Recurrent neural networks

Ch. 10 Deep Learning book

by Goodfellow et al. (2016)

Dr. Odette Scharenborg

Associate professor & Delft Technology Fellow
Multimedia Computing Group

Slides partially based on slides Silvia-Laura Pintea (2018)

My background

- MA Language, Speech, and Computer Science @Radboud University (2000)
- PhD Human and Automatic Speech Recognition @Radboud University (2005)
- Associate professor Multimedia Computing Group @TU Delft (since 2018)

Research topic: Human speech processing inspired automatic speech recognition

Question

- Imagine a model for transcribing speech into text (= automatic speech recognition)
- What is the problem when using a convolutional network?
- Answer: Convolutions are well-suited for grids of values, X, whereas speech-to-text is a sequential task.

- Last week: Convolutional neural networks

→ Specialized for processing a grid of values, e.g., images

- Today: Recurrent neural networks (Rumelhart et al., 1986)

→ Specialized for processing sequential data:

$x^{(1)}, \ldots, x^{(t)}$

After this lecture

You will

- Know in what situations RNNs are used
- Know how it is defined
- Know how to train it
- Know the main challenge for RNNs
- Know about the most important RNN architectures
- Have seen an in-depth example of the use of an RNN

1. Definition of an RNN
2. Training an RNN
3. Other architectures
4. Challenges
5. Use of an RNN

Example

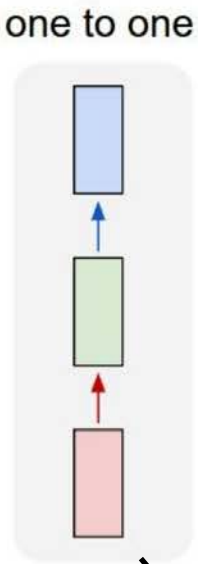
1. *I went to the US last week.*
2. *Last week, I went to the US.*

Question to ML model: Where did I go last week?

Task of the ML model: Recognize ‘the US’ as the relevant information irrespective of where in the sentence it is

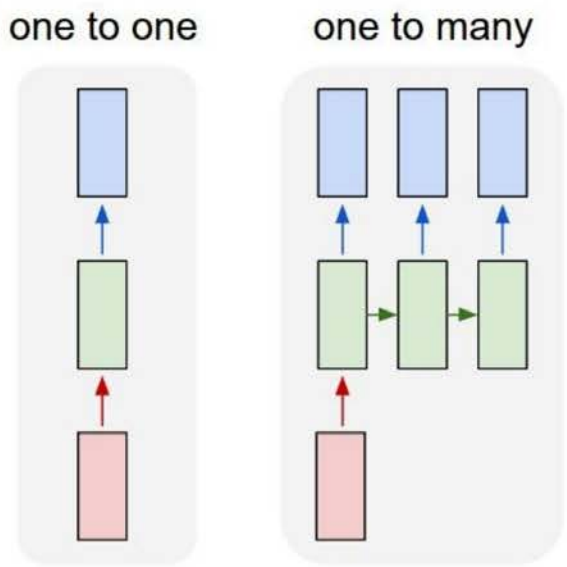
- Separate parameters: All rules of the language need to be learned separately
- RNNs: Same weights across several time steps → sharing of the same information on different time steps

Sequential data



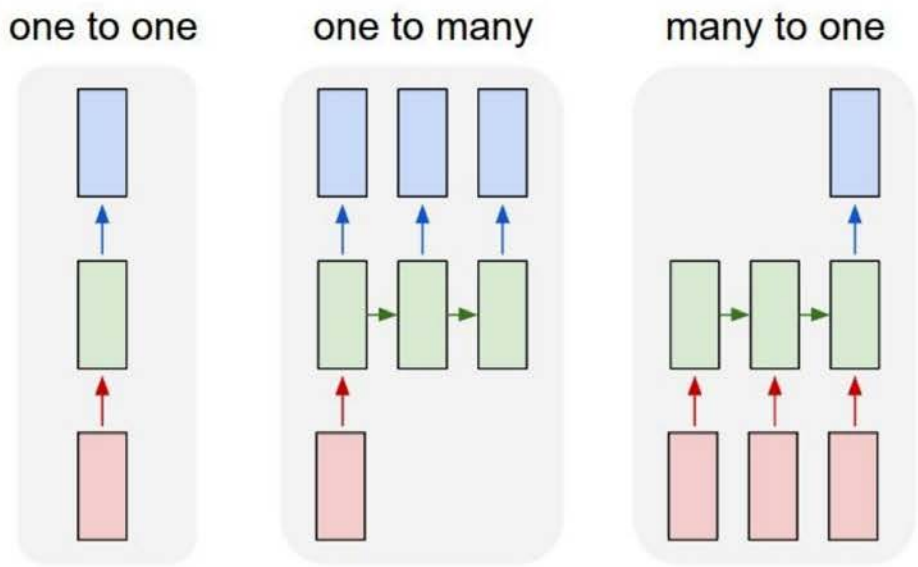
e.g., character recognition:
image to letter

Sequential data



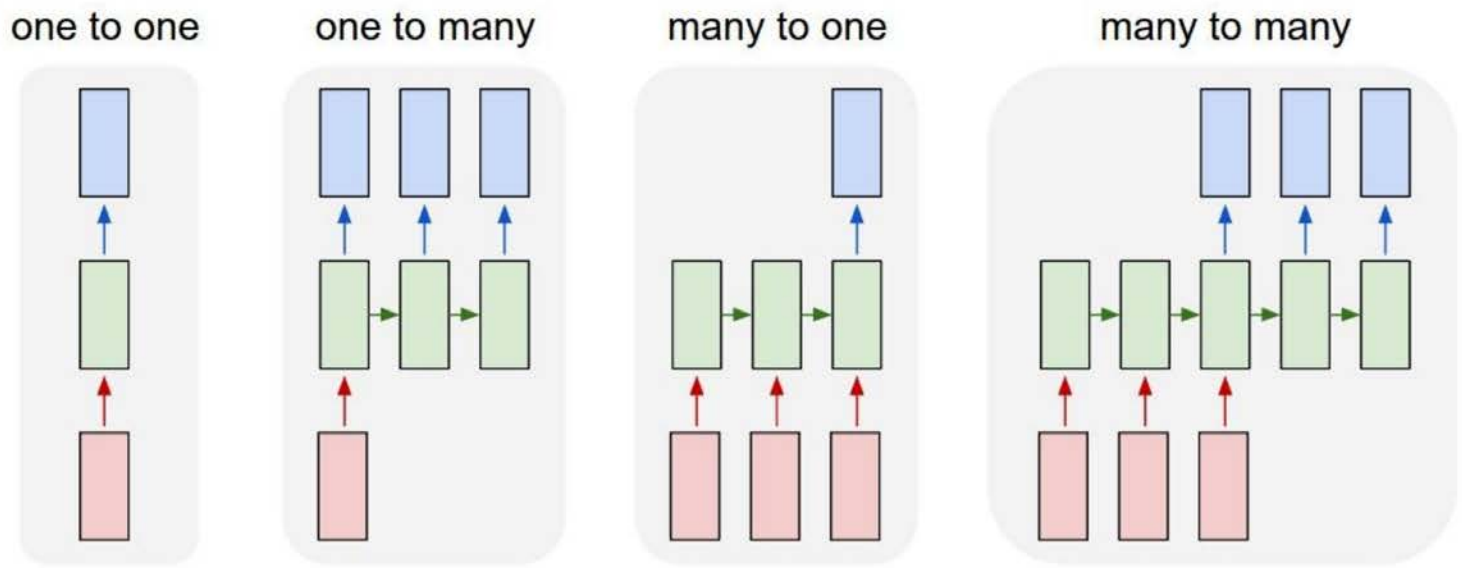
e.g., image captioning: image to sequence of words

Sequential data



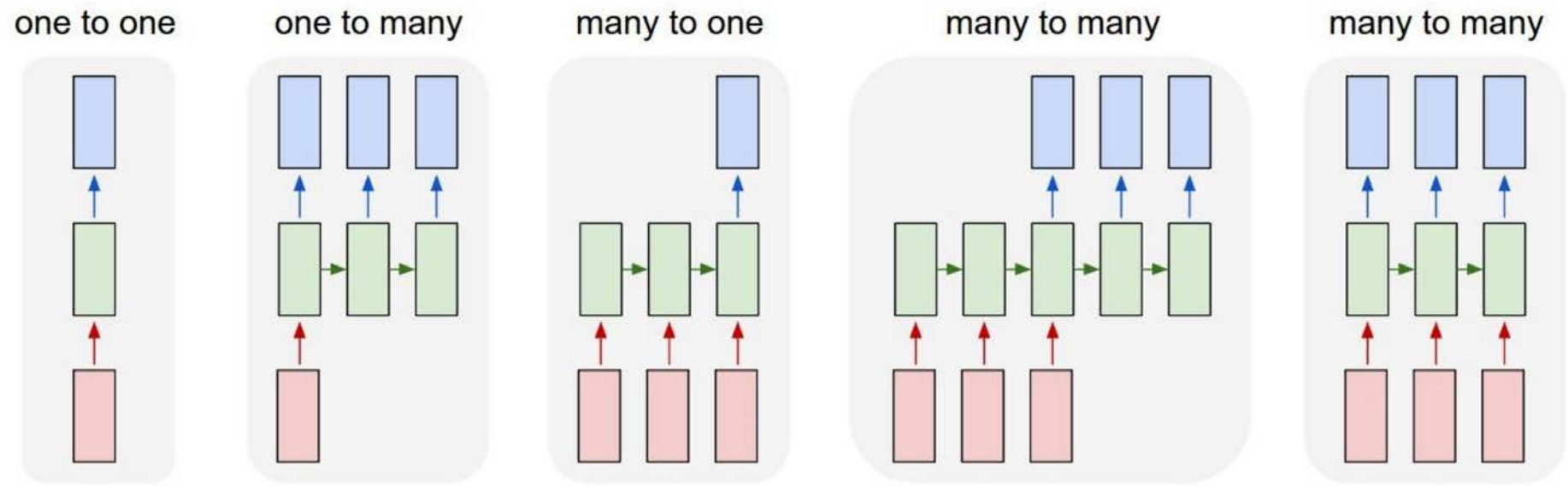
e.g., emotion classification:
sequence of words to emotion

