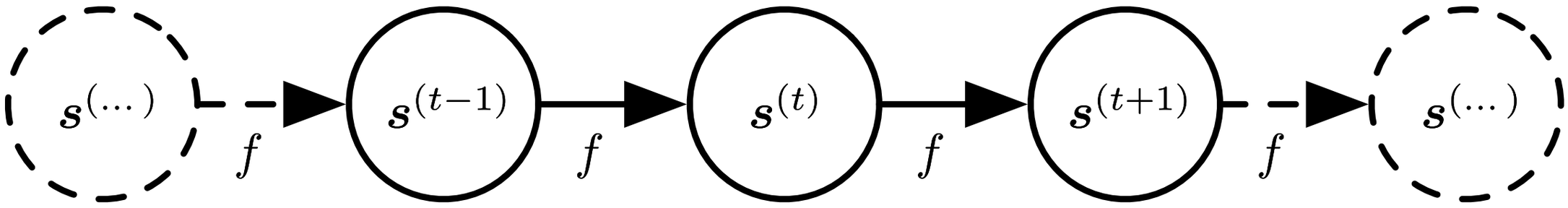
Sequence Modeling: Recurrent and Recursive Nets

Rupam Acharya

Basics

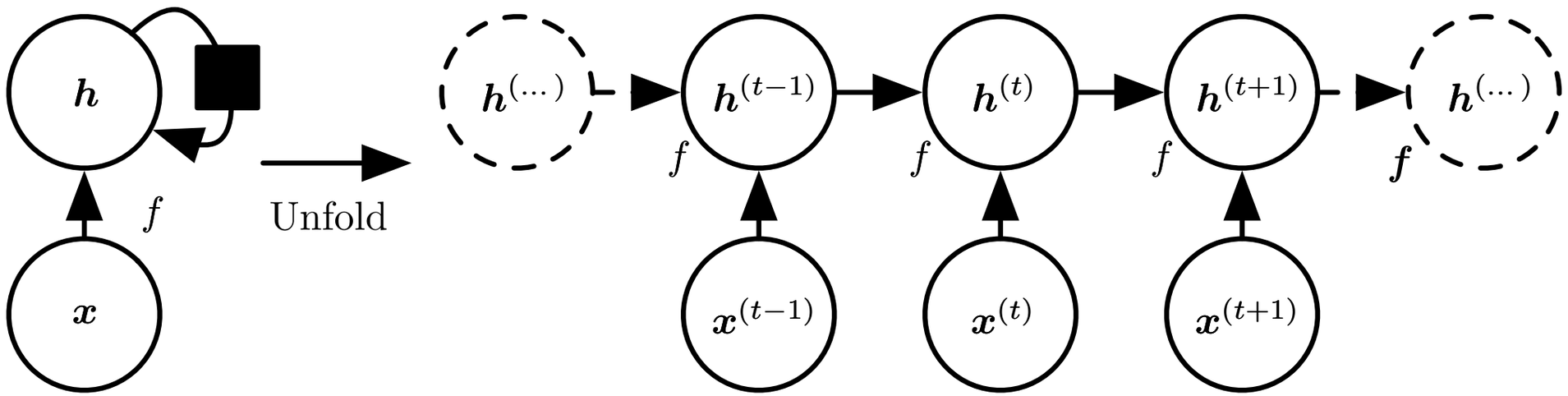
- CNN's are specialized for grid like inputs
- RNN's process sequential inputs
- Can process input of variable size unlike CNN
- Parameter Sharing in RNN helps to work with different length inputs

Classical Dynamical Systems



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

Unfolding Computation Graphs



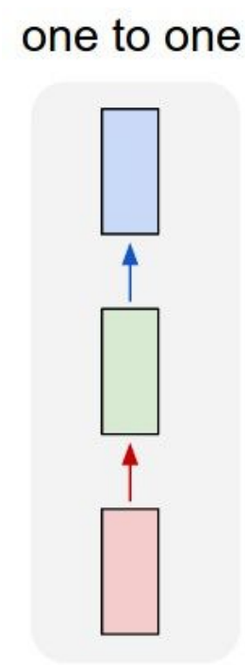
$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta}),$$

