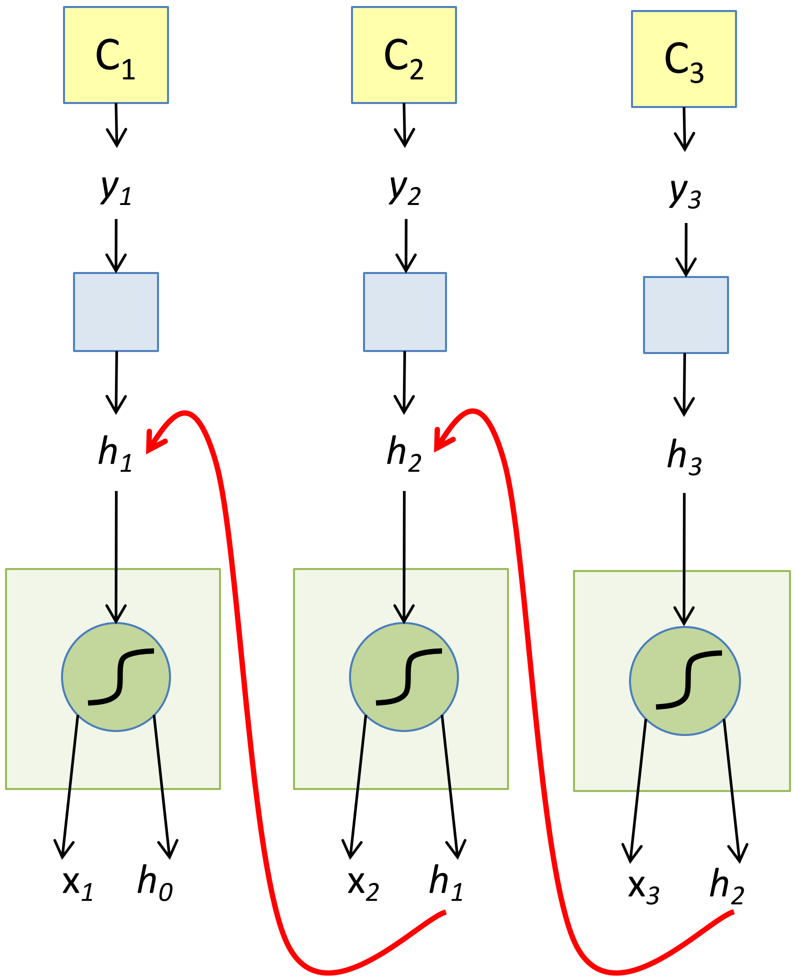
Slide credit: A. Mallya

The Unfolded Vanilla RNN Forward



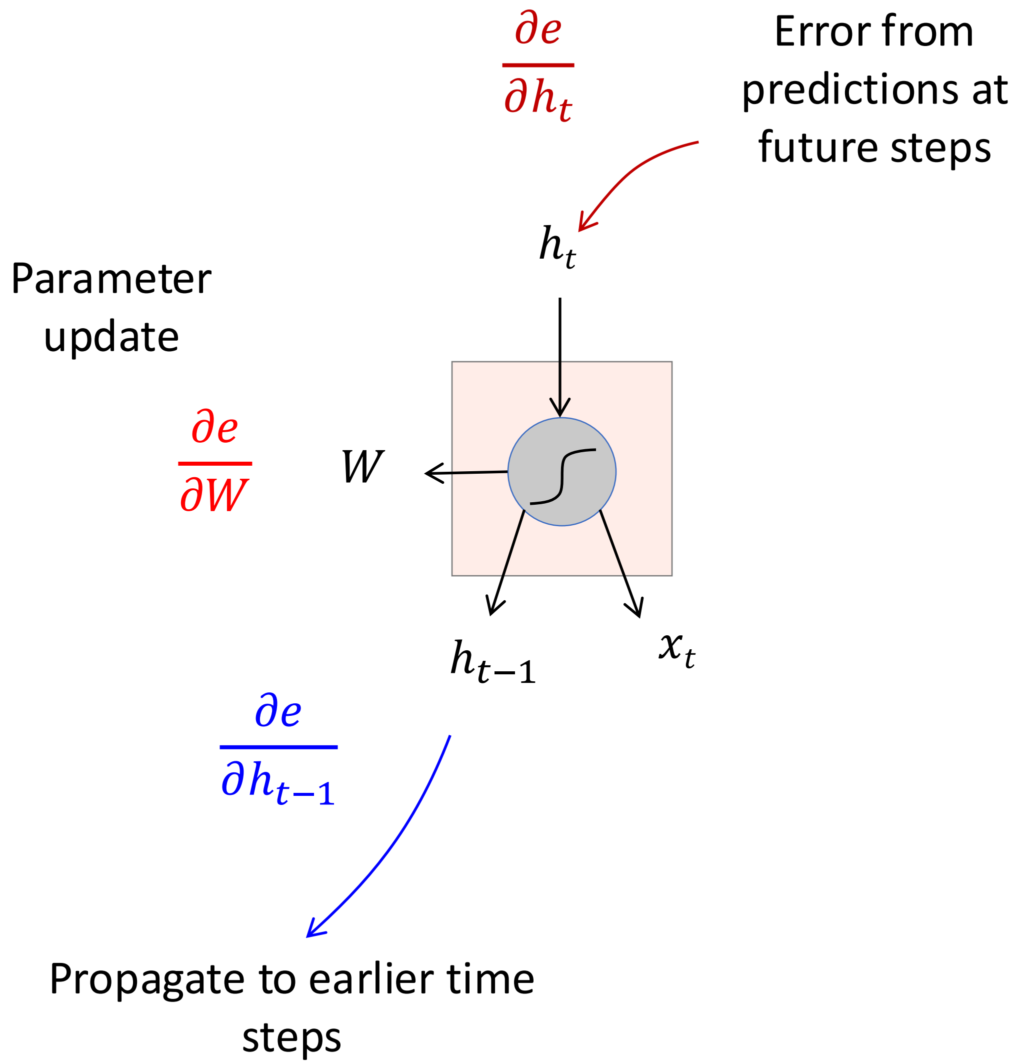
Slide credit: A. Mallya

The Unfolded Vanilla RNN Backward



Slide credit: A. Mallya

RNN backward pass



$$h_t = \tanh(W_x x_t + W_h h_{t-1})$$

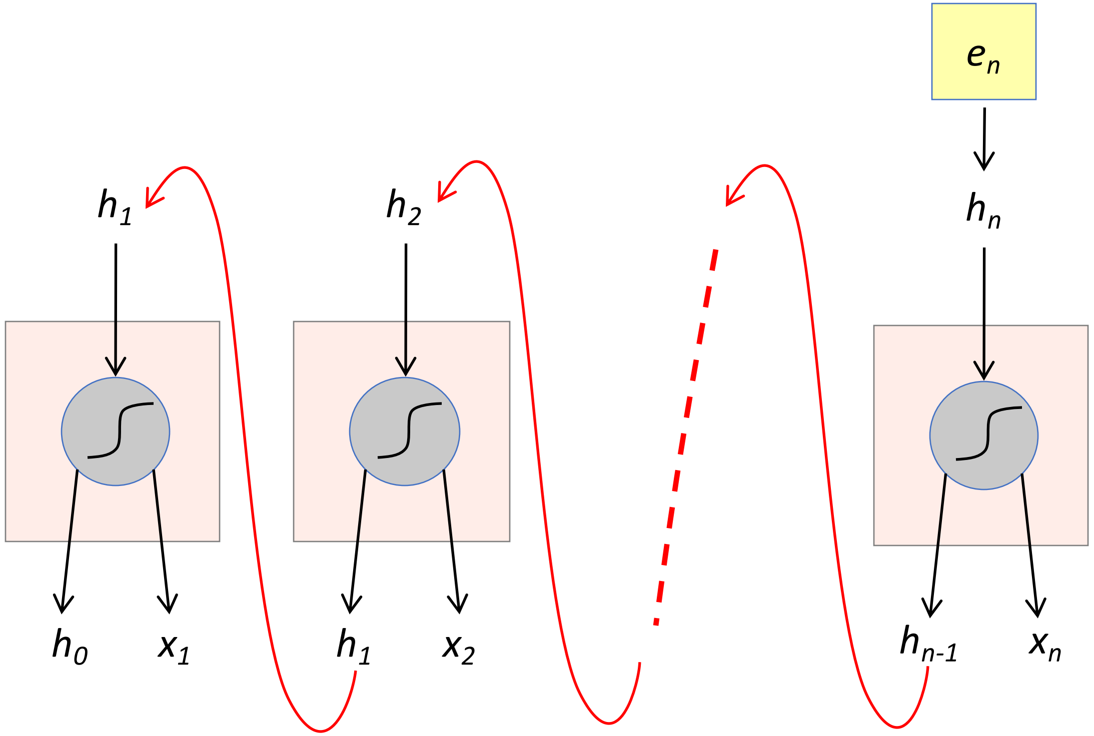
$$\frac{\partial e}{\partial W_h} = \frac{\partial e}{\partial h_t} \odot \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) h_{t-1}^T$$

$$\frac{\partial e}{\partial W_x} = \frac{\partial e}{\partial h_t} \odot \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) x_t^T$$

$$\frac{\partial e}{\partial h_{t-1}} = W_h^T \left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

Slide credit: S. Lazebnik

Vanishing and exploding gradients



$$\frac{\partial e}{\partial h_{t-1}} = W_h^T\left(1 - \tanh^2(W_x x_t + W_h h_{t-1})\right) \odot \frac{\partial e}{\partial h_t}$$

Computing gradient for h_0
involves many multiplications by W_h^T
(and rescalings between 0 and 1)