Sequential data



e.g., machine translation: sequence of words to sequence of words

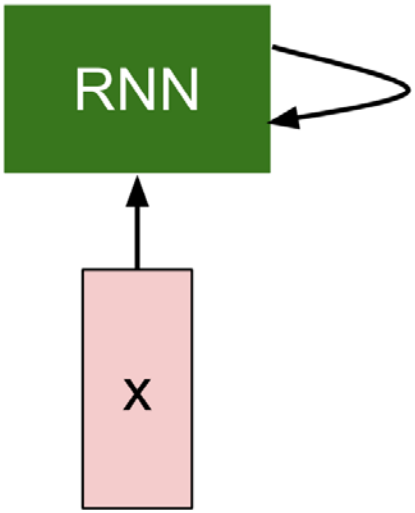
Sequential data



e.g., video classification on the frame level

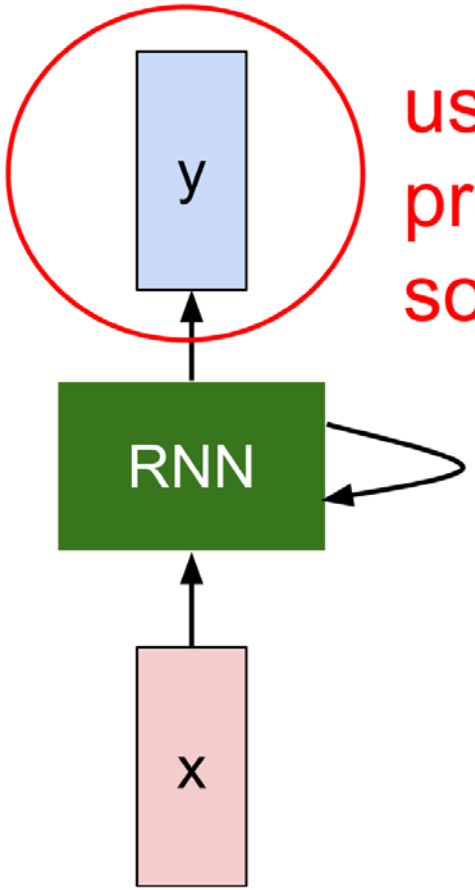
Slide adapted from Fei-Fei Li & Justin Johnson & Serena Yeung

Recurrent Neural Network



Slide adapted from Fei-Fei Li & Justin Johnson & Serena Yeung

Recurrent Neural Network



usually want to predict a vector at some time steps

Recurrent Neural Network

An RNN processes a sequence of vectors $\boldsymbol{x}$ by applying a **recurrence formula** at every time step:

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

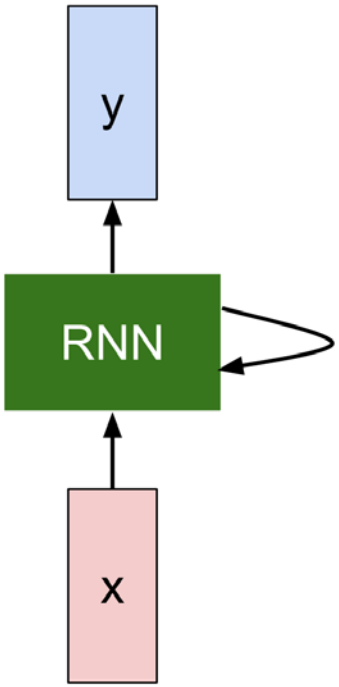
new state

some function with parameter W

old state

input vector at some time step

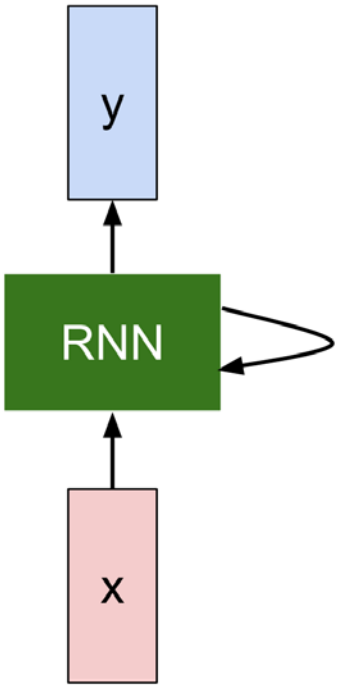
recurrence



Recurrent Neural Network

An RNN processes a sequence of vectors $\boldsymbol{x}$ by applying a **recurrence formula** at every time step:

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



Notice: the same function and the same set of parameters are used at every time step.

RNN as a computational graph

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}; \boldsymbol{\theta})$$

recurrence

$t = 3$

$$\boldsymbol{s}^{(3)} = f(\boldsymbol{s}^{(2)}; \boldsymbol{\theta})$$

$$= f(f(\boldsymbol{s}^{(1)}; \boldsymbol{\theta}); \boldsymbol{\theta})$$

no recurrence

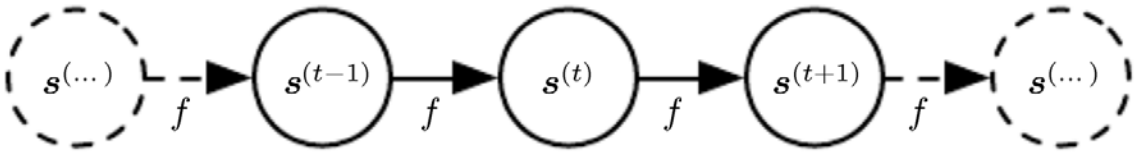


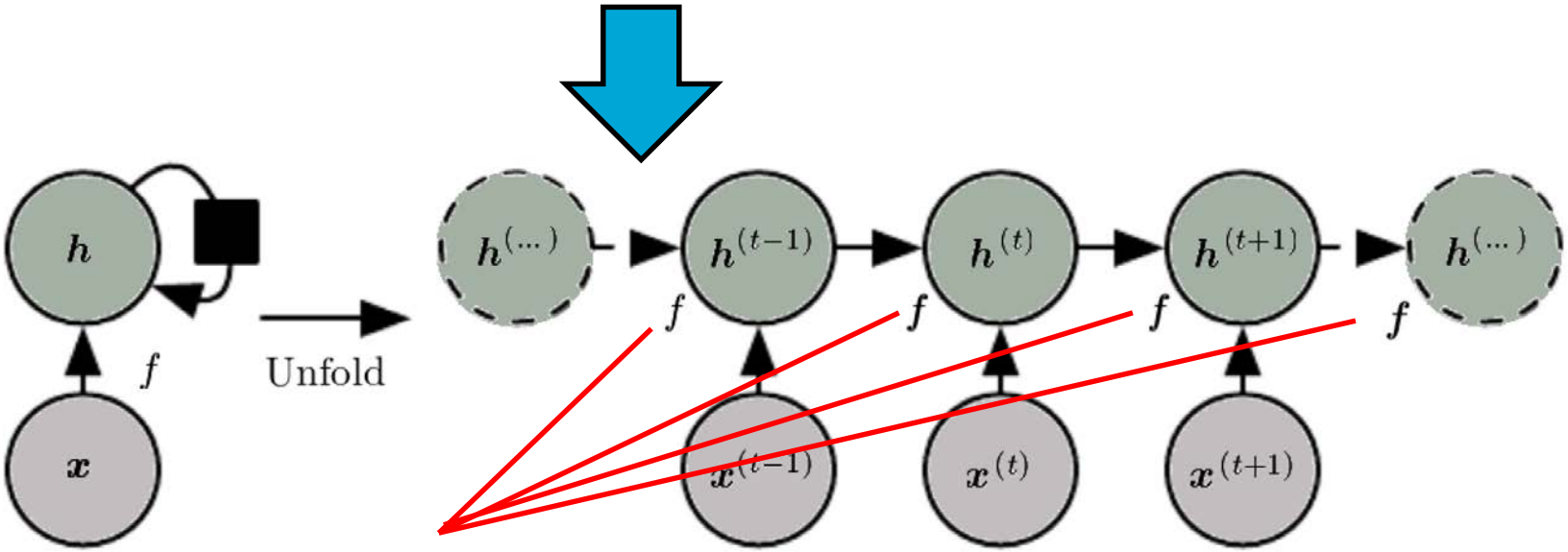
Figure 10.1: The classical dynamical system described by equation 10.1, illustrated as an unfolded computational graph. Each node represents the state at some time t, and the function f maps the state at t to the state at $t+1$. The same parameters (the same value of $\boldsymbol{\theta}$ used to parametrize f) are used for all time steps.

An example with input x

$$\boldsymbol{s}^{(t)} = f(\boldsymbol{s}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

- Computational graph → RNN: state s is hidden unit h

$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$



Possible to use the same f with the same parameters at every time step

Unfolding the recurrence after t steps with function $g^{(t)}$

$$\begin{aligned}\boldsymbol{h}^{(t)} &= g^{(t)}\left(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right) \\ &= f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}).\end{aligned}$$

- $g^{(t)}$ takes the whole past sequence

$$\left(\boldsymbol{x}^{(t)}, \boldsymbol{x}^{(t-1)}, \boldsymbol{x}^{(t-2)}, \ldots, \boldsymbol{x}^{(2)}, \boldsymbol{x}^{(1)}\right)$$

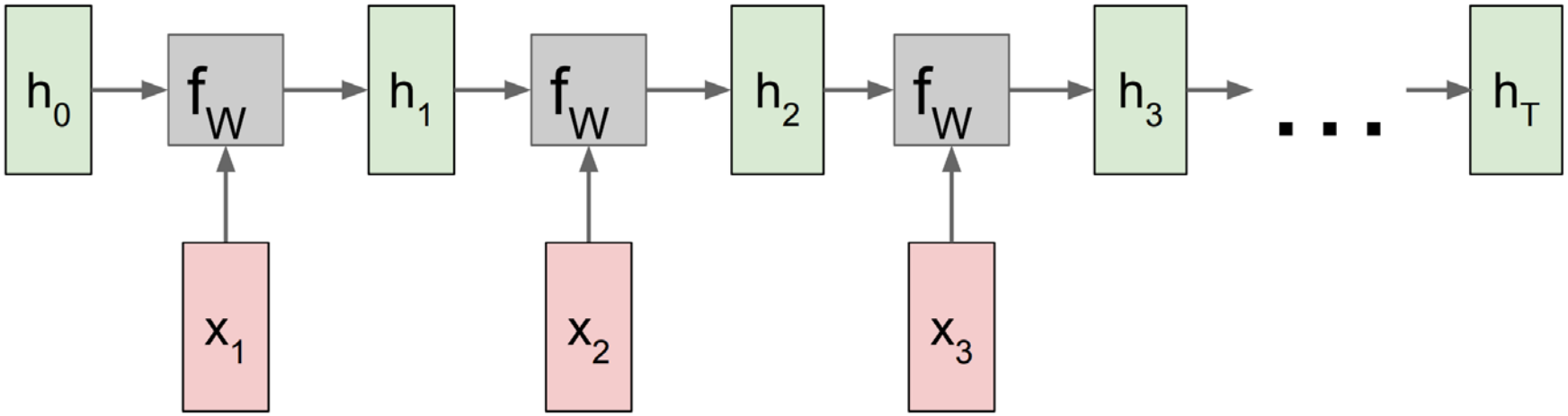
as its input and produces the current state