- Predict Future from the past
- Network learns lossy summary of the past sequence

Advantage of Unfolding

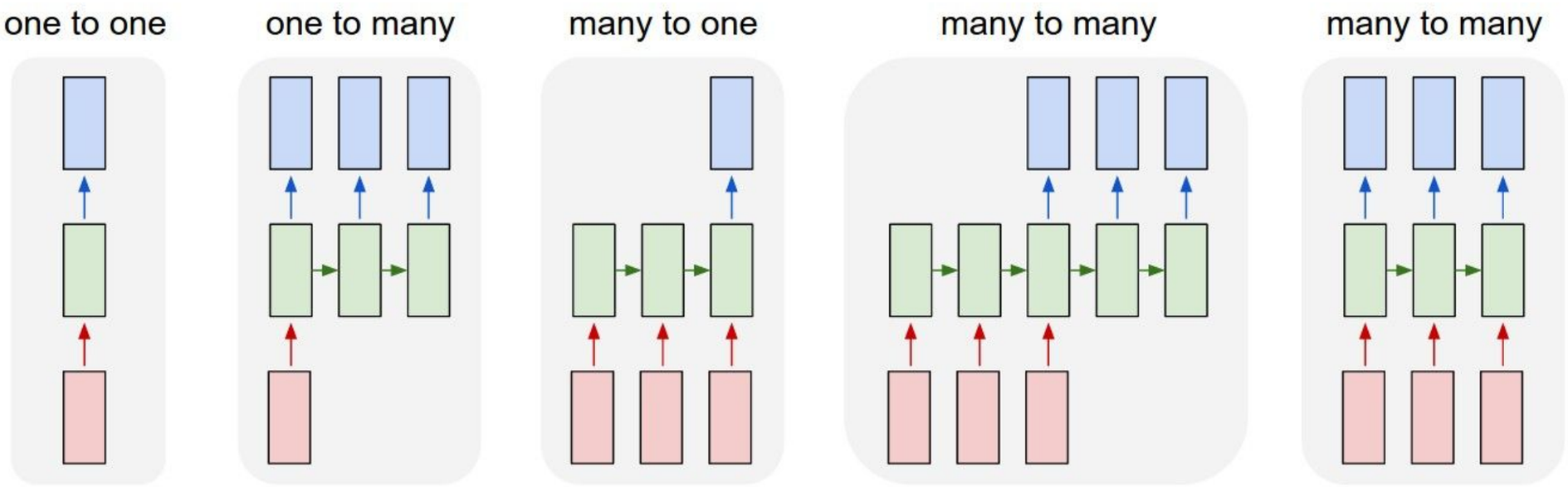
- Regardless of sequence length the input sizee is always same
- Same set of parameter is used over the time
- This motivates the RNN