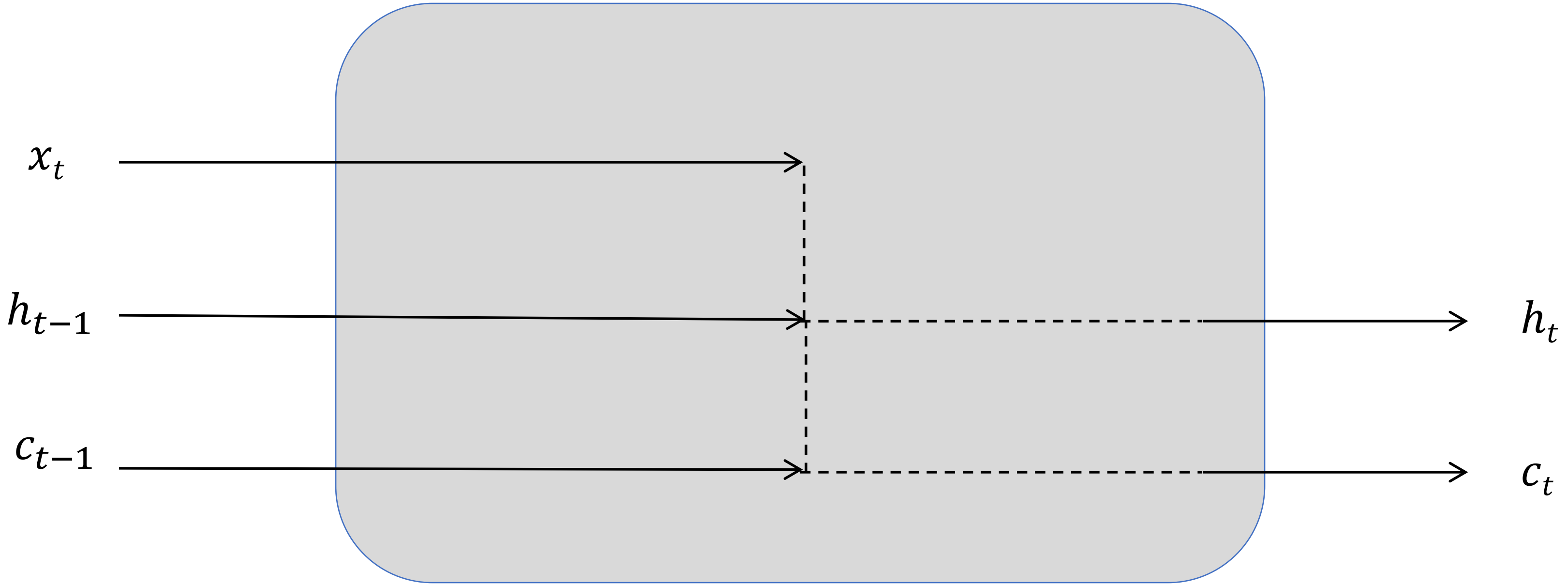
Gradients will *vanish* if largest singular value of W_h is less than 1
and *explode* if it's greater than 1

https://notesonai.com/vanishing+and+exploding+gradients

Slide credit: S. Lazebnik

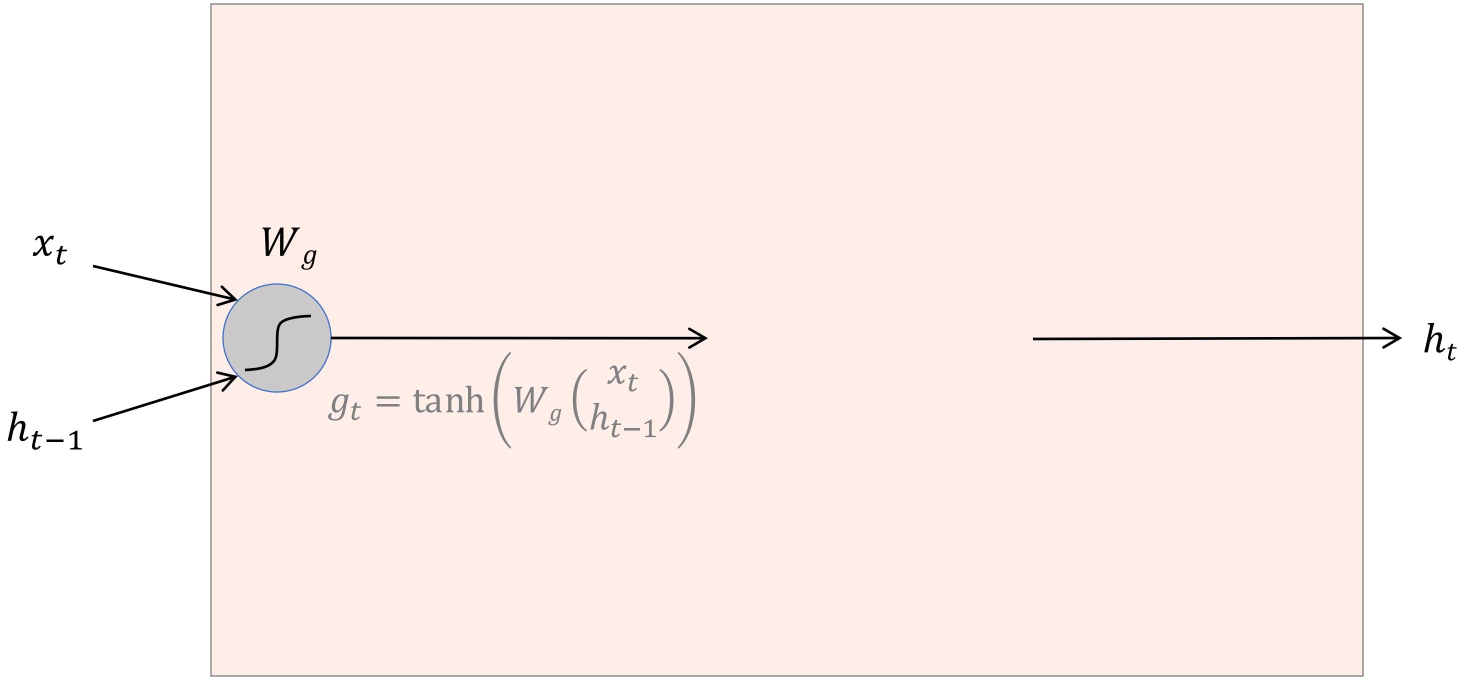
Long short-term memory (LSTM)

- Add a *memory cell* that is not subject to matrix multiplication or squishing, thereby avoiding gradient decay



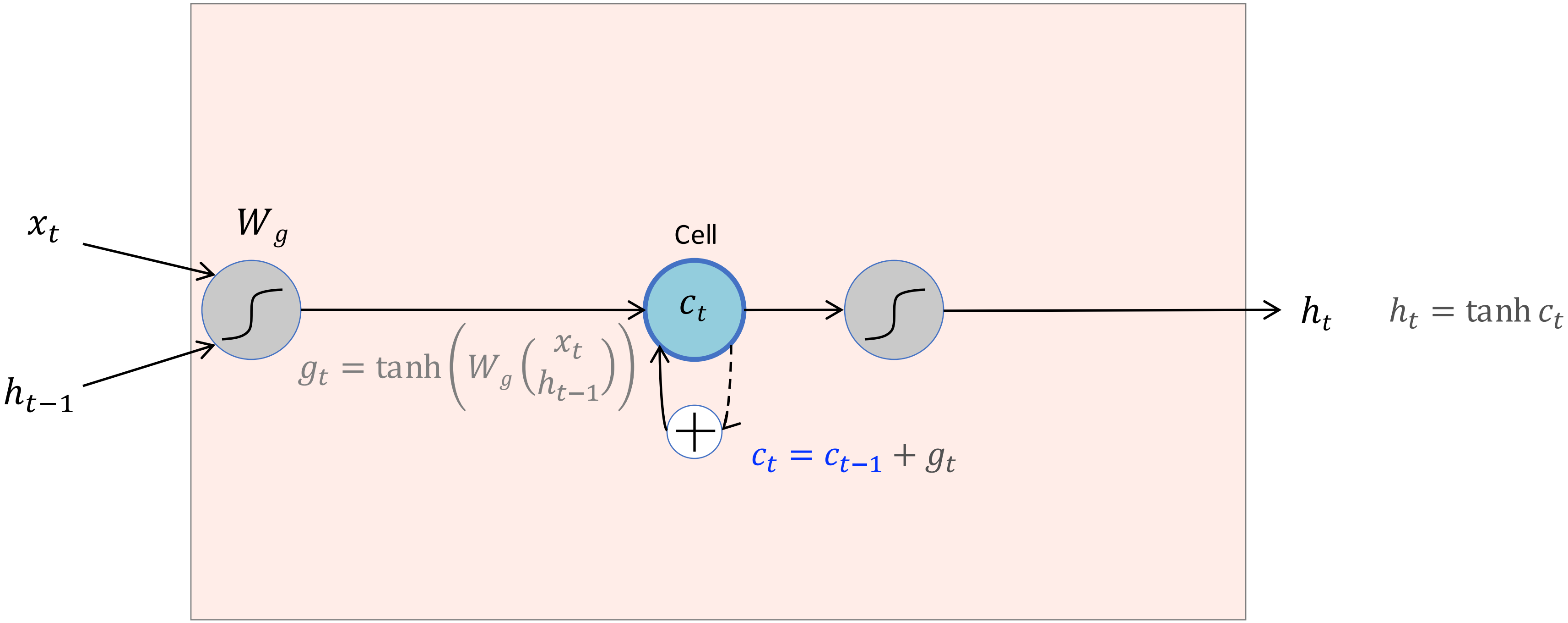
S. Hochreiter and J. Schmidhuber, Long short-term memory, Neural Computation 9 (8), pp. 1735–1780, 1997