Two major advantages of recurrence

- Regardless of sequence length, the learned model always has the same input size, because sequence length is specified in terms of state transitions rather than a variable-length history of states

- The *same* transition function f with the same parameters can be used at every time step

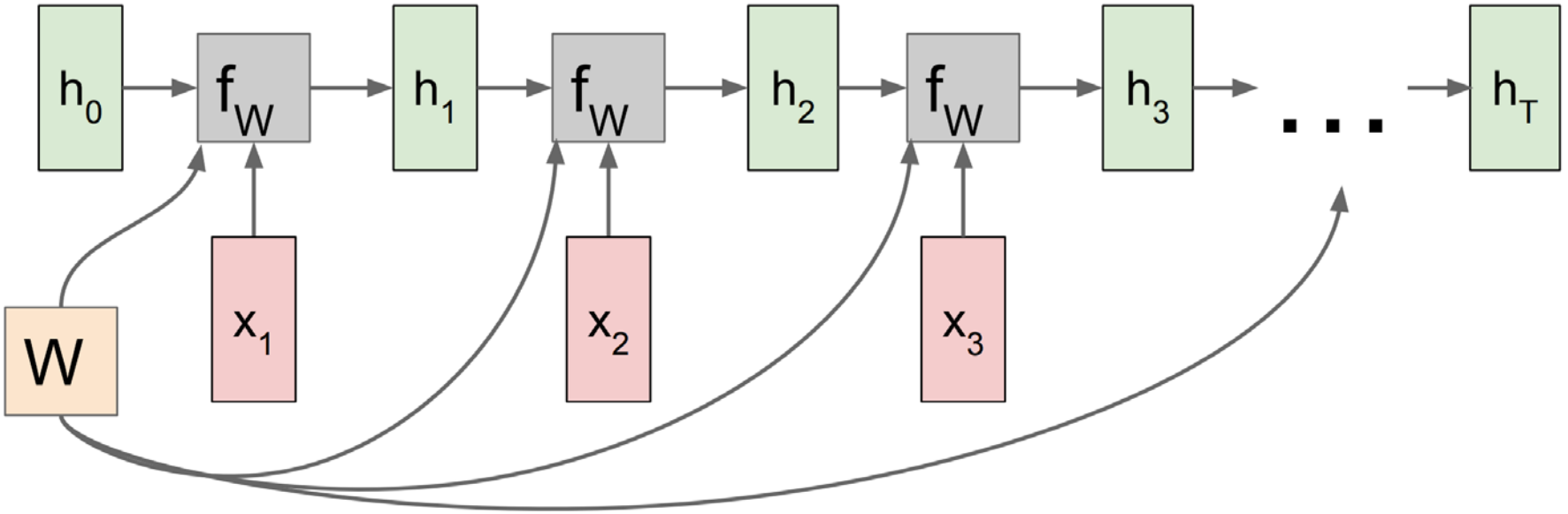
Computational graph: Parameter sharing



Slide adapted from Fei-Fei Li & Justin Johnson & Serena Yeung

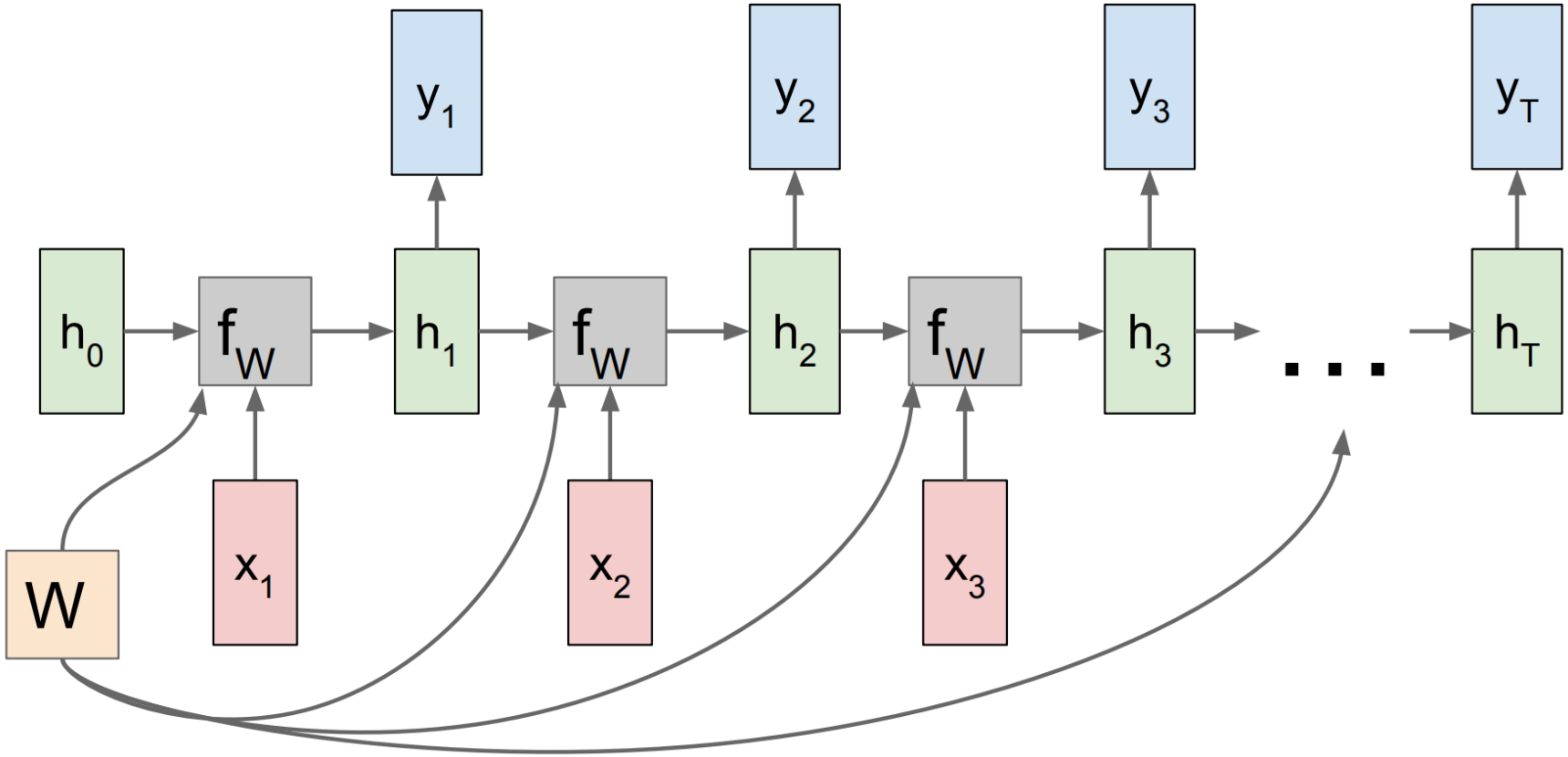
Computational graph: Parameter sharing

- Reuse the same weight matrix at every time step



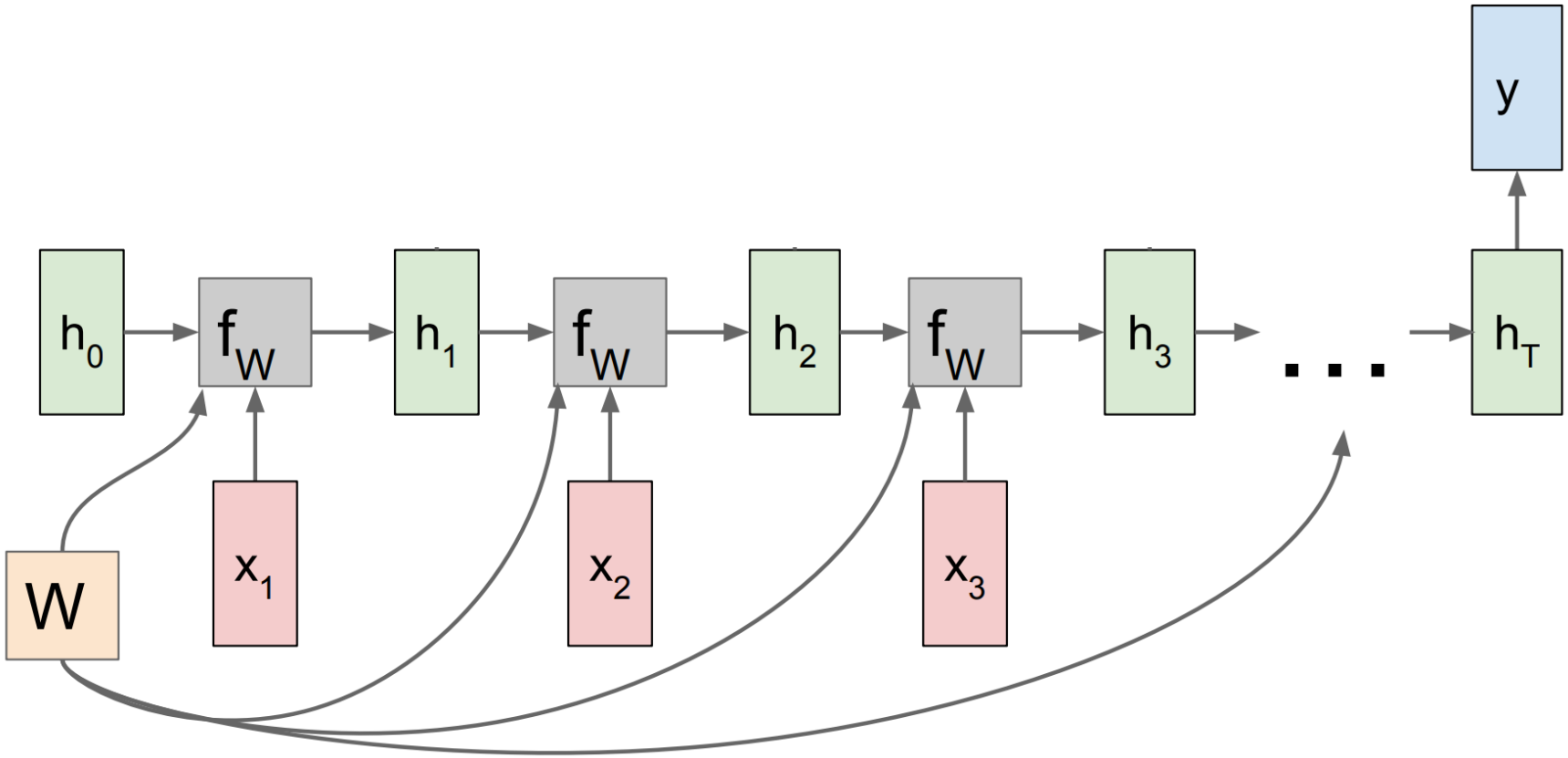
Slide adapted from Fei-Fei Li & Justin Johnson & Serena Yeung