“Vanilla” Neural Network



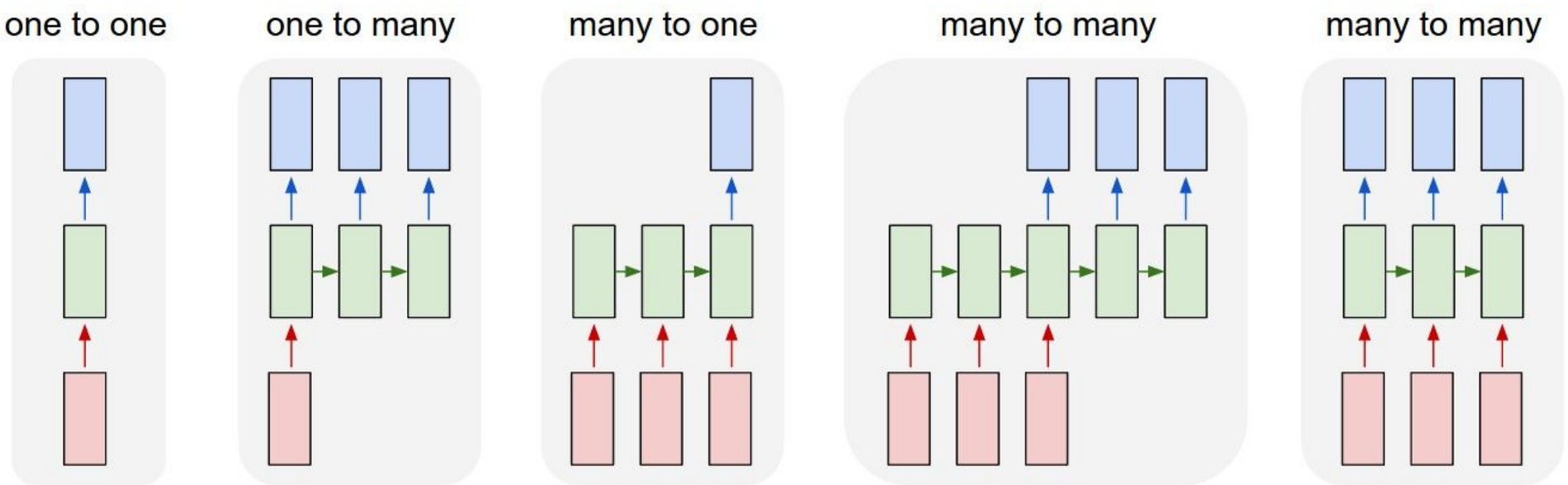
Vanilla Neural Networks

Recurrent Neural Networks: Process Sequences



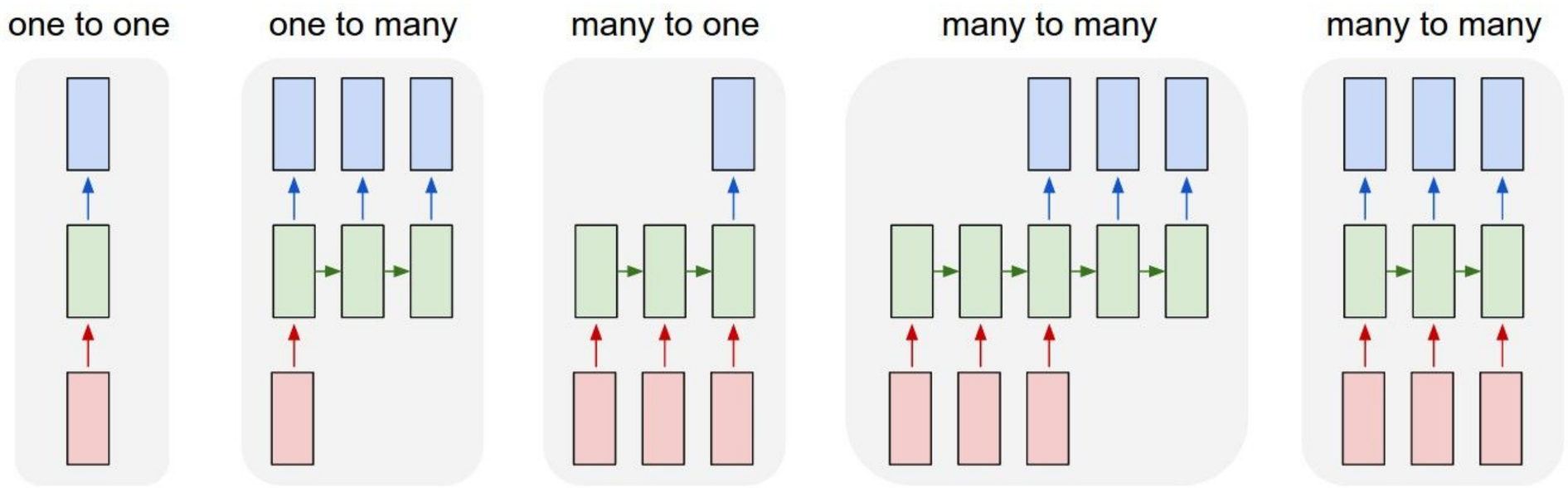
e.g. **Image Captioning**
image -> sequence of words