The LSTM cell



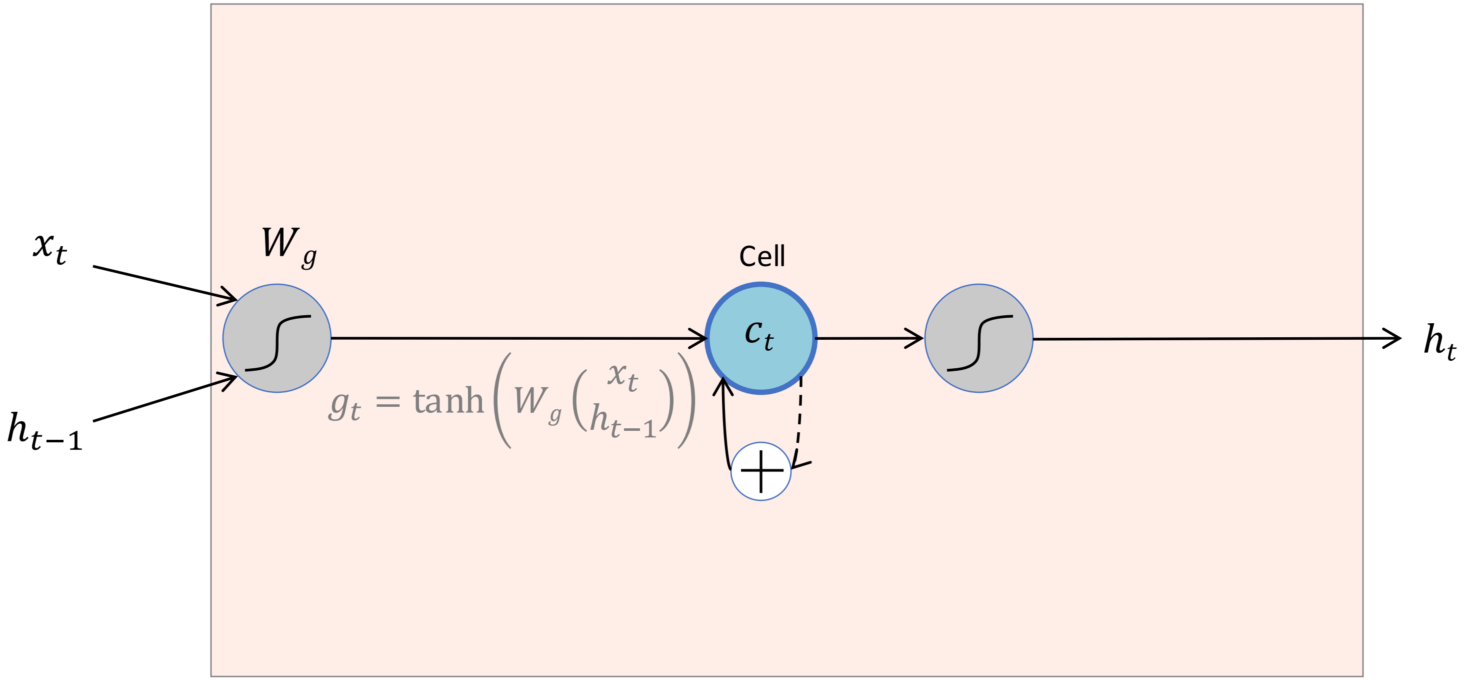
Slide credit: S. Lazebnik

The LSTM cell



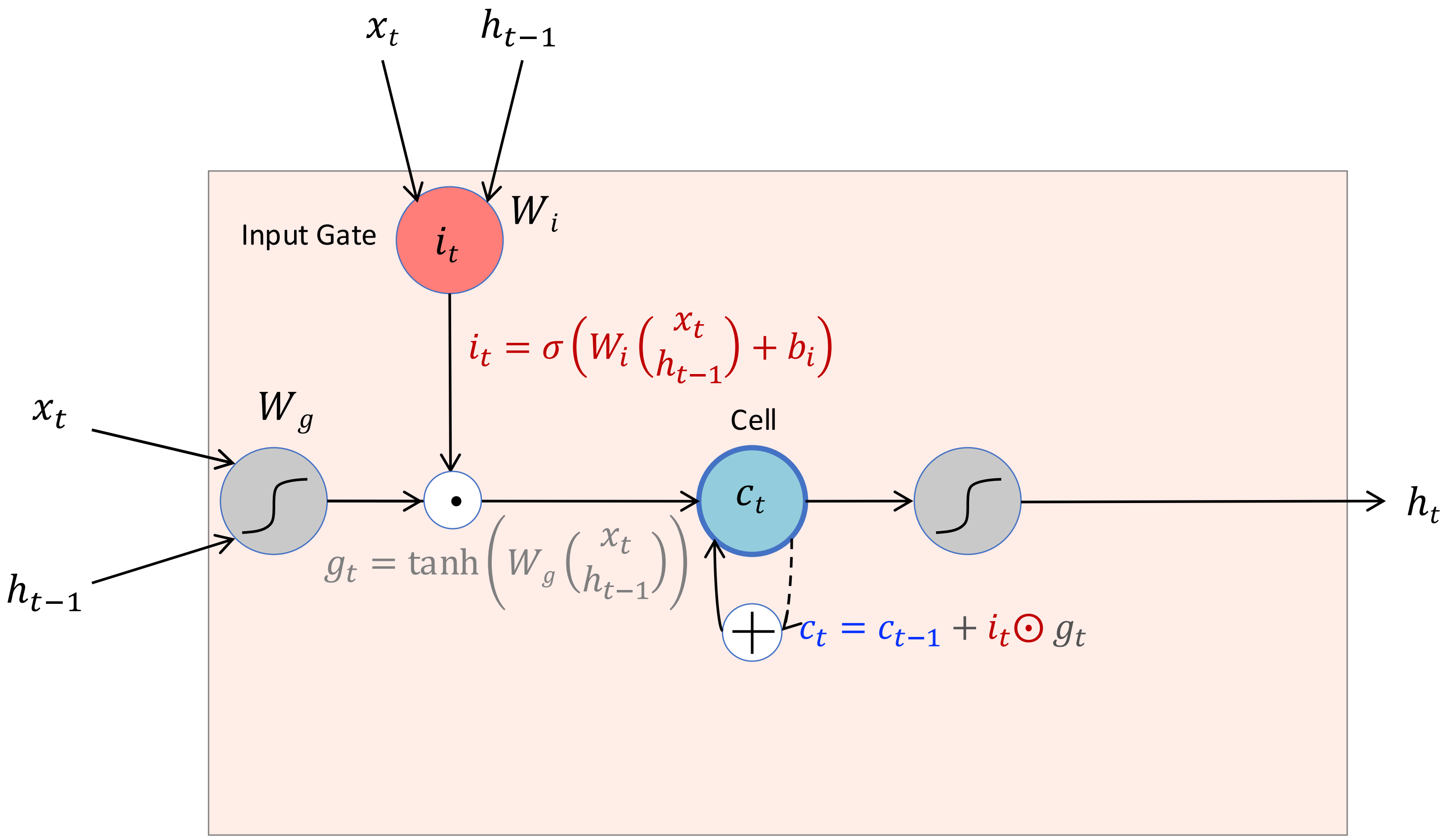
Slide credit: S. Lazebnik

The LSTM cell



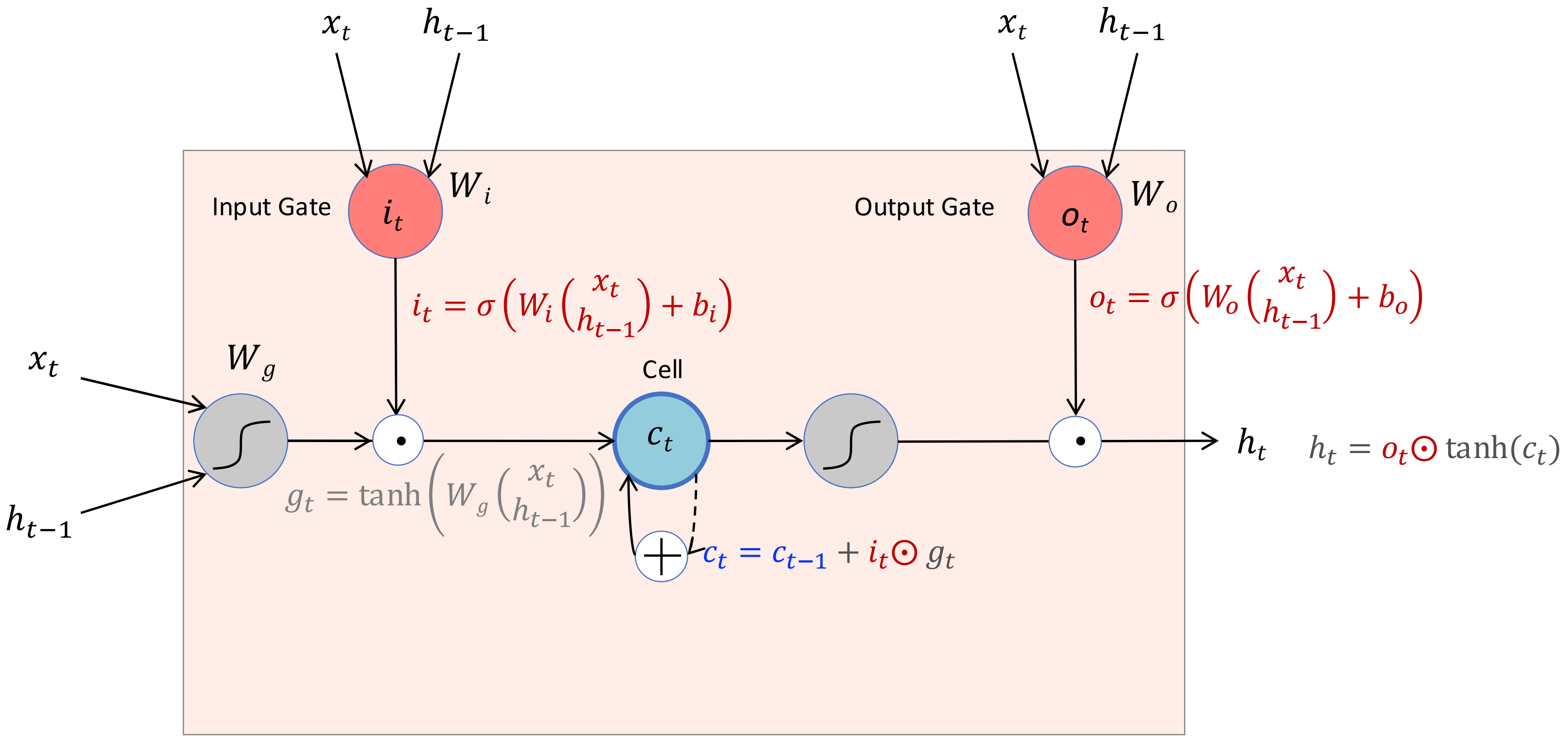
Slide credit: S. Lazebnik

The LSTM cell



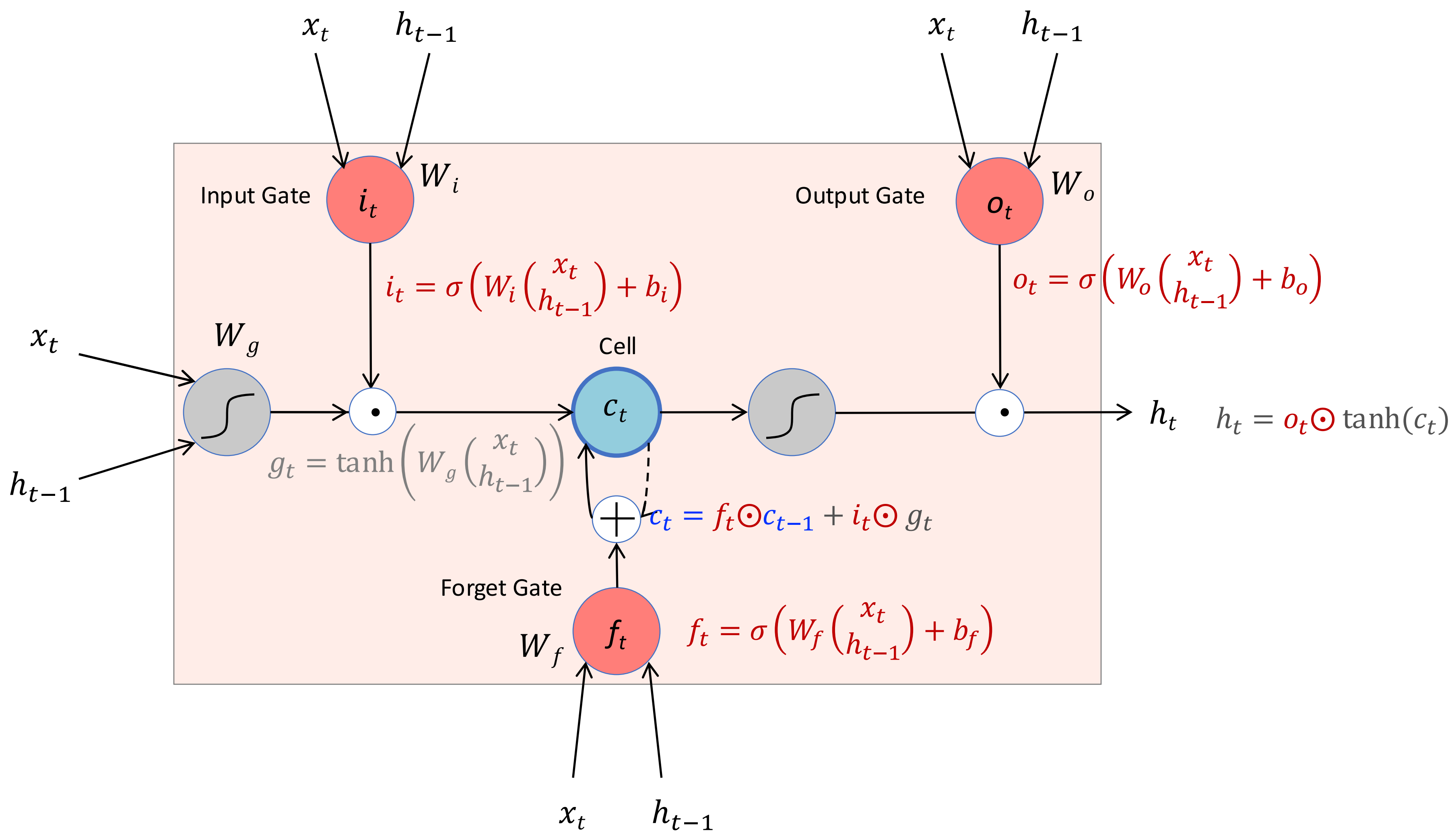
Slide credit: S. Lazebnik

The LSTM cell



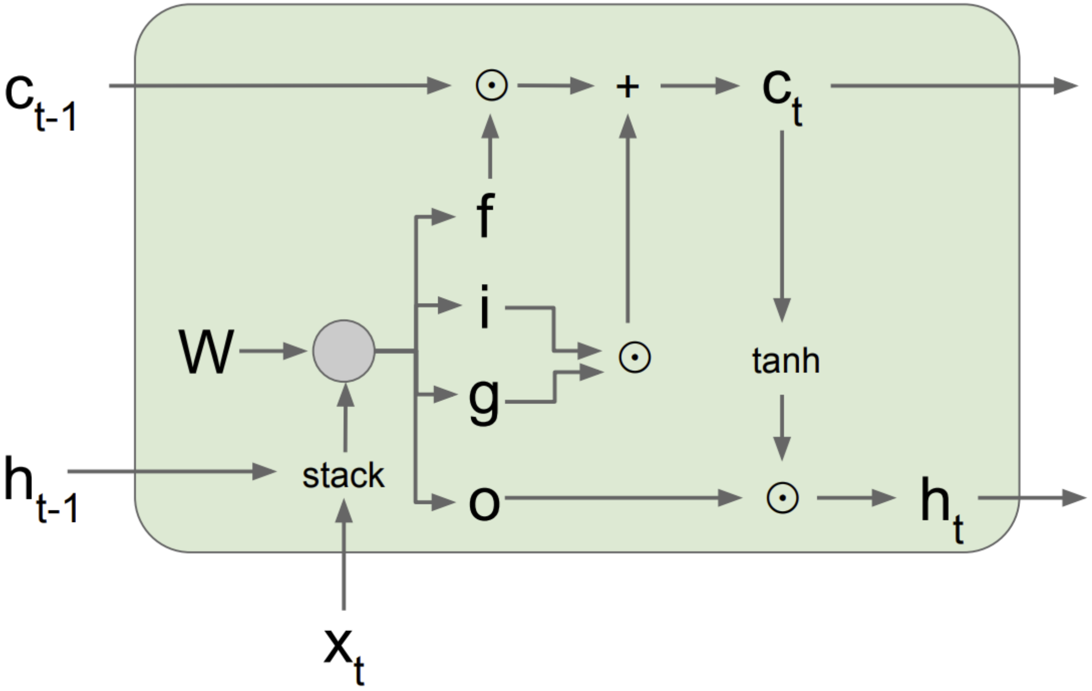
Slide credit: S. Lazebnik

The LSTM cell



Slide credit: S. Lazebnik

LSTM forward pass summary



$$\begin{pmatrix} g_t \\ i_t \\ f_t \\ o_t \end{pmatrix} = \begin{pmatrix} \tanh \\ \sigma \\ \sigma \\ \sigma \end{pmatrix} \begin{pmatrix} W_g \\ W_i \\ W_f \\ W_o \end{pmatrix} \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

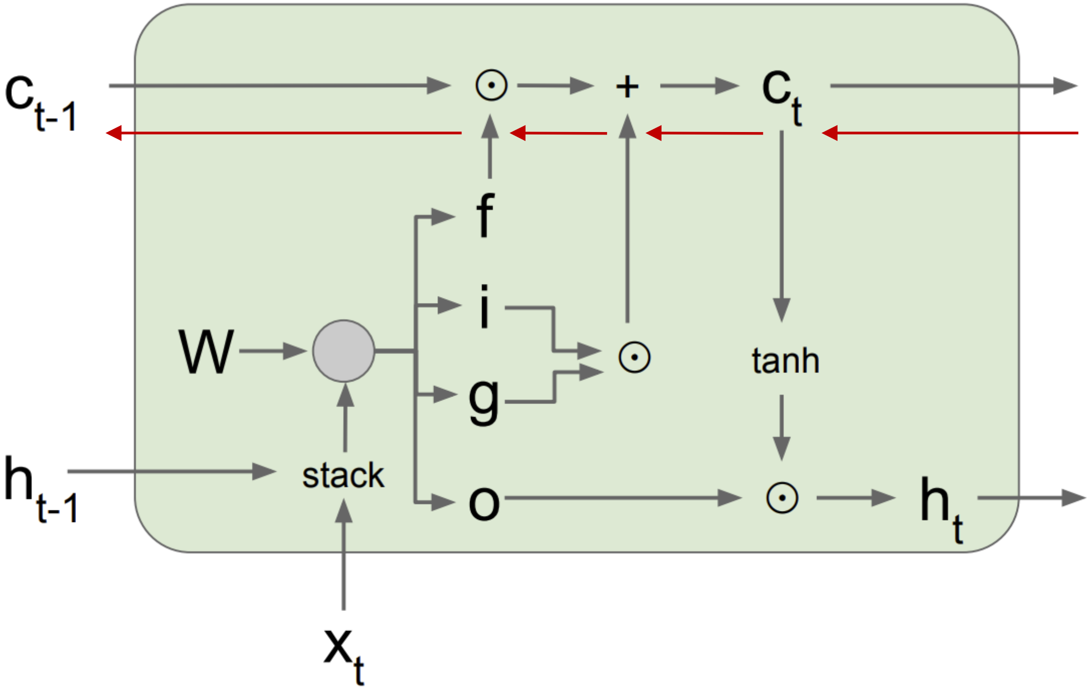
$$c_t = f_t \odot c_{t-1} + i_t \odot g_t$$

$$h_t = o_t \odot \tanh c_t$$

Slide credit: S. Lazebnik

Figure source

LSTM backward pass



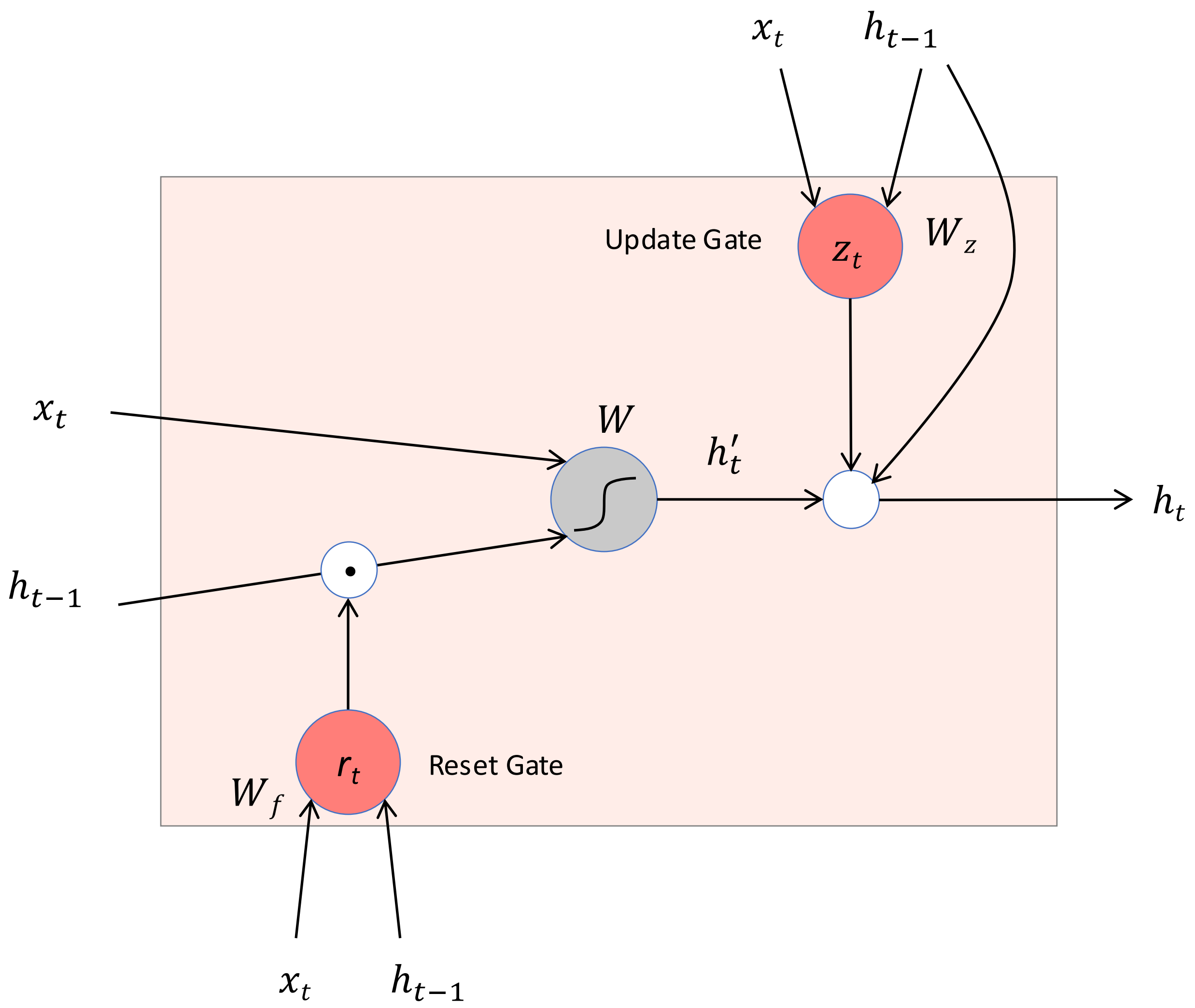
Gradient flow from c_t to c_{t-1} only involves back-propagating through addition and elementwise multiplication, not matrix multiplication or tanh

For complete details: [Illustrated LSTM Forward and Backward Pass]

[Figure source]

Slide credit: S. Lazebnik

Gated recurrent unit (GRU)