Many-to-many



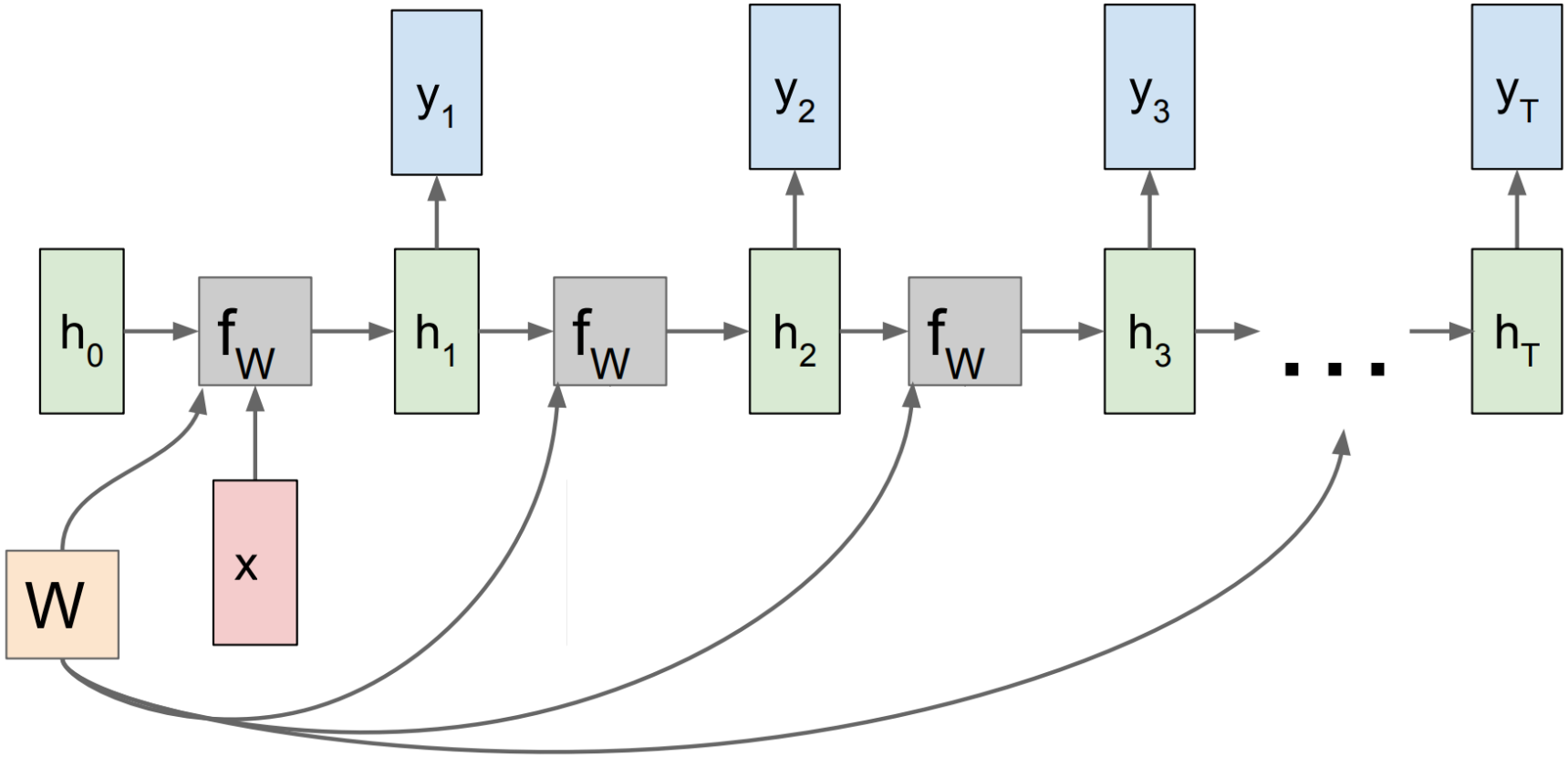
Slide adapted from Fei-Fei Li & Justin Johnson & Serena Yeung

Many-to-one



Slide adapted from Fei-Fei Li & Justin Johnson & Serena Yeung

One-to-many



Slide adapted from Fei-Fei Li & Justin Johnson & Serena Yeung

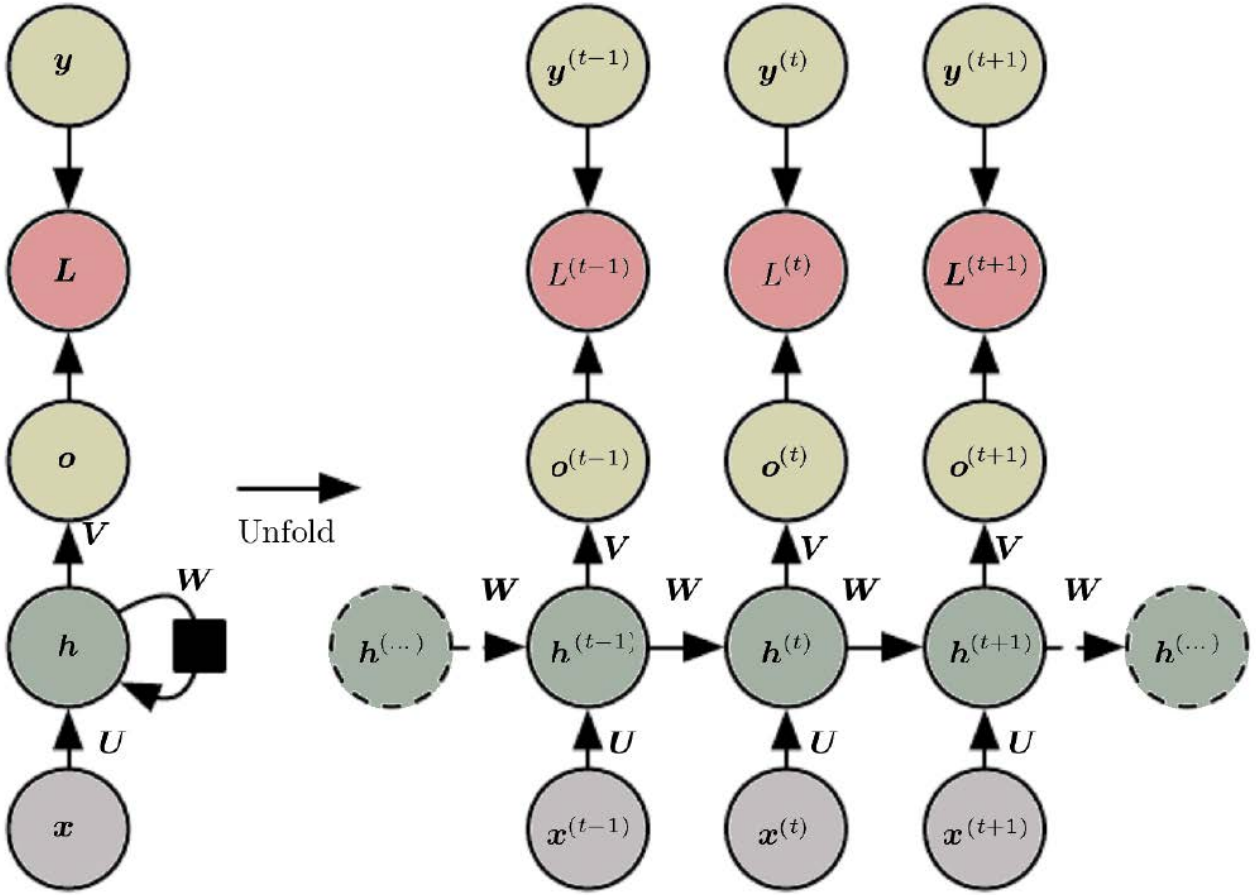
Strengths of an RNN

- Learns a single model f that operates on all time steps and all sequence lengths rather than a separate $g^{(t)}$ for all possible time steps
- Generalization to sequence lengths unseen during training
- Sharing statistical strength across different sequence lengths and across different positions in time

1. Definition of an RNN
2. Training an RNN
3. Other architectures
4. Challenges
5. Use of an RNN

Where to add the recurrent connections?