Recurrent Neural Networks: Process Sequences



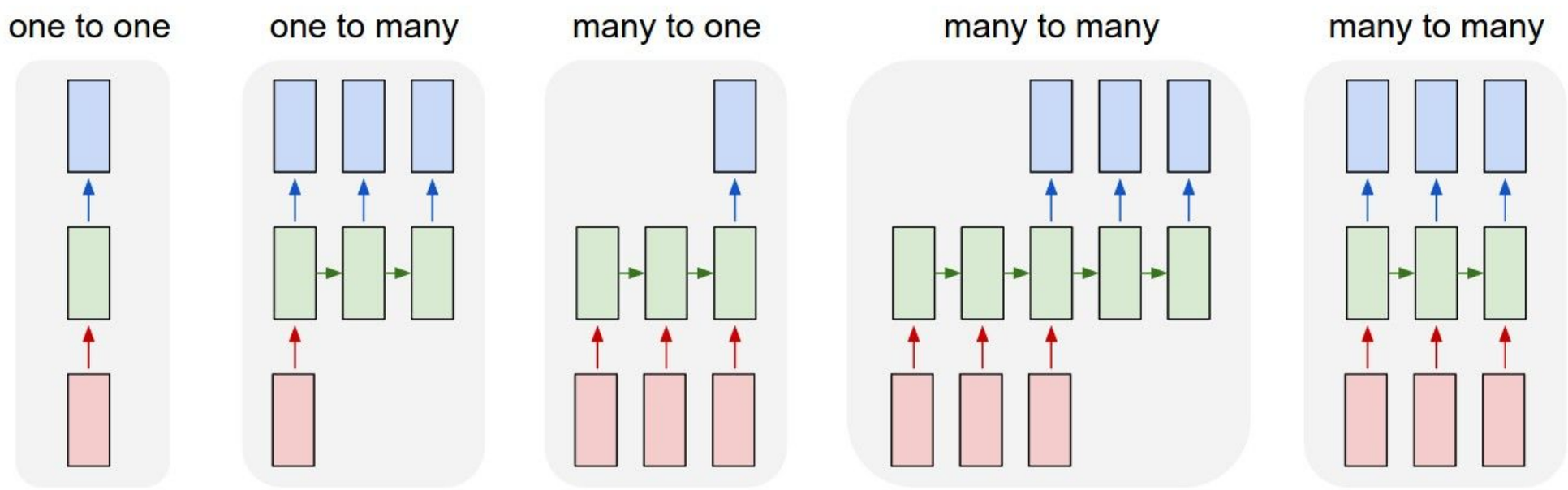
e.g. **Sentiment Classification**
sequence of words -> sentiment

Recurrent Neural Networks: Process Sequences



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

Recurrent Neural Networks: Process Sequences



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

Different Design Pattern

- Recurrent networks that produce an output at each time step and have recurrent connections between hidden units, illustrated in figure 10.3.
- Recurrent networks that produce an output at each time step and have recurrent connections only from the output at one time step to the hidden units at the next time step, illustrated in figure 10.4
- Recurrent networks with recurrent connections between hidden units, that read an entire sequence and then produce a single output, illustrated in figure 10.5.