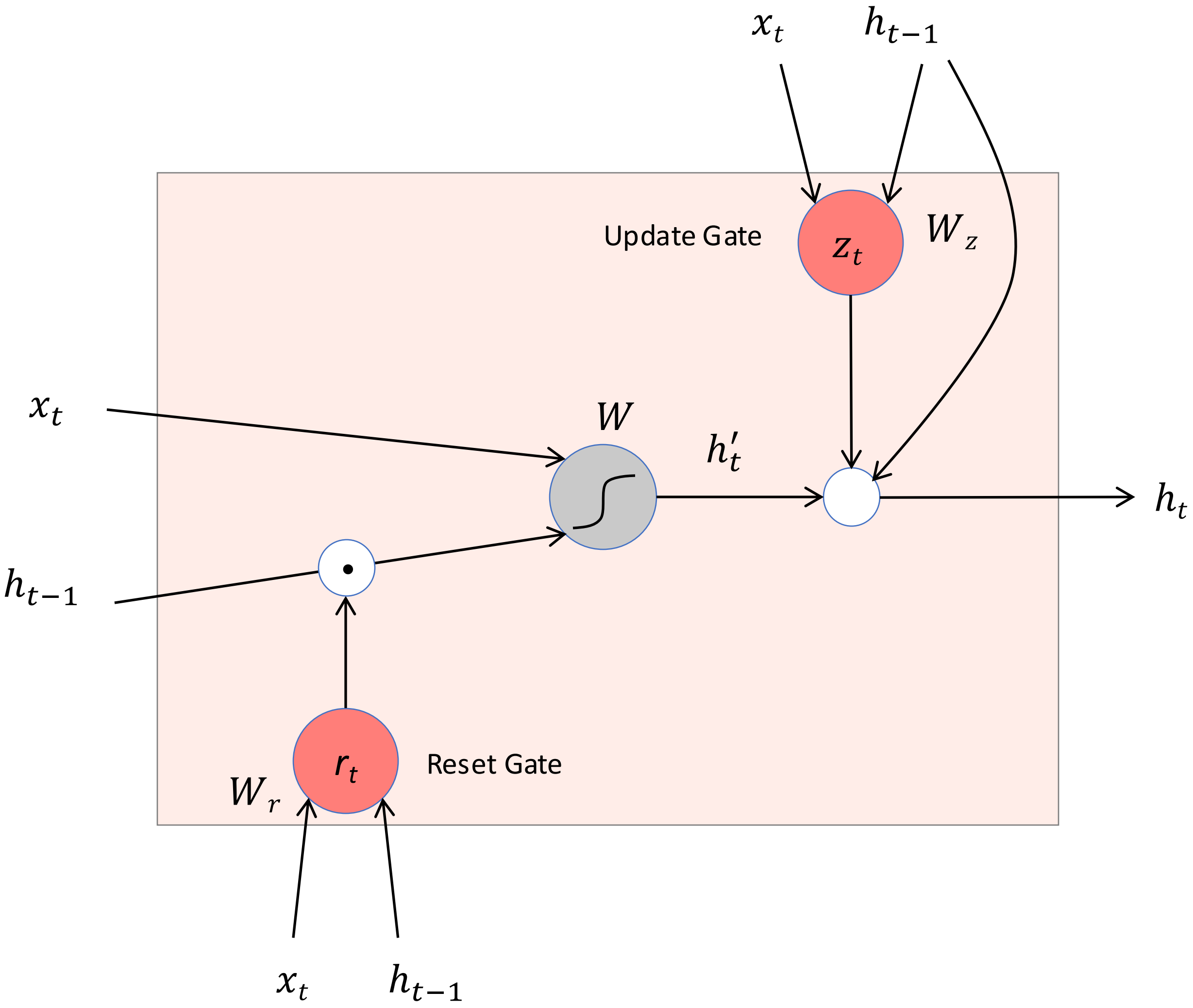


- Get rid of separate cell state
- Merge "forget" and "output" gates into "update" gate

K. Cho et al., Learning phrase representations using RNN encoder-decoder for statistical machine translation, ACL 2014

Slide credit: S. Lazebnik

Gated recurrent unit (GRU)



$$r_t = \sigma\left(W_r \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_t\right)$$

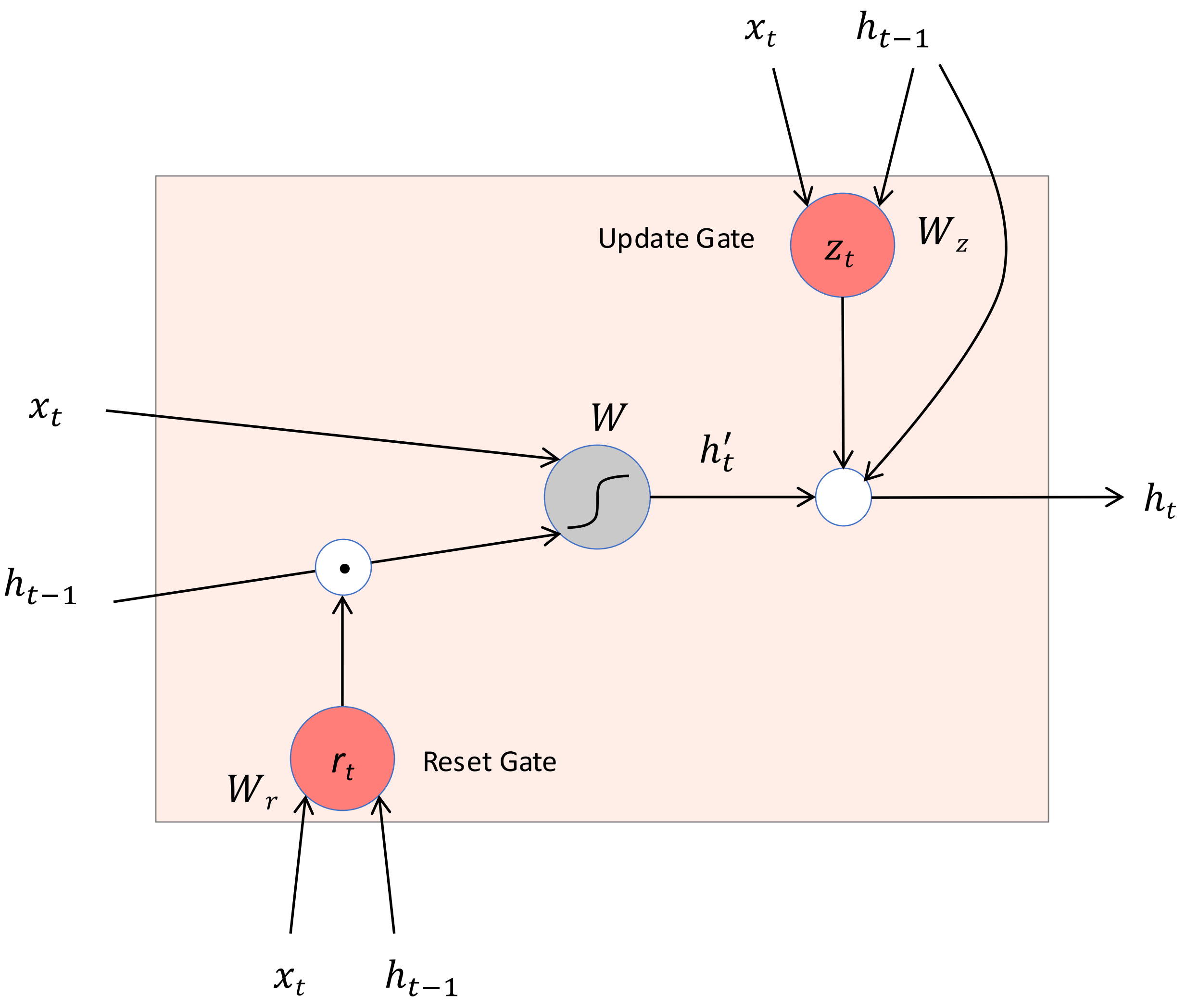
$$h'_t = \tanh\left(W \begin{pmatrix} x_t \\ r_t \odot h_{t-1} \end{pmatrix}\right)$$

$$z_t = \sigma\left(W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_z\right)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h'_t$$

Slide credit: S. Lazebnik

ConvLSTM/ConvGRU



convolution

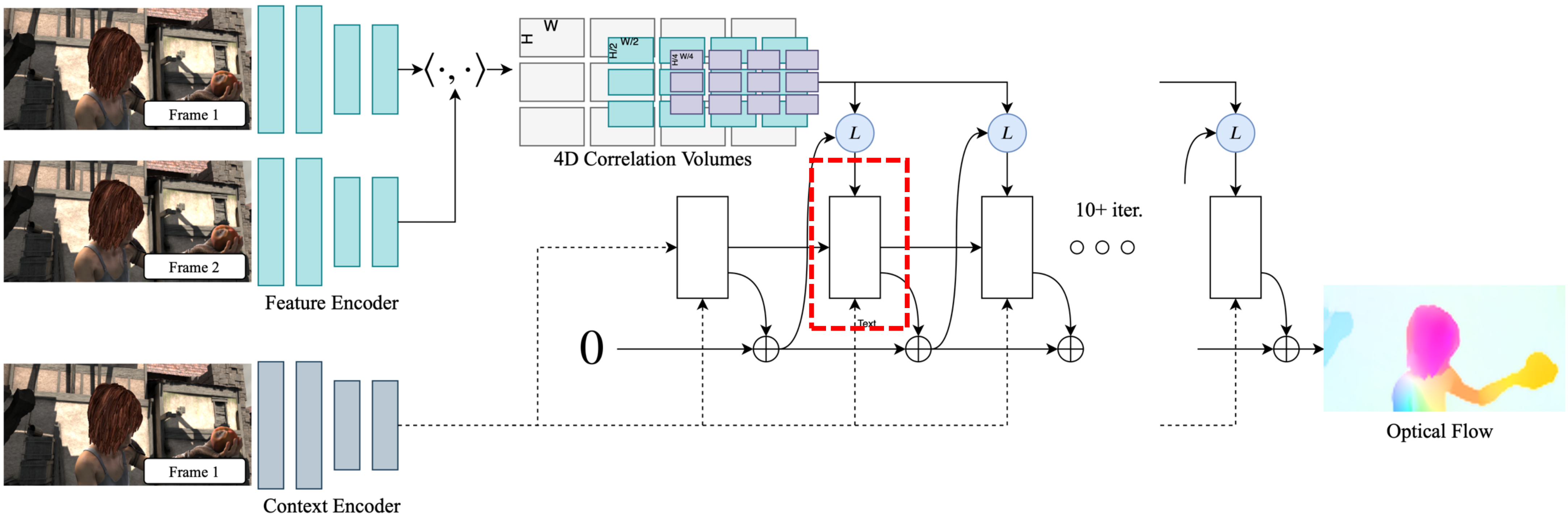
$$r_t = \sigma\left(W_r * \binom{x_t}{h_{t-1}} + b_t\right)$$

$$h_t' = \tanh\left(W * \binom{x_t}{r_t \odot h_{t-1}} \right)$$

$$z_t = \sigma\left(W_z * \binom{x_t}{h_{t-1}} + b_z\right)$$

$$h_t = (1 - z_t) \odot h_{t-1} + z_t \odot h_t'$$

ConvLSTM/ConvGRU



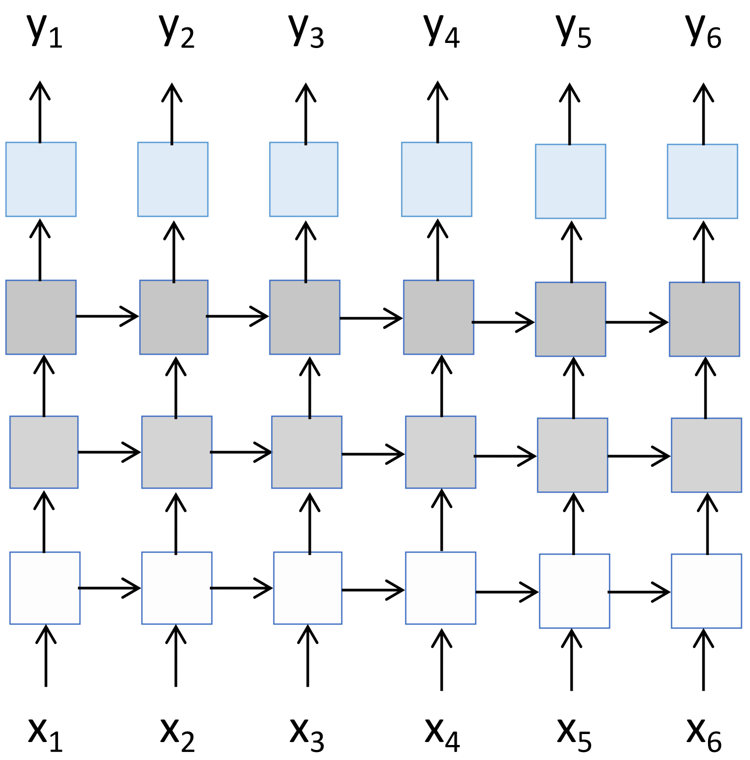
[Teed and Deng. ECCV 2020. Best paper award]