Representative RNN



Training an RNN

- Forward propagation: Initial state $h^{(0)}$

Then for t=1 to t=tau

$$
\begin{aligned}
\boldsymbol{a}^{(t)} &= \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)}, \\
\boldsymbol{h}^{(t)} &= \tanh(\boldsymbol{a}^{(t)}), \\
\boldsymbol{o}^{(t)} &= \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)}, \\
\hat{\boldsymbol{y}}^{(t)} &= \mathrm{softmax}(\boldsymbol{o}^{(t)}),
\end{aligned}
$$

- b and c are bias vectors
- U (input-to-hidden), V (hidden-to-output), and W (hidden-to-hidden) are weight matrices

Training an RNN

- The total loss for a given sequence of x values paired with a sequence of y training values would then be the sum of the losses over all the time steps:

$$L\left(\{\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(\tau)}\},\{\boldsymbol{y}^{(1)},\ldots,\boldsymbol{y}^{(\tau)}\}\right)$$
$$=\sum_{t} L^{(t)}$$

Training an RNN

- The gradient of the loss is calculated using a forward propagation pass followed by a backward propagation pass and used for retraining the RNN (using standard algorithms)

Expensive:

- In time: forward pass is necessarily sequential
- In memory: states computed in the forward pass must be stored until they are reused during the backward pass

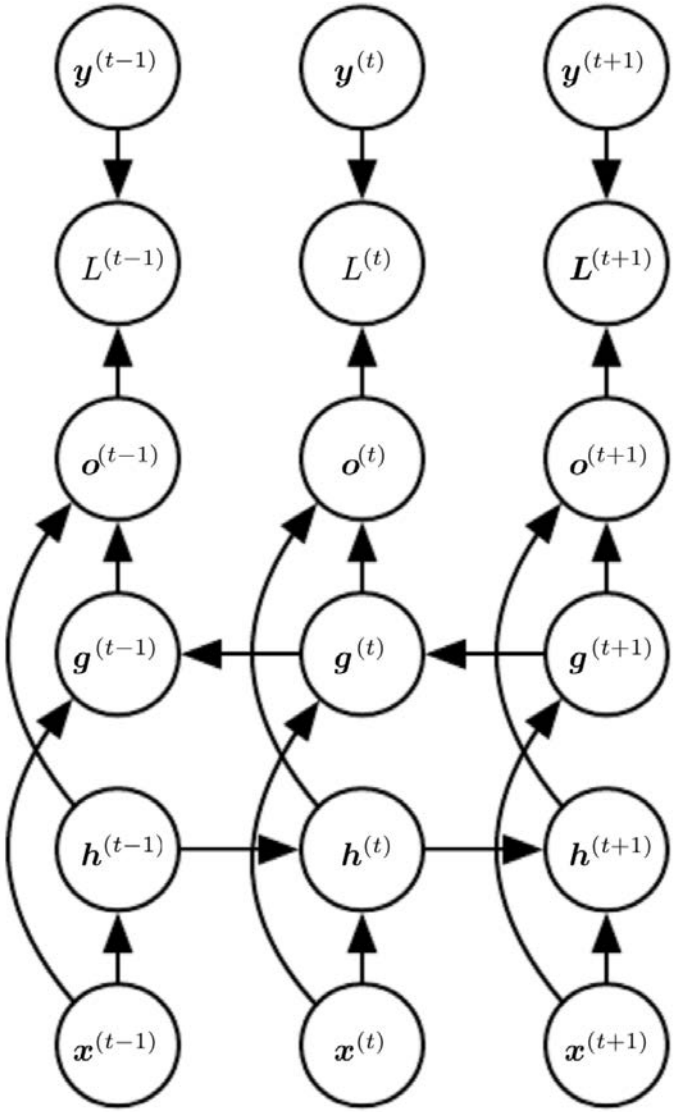
1. Definition of an RNN
2. Training an RNN
3. Other architectures
4. Challenges
5. Use of an RNN

Special RNN architectures

Bidirectional RNN

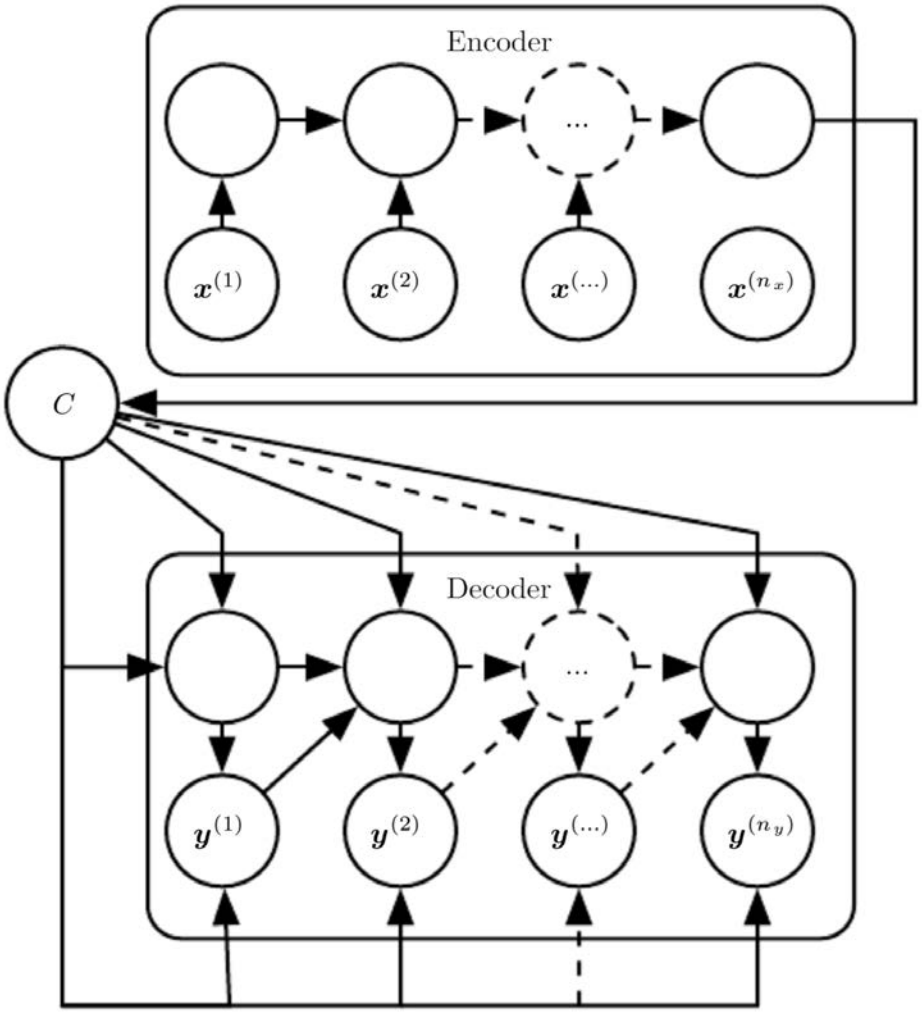
- $h(t)$ = forward pass
- $g(t)$ = backward pass

→Useful for e.g., automatic speech recognition where the pronunciation of a sound can be dependent on future sounds due to coarticulation



Encoder-decoder *or* sequence-to-sequence

→Useful for differences in length between input and output, e.g., as in machine translation



1. Definition of an RNN
2. Training an RNN
3. Other architectures
4. Challenges
5. Use of an RNN

Long-term dependencies

- RNNs compute $\mathbf{h}^{(t)} = \mathbf{W}^T \mathbf{h}^{(t-1)}$ many times

- Question: What is the problem with these long-term dependencies?

- Answer: Vanishing or exploding gradients

→ It might take a very long time to learn long-term dependencies

Various solutions

→ Design a model that operates at multiple time scales

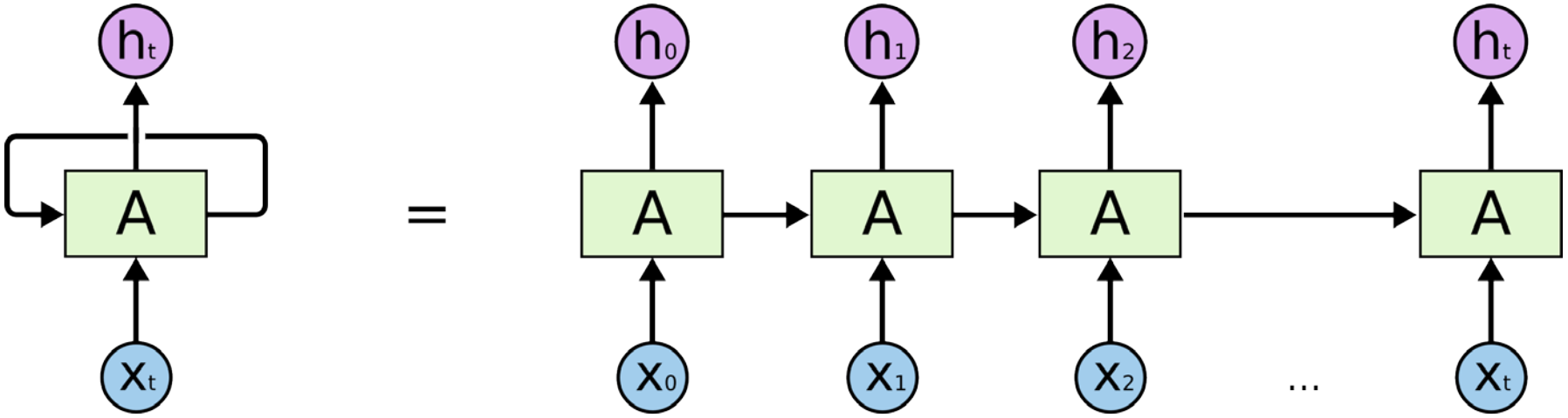
- The most effective model: gated RNNs

→Designed to remember information for long periods of time

→Scales and transfers information from the distant past to the present more efficiently

Long Short Term Memory (LSTMs)

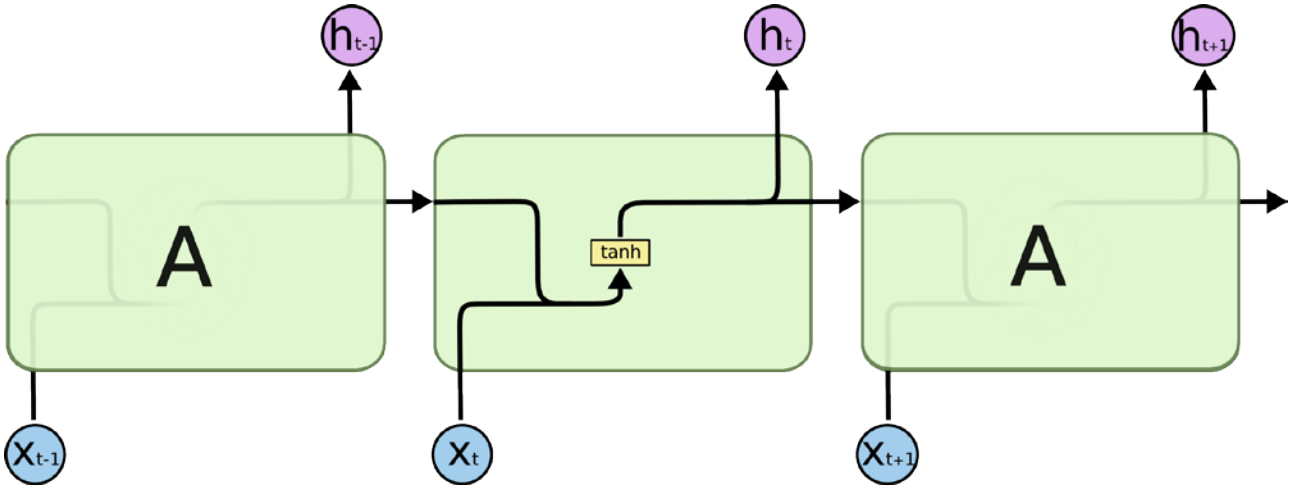
- An unrolled recurrent neural network



Let's open it up

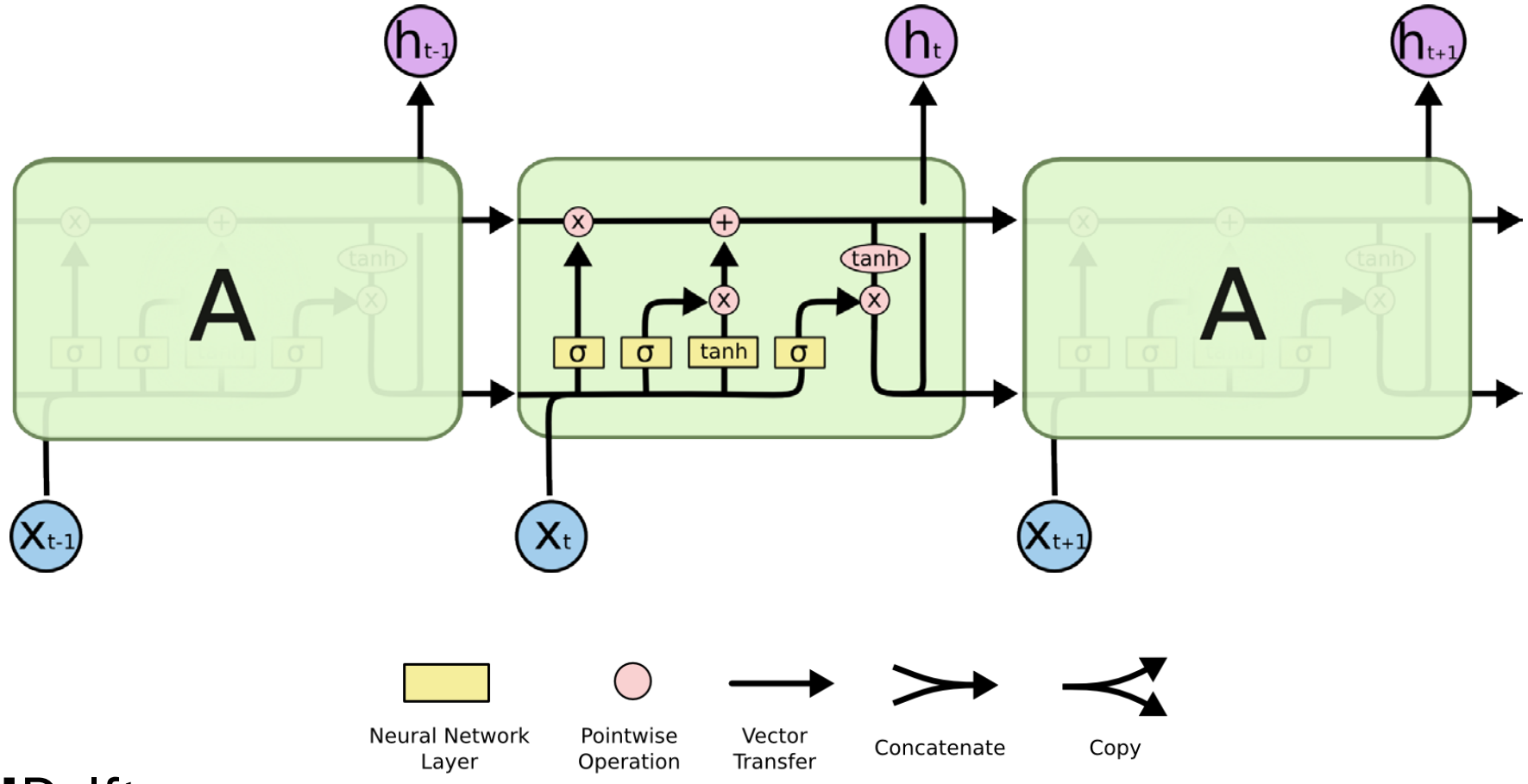
The next LSTM slides follow the tutorial of Christopher Olah (Google Brain): http://colah.github.io/posts/2015-08-Understanding-LSTMs/

- The repeating module in a standard RNN contains a single (often tanh) layer



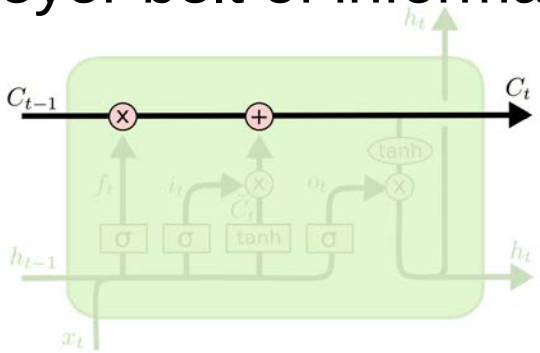
We had $h^{(t)} = \tanh^{(a(t))} = (Wh^{(t-1)} + b) + Ux^{(t)}$

LSTMs: 4 interacting layers

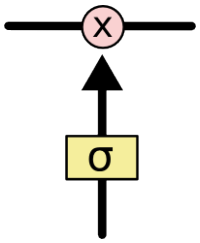


The key to LSTMs is the cell state C_t

- Kind of like a conveyor belt of information



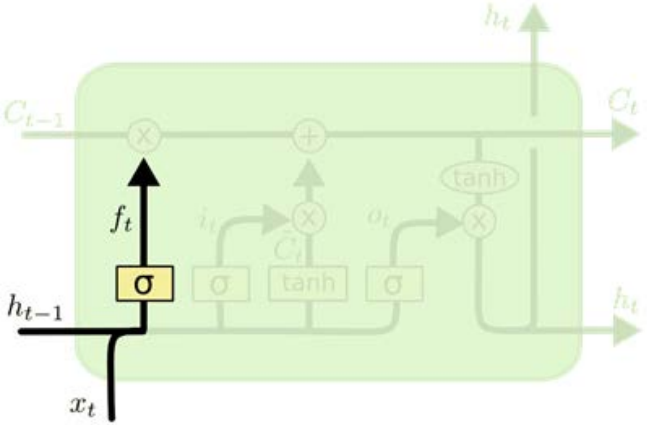
- Adding and removing information to C_t via gates: sigmoid + pointwise multiplication



- Question: Why sigmoids?
- Answer: Sigmoids return values between [0,1]
  0 = forget this, 1 = remember completely

Forget gate: Removing information C_t

- Removing information from the input or the history



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

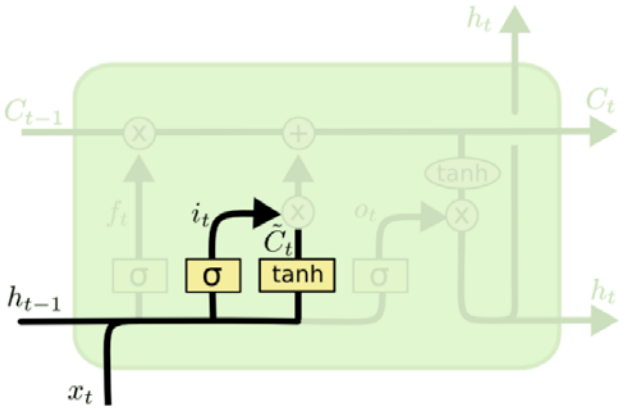
e.g. removing gender information (F) when a new subject is encountered:

"She spoke perfect French. The man, though, was born in Germany and he… "

Input gate: Adding information

- Storing information in the cell state

1. What values to update?
2. Create a new vector



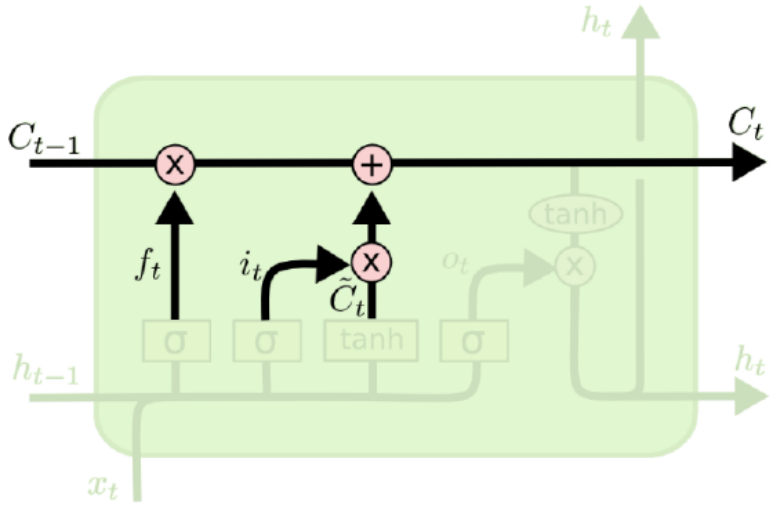
$$\mathbf{i}_t = \sigma(\mathbf{W}_i[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_i)$$

$$\tilde{\mathbf{C}}_t = \tanh(\mathbf{W}_C[\mathbf{h}_{t-1}, \mathbf{x}_t] + \mathbf{b}_C)$$

e.g. adding gender information (M) when a new subject is encountered:

“She spoke perfect French. The man, though, was born in Germany and he…”