RNN, LSTM, and GRU in Practice

- Always use LSTM or GRU
  - No vanilla/raw RNN
  - Unless for educational purposes ☺
- LSTM vs GRU
  - Sometimes LSTM works well, sometimes GRU works well
  - GRU is simpler, less parameters, slightly faster

Multi-layer RNNs

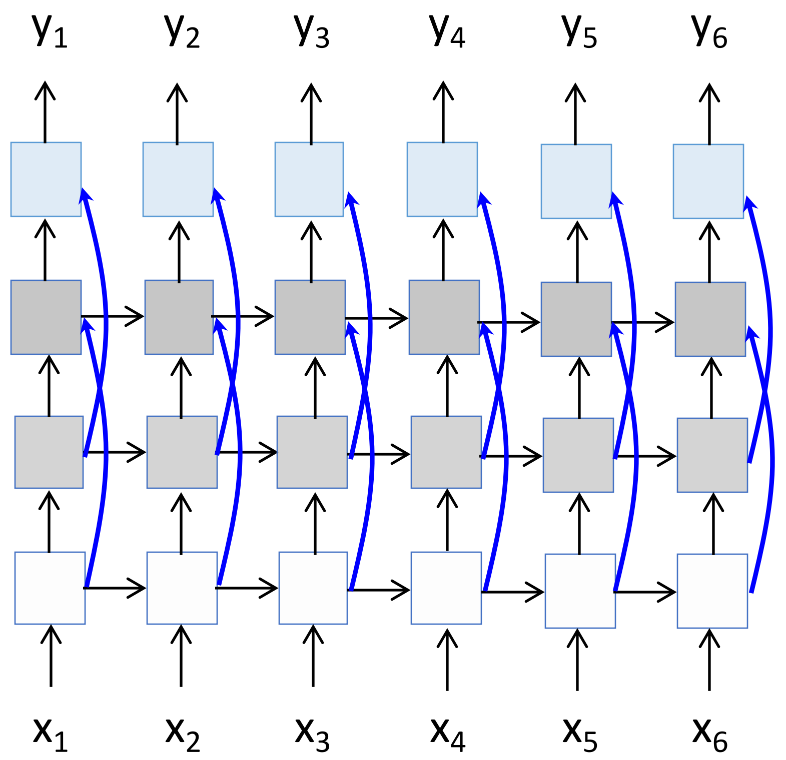
- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …

Multi-layer RNNs

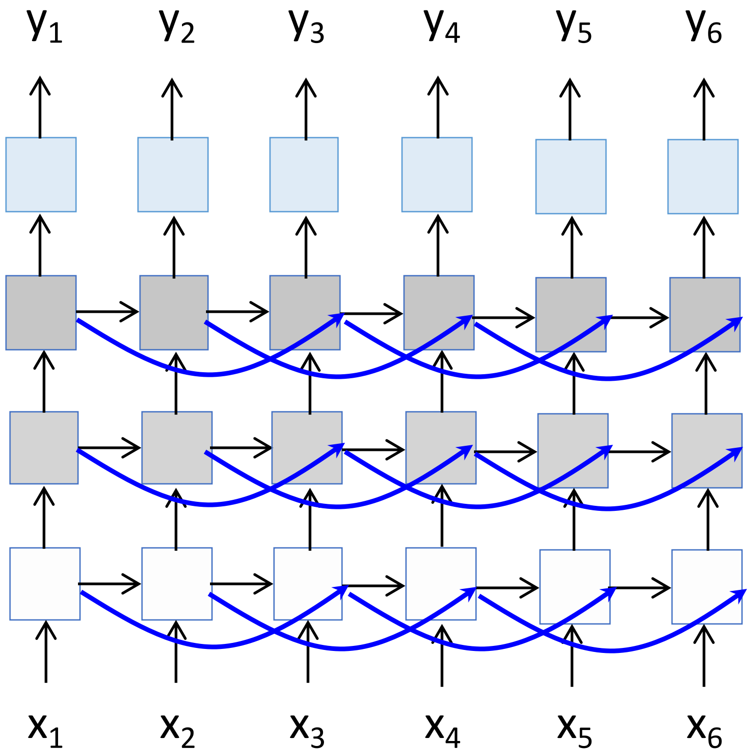
- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …

Multi-layer RNNs

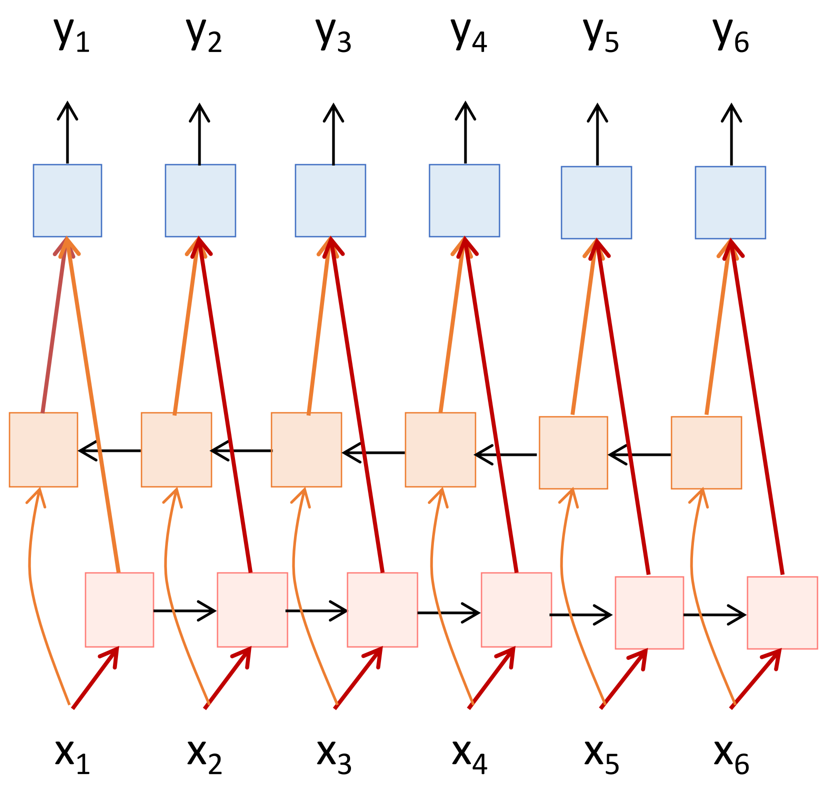
- We can of course design RNNs with multiple hidden layers



- Anything goes: skip connections across layers, across time, …

Bi-directional RNNs

- RNNs can process the input sequence in forward and in the reverse direction (common in speech recognition)



Sequence classification

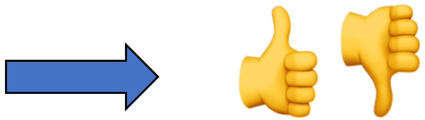
"The food is usually not so good"

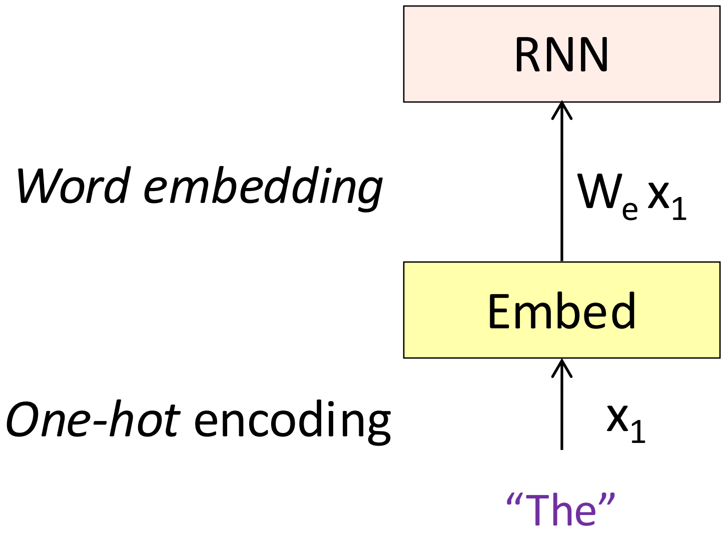
One-hot encoding x_1

"The"

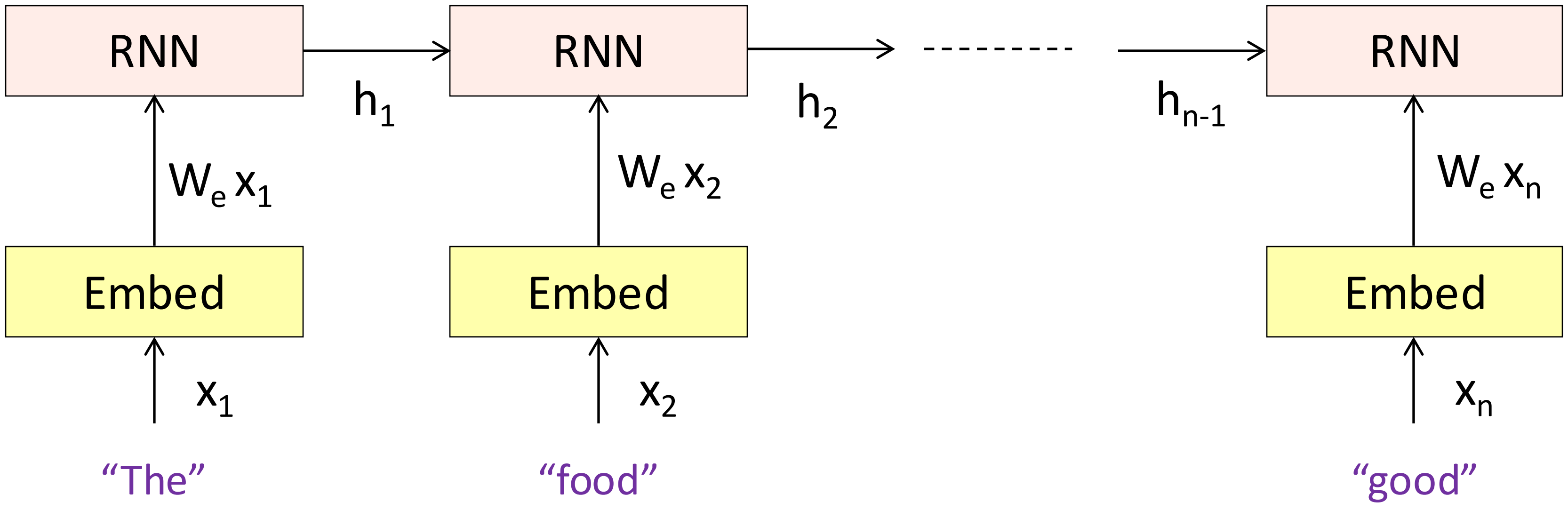
Sequence classification

"The food is usually not so good"

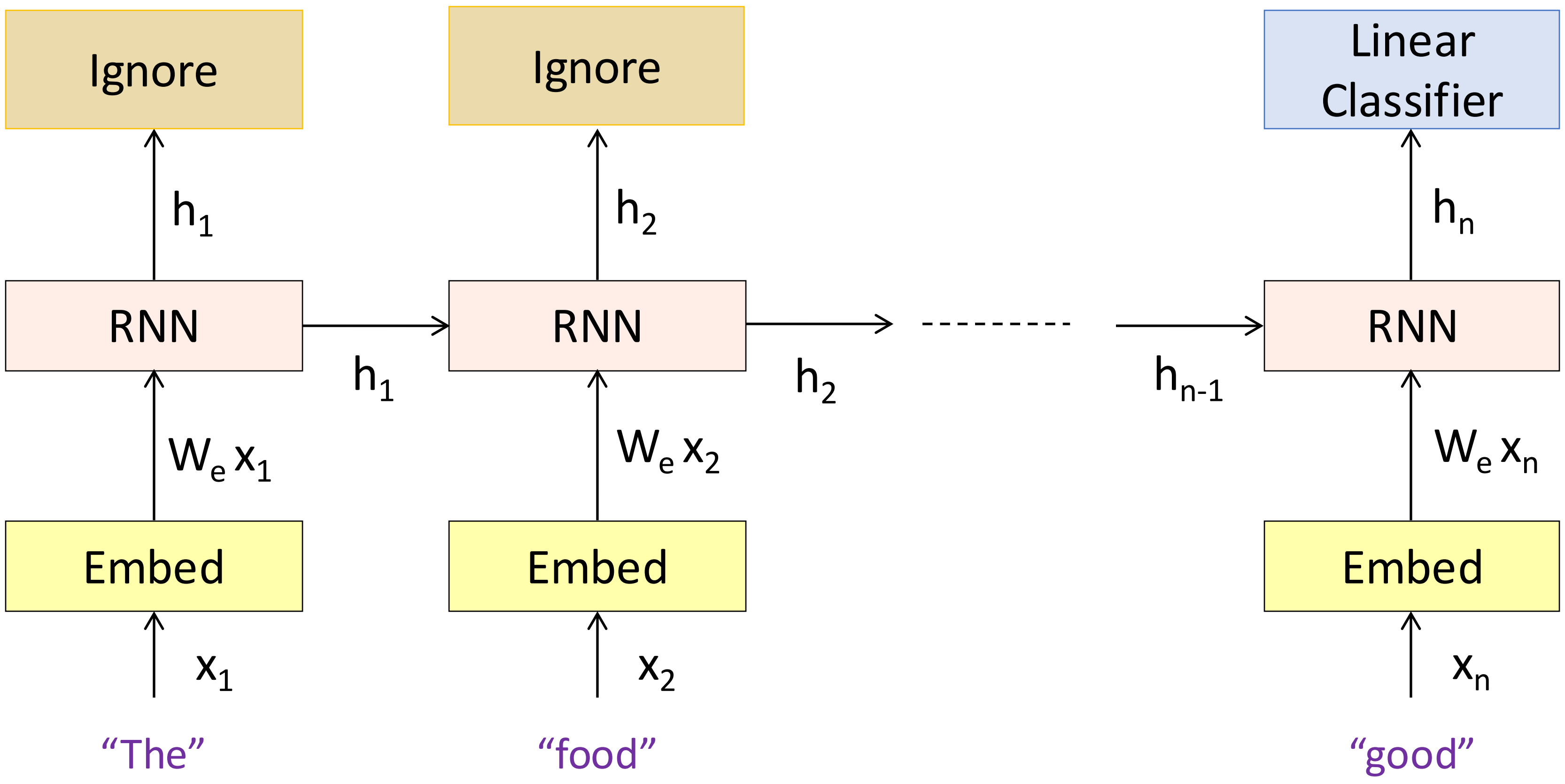




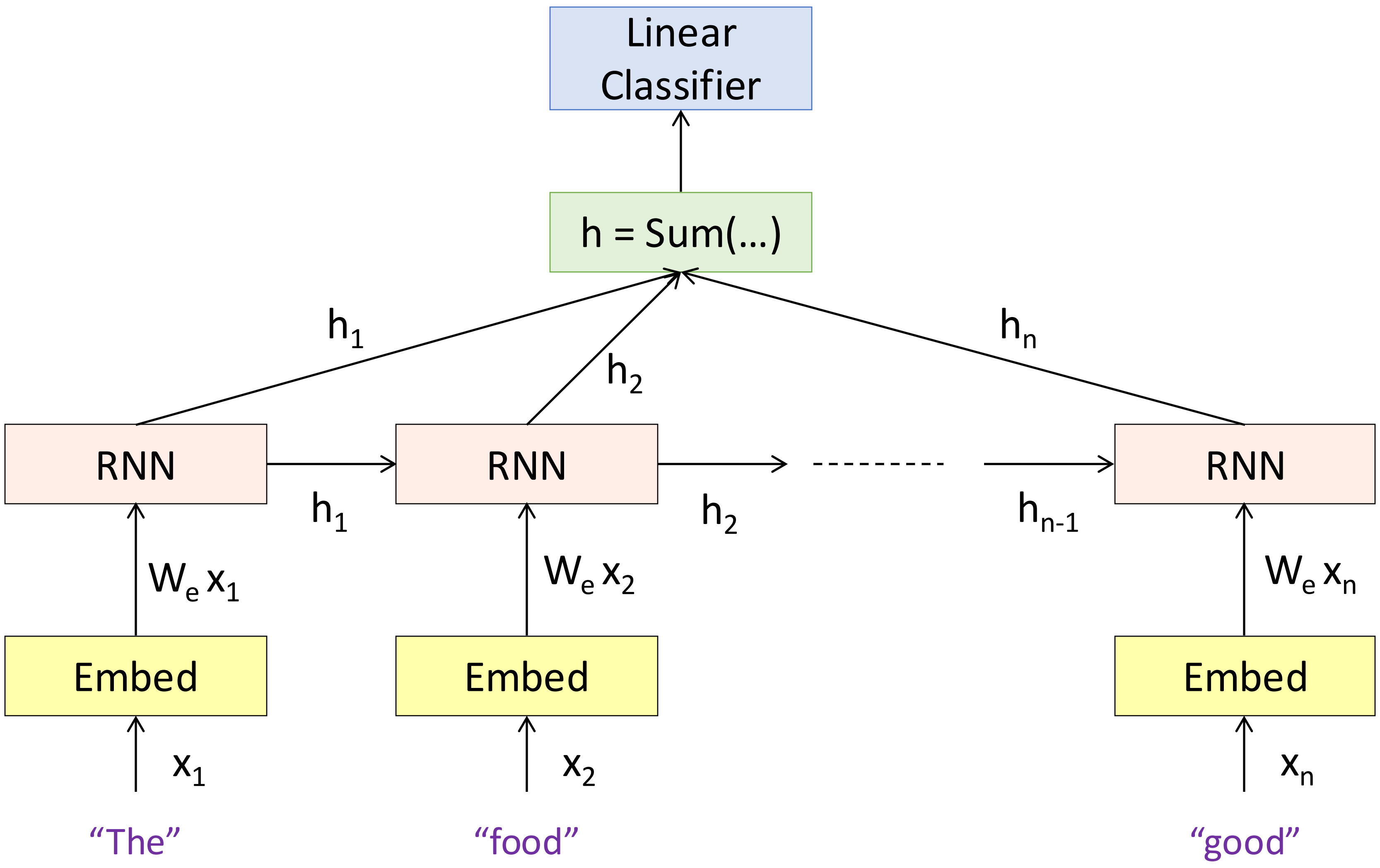
Sequence classification



Sequence classification

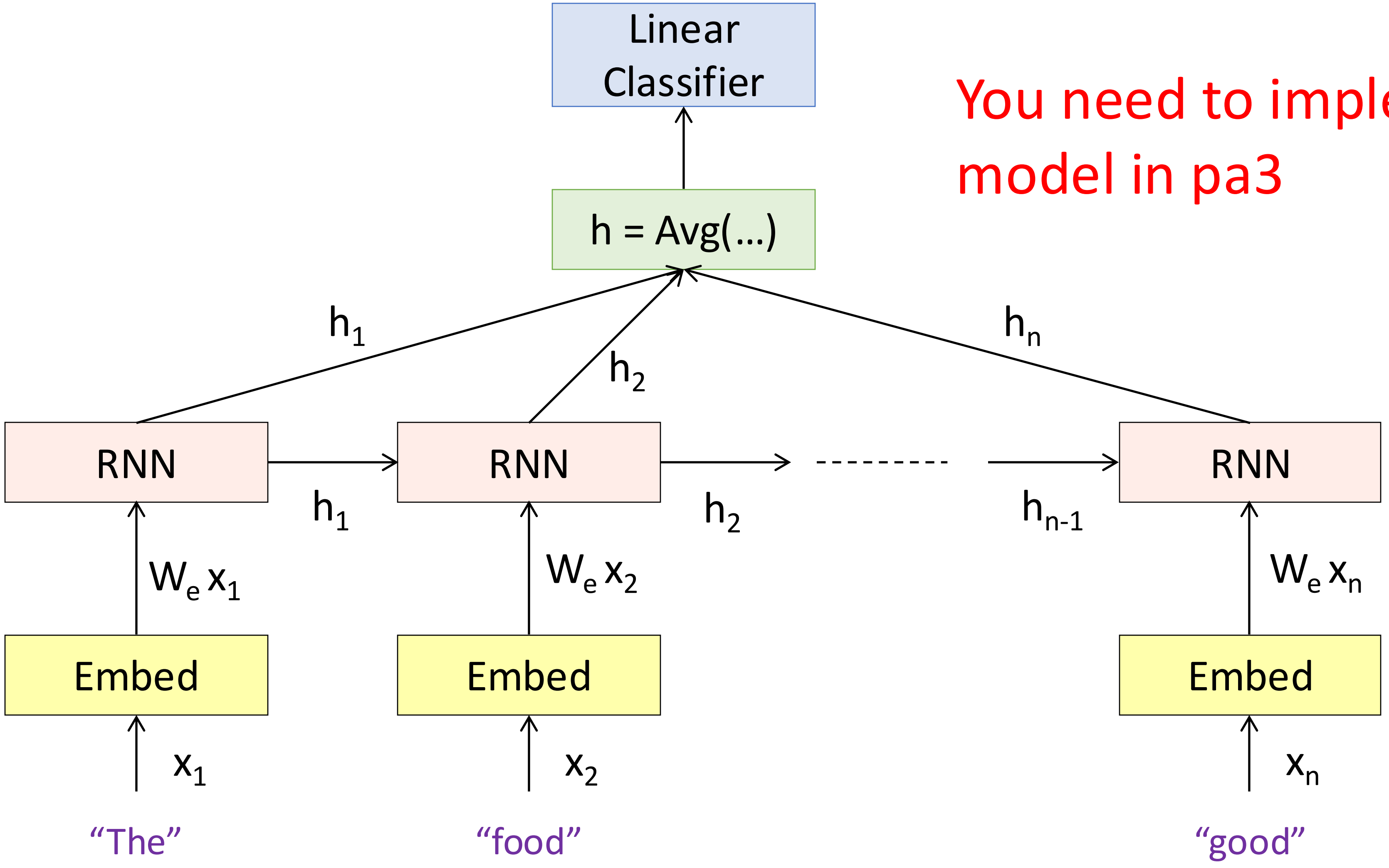


Sequence classification



http://deeplearning.net/tutorial/lstm.html

Sequence classification



You need to implement this model in pa3

http://deeplearning.net/tutorial/lstm.html

Sequence classification