Updating the old cell state C_{t-1}



C_t as input to the 3 gates of C_{t+1}

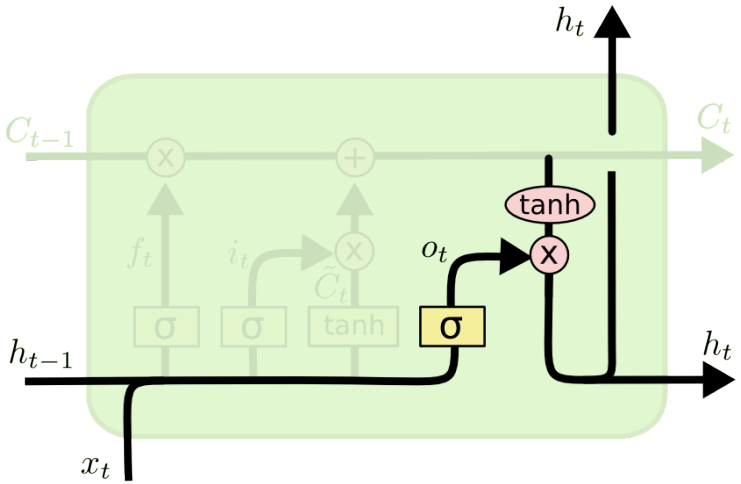
$$\mathbf{C}_t = \mathbf{f}_t \otimes \mathbf{C}_{t-1} + \mathbf{i}_t \otimes \tilde{\mathbf{C}}_t.$$

forgetting what we decided to forget

adding what we decided to add

Output gate

1. Sigmoid layer to decide which parts of the cell state are going to be output to the next layer
2. Tanh → values between -1 and 1
3. Multiply by the output of the sigmoid gate so we only output the parts we decided to



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

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

Other gated RNNs

- Units are known as gated recurrent units (GRUs)
- Single gating unit simultaneously controls forgetting and decision to update

- Other variations of LSTMs:

Peephole connections: F.A. Gers and J. Schmidhuber (2000). Recurrent nets that time and count, IJCNN.

- Other gated RNN models:

J. Chung et al. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling, NIPS.

Models with explicit memory

Deep nets are good at processing large amounts of info to predict implicit information: e.g., the difference between cats and dogs but

- Struggle to memorize facts: e.g., a cat is a kind of animal
- Cannot remember info for too long, nor completely.

→ Explicit memory:

- The network has a set of memory cells that can be addressed.

  [S. Sukhbaatar et al. (2015). End-to-end memory networks, NIPS]
- Reading and writing to these memory cells without explicit supervision, through an attention mechanism.

  [J. K. Chorowski et al. (2015). Attention-based models for speech recognition", NIPS]

1. Definition of an RNN
2. Training an RNN
3. Other architectures
4. Challenges
5. Use of an RNN

Example of an RNN-based system

- **Image2speech**

[Hasegawa-Johnson, Black, Ondel, Scharenborg, Ciannella (2018). Image2speech: Automatically generating audio descriptions of images. Journal of International Science and General Applications, 1, 19-27]

The following slides are adapted from Hasegawa-Johnson (2018)

Background

- If there is no writing system, e.g., as in many Arabic dialects, then speech is the only communication tool, but:

Must one, e.g., speak in Modern Standard Arabic to be understood by one's cell phone?

- Task Definition: Can we develop speech technology in a dialect that is almost never written down in any standardized text format?

- Image2speech: An algorithm that observes an image, and generates a spoken description of the image, without requiring that the language of the description has any standardized text format

Datasets

- AMT recordings obtained from **Flickr8k** - 40k spoken captions available online
  - https://groups.csail.mit.edu/sls/downloads/
  - D. Harwath and J. Glass, *"Deep multimodal semantic embeddings for speech and images"* in IEEE ASRU, Scottsdale, Arizona, USA, December

-A brown and white dog is running through the snow
-A dog is running in the snow
-A dog running through snow
-A white and brown dog is running through a snow covered field
-The white and brown dog is running over the surface of the snow

Image2speech: system overview

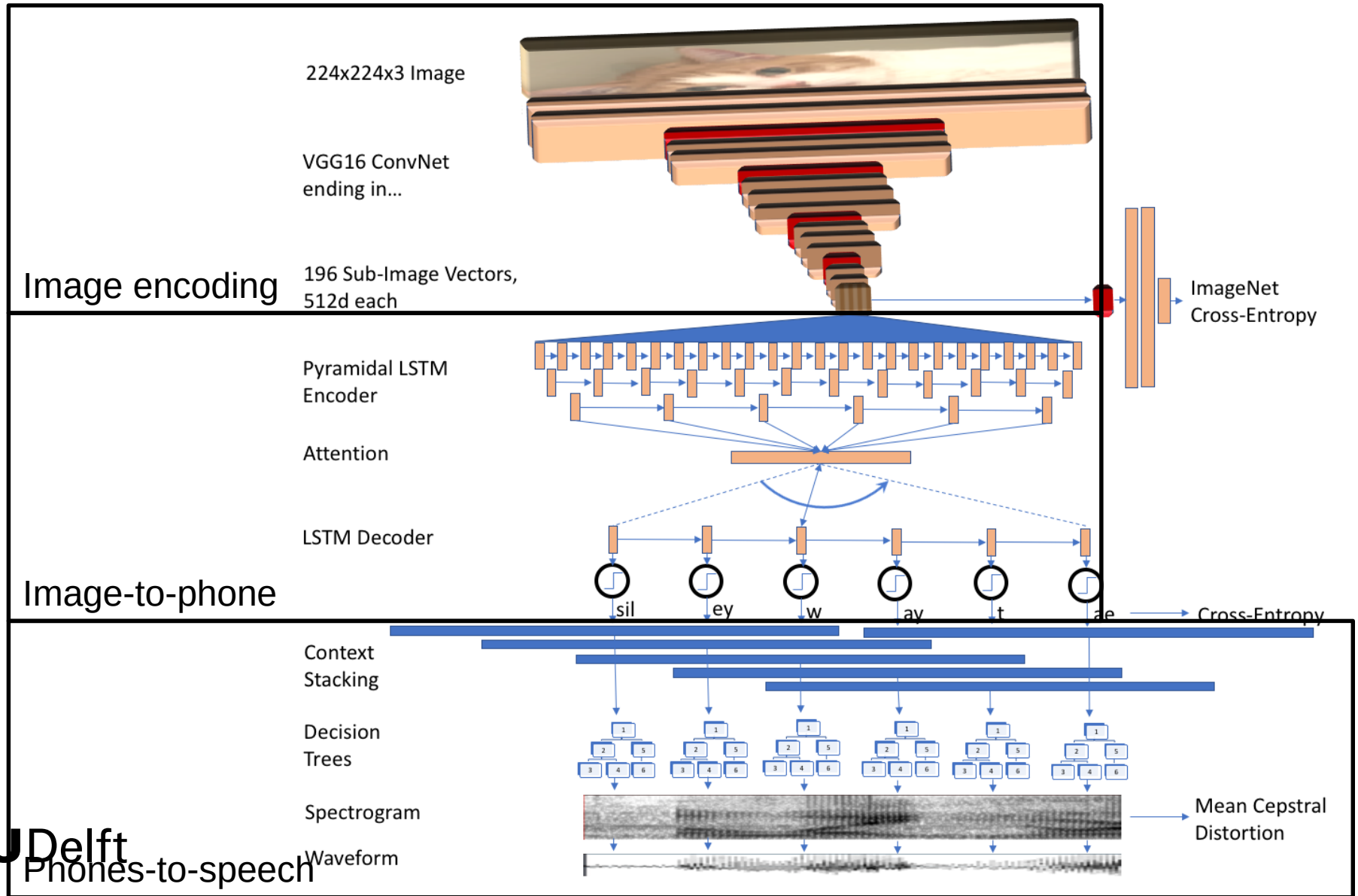


Image2speech: system overview

- Image representation: very large scale convolutional neural net, trained on imagenet classification
- Image-to-phone: neural machine translation! Sequence-to-sequence with attention
- Phones-to-speech: speech synthesis, audio frames selected using decision trees