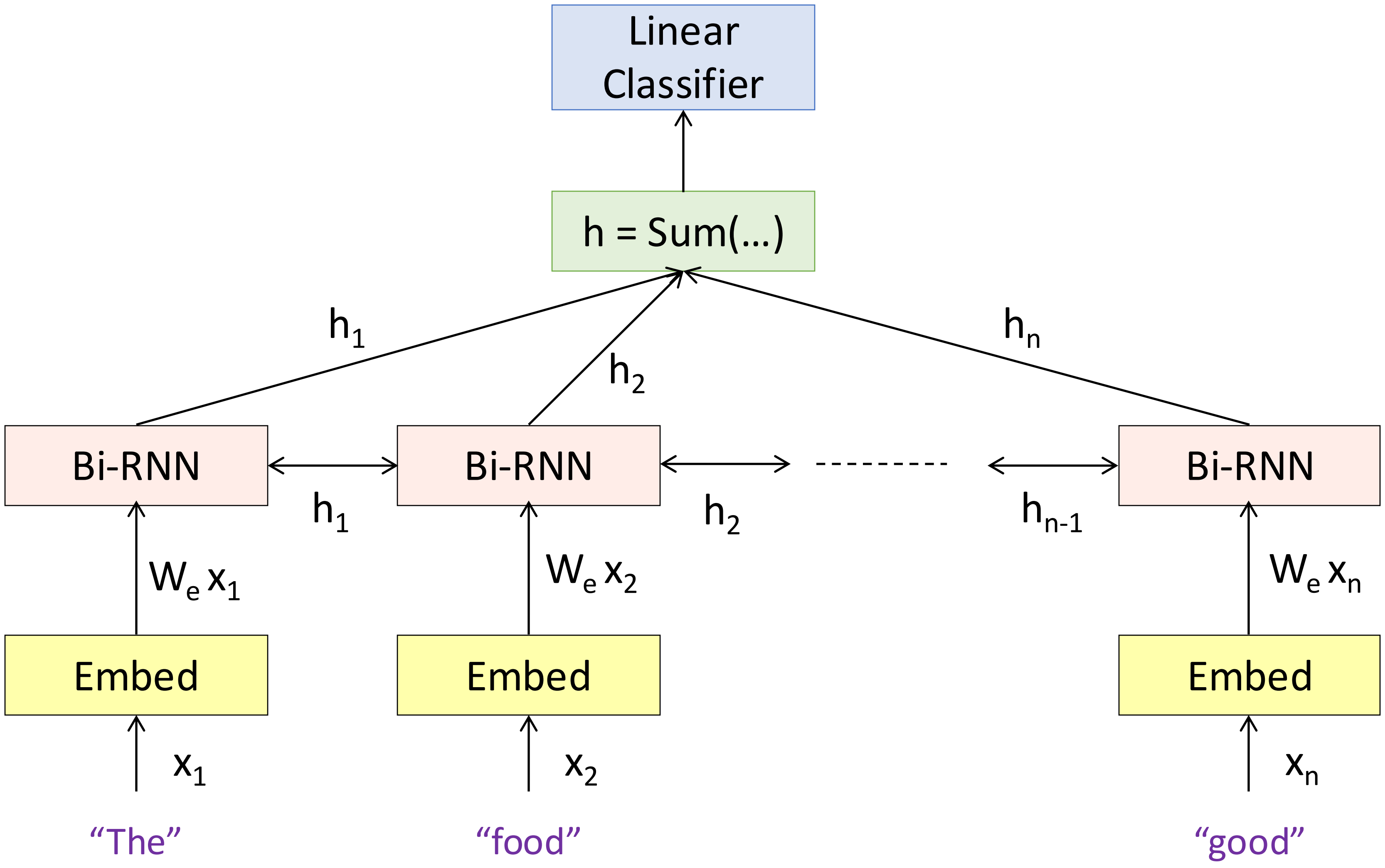
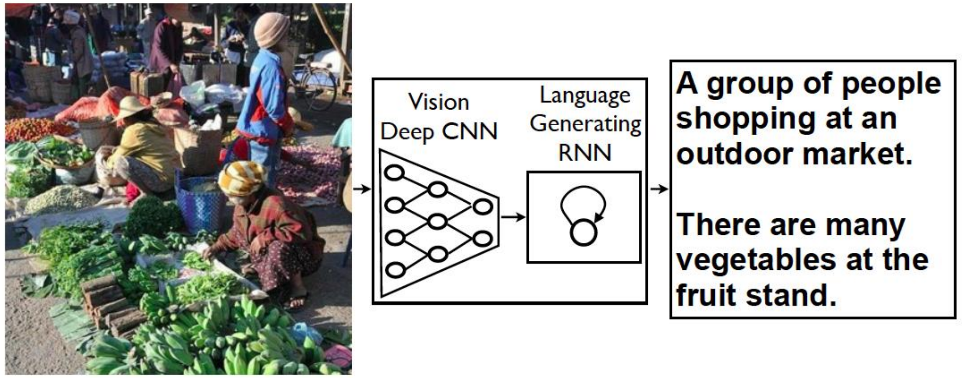
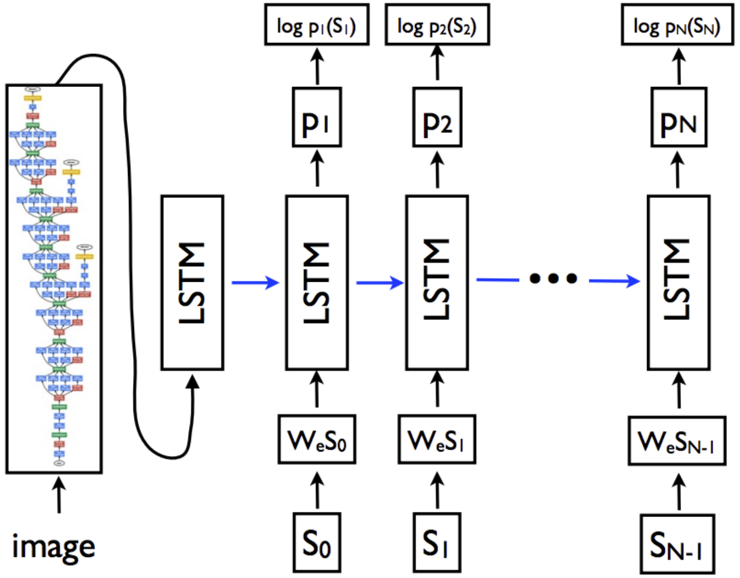


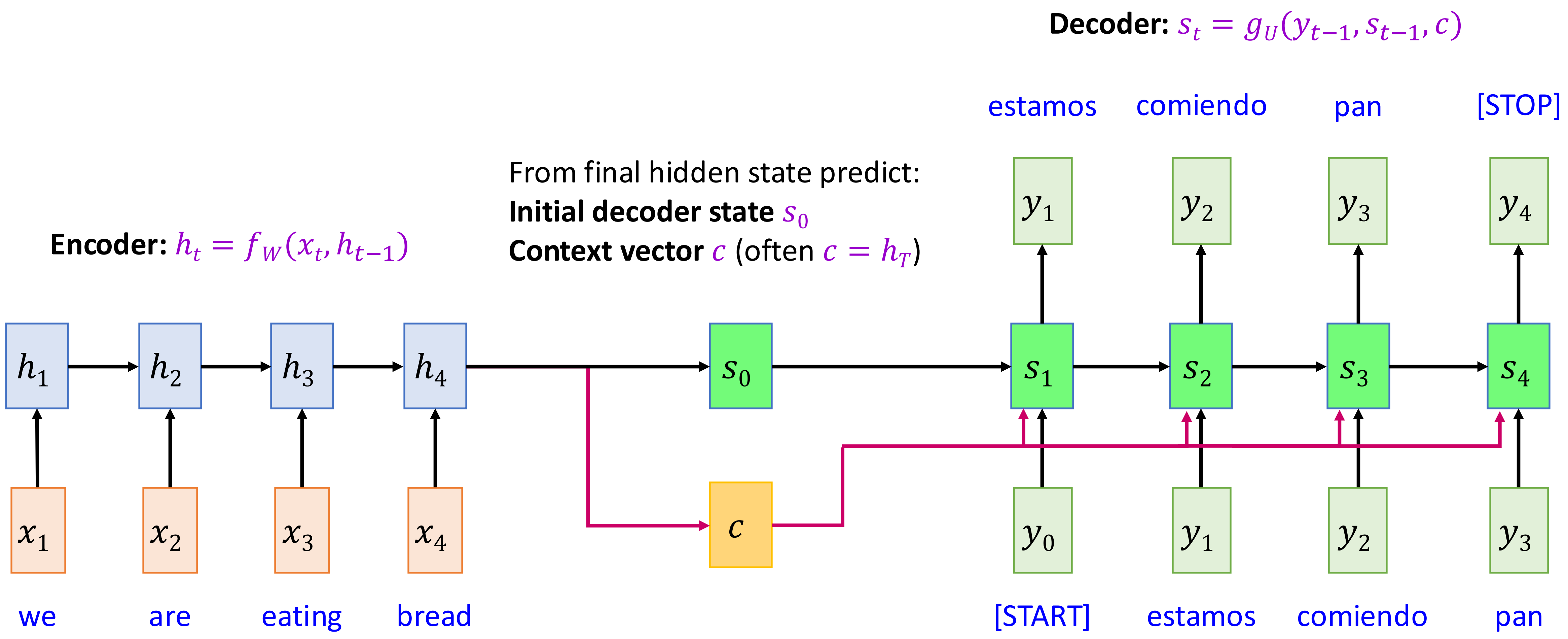
Image caption generation





O. Vinyals, A. Toshev, S. Bengio, D. Erhan, [Show and Tell: A Neural Image Caption Generator](), CVPR 2015

Sequence-to-sequence with RNNs



Next Class

- RNN with the attention mechanism