image2speech: system overviews

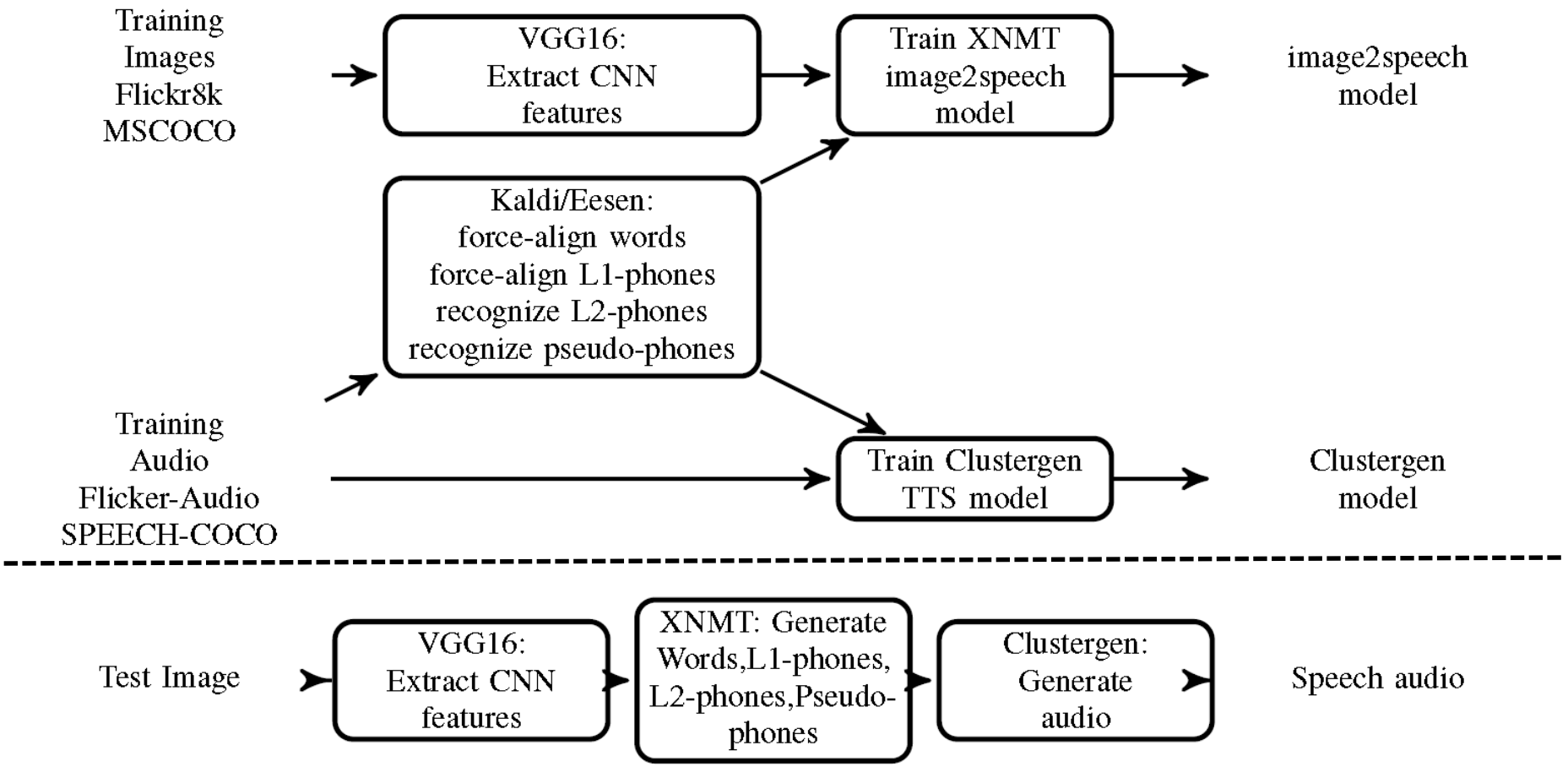


Image representation: CNNFEAT $\vec{s}_{mn}$

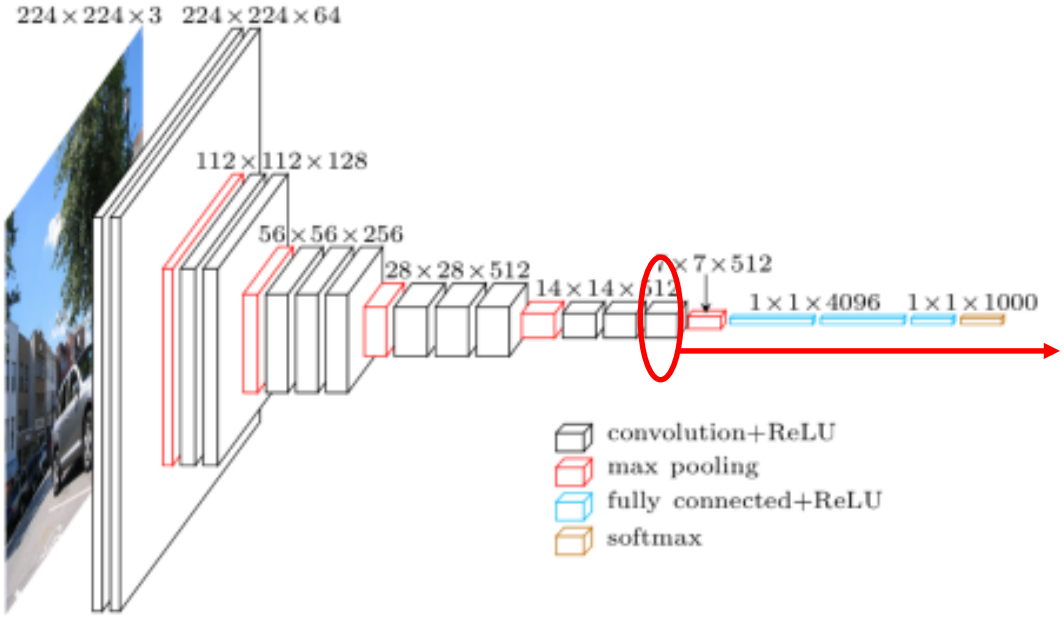


Figure copied from Simonyan & Zisserman, 2014.

- ImageNet = >500 images/noun of each of the nouns in WordNet.
- VGG = 13-layer CNN + 2-layer FCN, trained on 14m images, covering the 1000 most numerous nouns, 92.7% top-5 test accuracy.
- CNNFEAT: 196 feature vectors/image, 512d/vector, from the last CNN layer. Each receptive field covers about 40x40 pixels in the original 224x224 image.

image2phones: phones from images

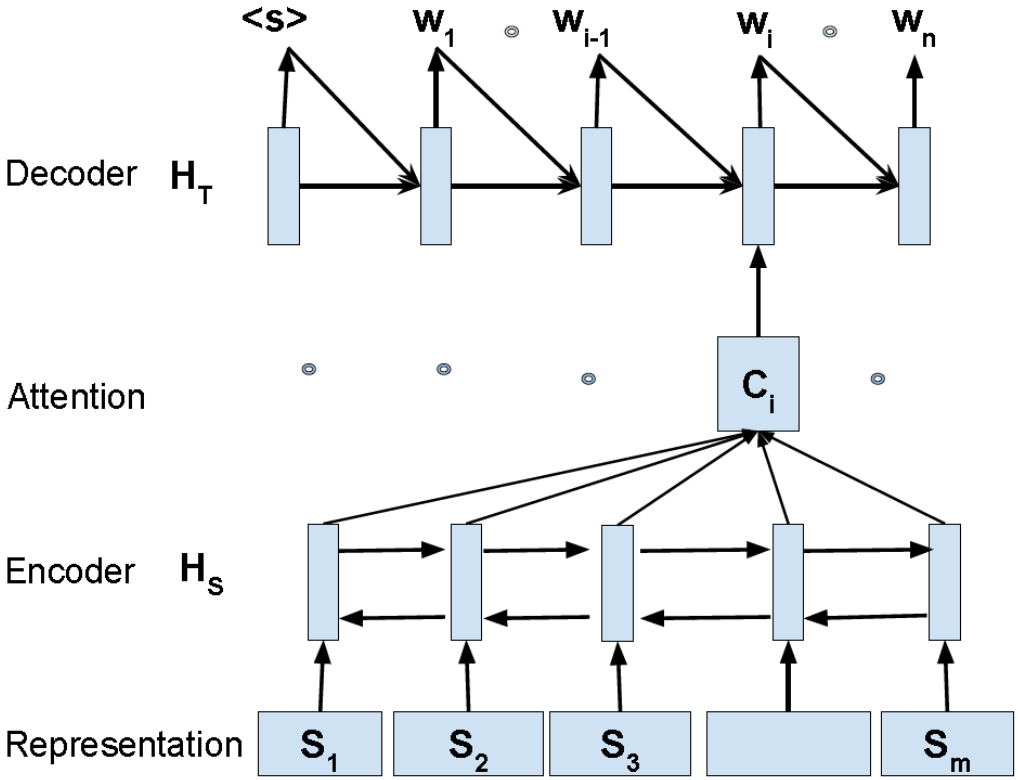


Figure copied without permission from Duong, Anastasopoulos, Chiang, Bird & Cohn, NAACL-HLT 2016.

- "Representation:" 196 vectors/image
- "Encoder:" PyramidalLSTM with one 128d state vector. Sequence is row-wise raster scan of the image.
- "Attention:" StandardAttender, 128d input, 128d state vector, N hidden nodes
- "Decoder:" MlpSoftmaxDecoder, 3 layers, 1024d hidden vectors
- Output vocabulary: force-aligned phones (flickr8k)

image2phones: phones from images

- Output: $y_t(k) = P(w_k|\vec{h}_t)$, where w_k is an acoustic unit (pseudo-phone), and $\vec{h}_t$ is the decoder state at time t
- Decoder state: $\vec{h}_t = f(\sum_{m,n} a_{mn}(t)\vec{g}_{mn}, \vec{h}_{t-1})$ is an output LSTM state vector, computed from its own history ($\vec{h}_{t-1}$), and from a weighted sum ($a_{mn}(t)$) of the image-encoder inputs ($\vec{g}_{mn}$)
- Attention: $a_{mn}(t) = f(\vec{g}_{mn}, \vec{h}_{t-1})$ is a scalar MLP, specifying how much attention the t'th output should pay to the (m,n)th input
- Encoder state: $\vec{g}_{mn} = f(\vec{s}_{mn}, \vec{g}_{m,n-1})$ is an input LSTM state vector
- Input: sub-image features $\vec{s}_{mn} = CNN(subimage(m,n))$, where $m \in \{1, \ldots, 14\}$ and $n \in \{1, \ldots, 14\}$ are sub-image row and column, respectively

Results on Flicker8K

Images and Reference Texts: Hodosh, Young & Hockenmaier, 2013. Waveforms: Harwath and Glass, 2015

- Ref1: “The boy laying face down on a skateboard is being pushed along the ground by another boy.”
- Ref2: “Two girls play on a skateboard in a court yard.”
- Hyp (128d attender): SIL SIL T UW M EH N AA R R AY D IX NG AX R EH D AE N W AY T SIL R EY S SIL *(Two men are riding a red and white race)*
- Hyp (64d attender): SIL SIL T UW W IH M AX N W AO K IX NG AA N AX S T R IY T SIL *(Two women walking on a street)*

- Ref1: “A boy in a blue top is jumping off some rocks in the woods.”
- Ref2: “A boy jumps off a tan rock.”
- Hyp (128d attender): SIL SIL EY M AE N IH Z JH AH M P IX NG IH N DH AX F AO R EH S T SIL *(A man is jumping in the forest)*
- Hyp (64d attender): SIL SIL EY Y AH NG B OY W EY R IX NG AX B L UW SH ER T SIL IH Z R AY D IX NG AX HH IH L SIL *(A young boy wearing a blue shirt is riding a hill)*

TU Delft

Concluding remarks Image2speech

- New AI task
- Going from image to spoken captions is possible when the phone set is known
- The system seems to learn whole word (sequence)s

→ Lots of research needed to build a system that works for a truly unwritten language

Concluding remarks

- RNNs are the best models for sequential data
- Uses parameter sharing
- Allows for the prediction of
  - *Whole input sequences* (bidirectional RNN)
  - *Variable length sequences* (encoder-decoder)
- The problem of vanishing or exploding gradients can be dealt with using gated RNNs
- LSTMs are the most powerful RNNs