Recurrent Hidden Units

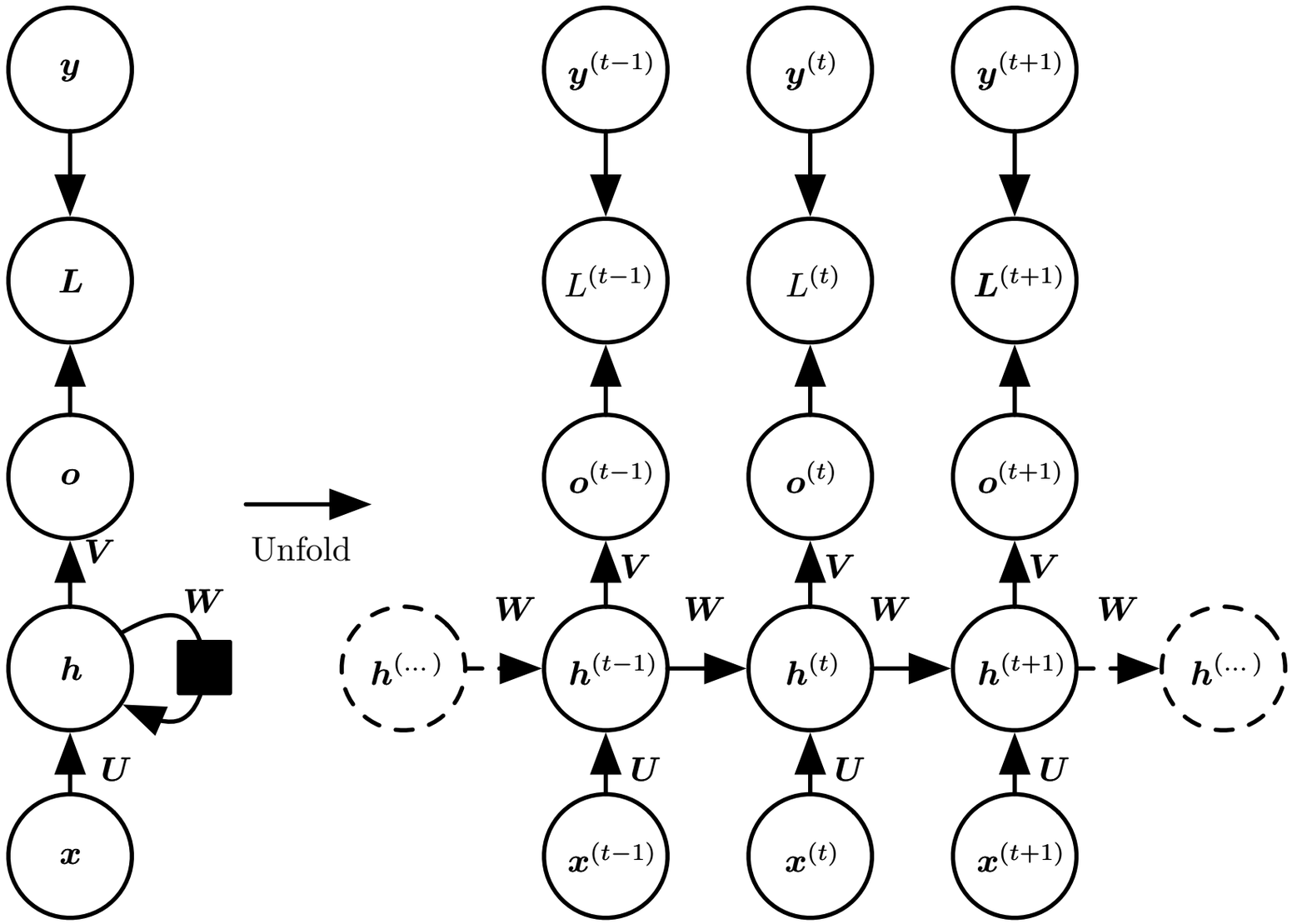


Figure 10.3

Parameter Setting

$t = 1$ to $t = \tau$, we apply the following update equations:

$$\boldsymbol{a}^{(t)} = \boldsymbol{b} + \boldsymbol{W}\boldsymbol{h}^{(t-1)} + \boldsymbol{U}\boldsymbol{x}^{(t)} \tag{10.8}$$

$$\boldsymbol{h}^{(t)} = \tanh(\boldsymbol{a}^{(t)}) \tag{10.9}$$

$$\boldsymbol{o}^{(t)} = \boldsymbol{c} + \boldsymbol{V}\boldsymbol{h}^{(t)} \tag{10.10}$$

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}(\boldsymbol{o}^{(t)}) \tag{10.11}$$

$$L\left(\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(\tau)}\}, \{\boldsymbol{y}^{(1)}, \ldots, \boldsymbol{y}^{(\tau)}\}\right) \tag{10.12}$$

$$= \sum_t L^{(t)} \tag{10.13}$$

$$= -\sum_t \log p_{\text{model}}\left(y^{(t)} \mid \{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(t)}\}\right), \tag{10.14}$$

- Runtime is $O(\tau)$ and cannot be reduced

Recurrence through only the Output

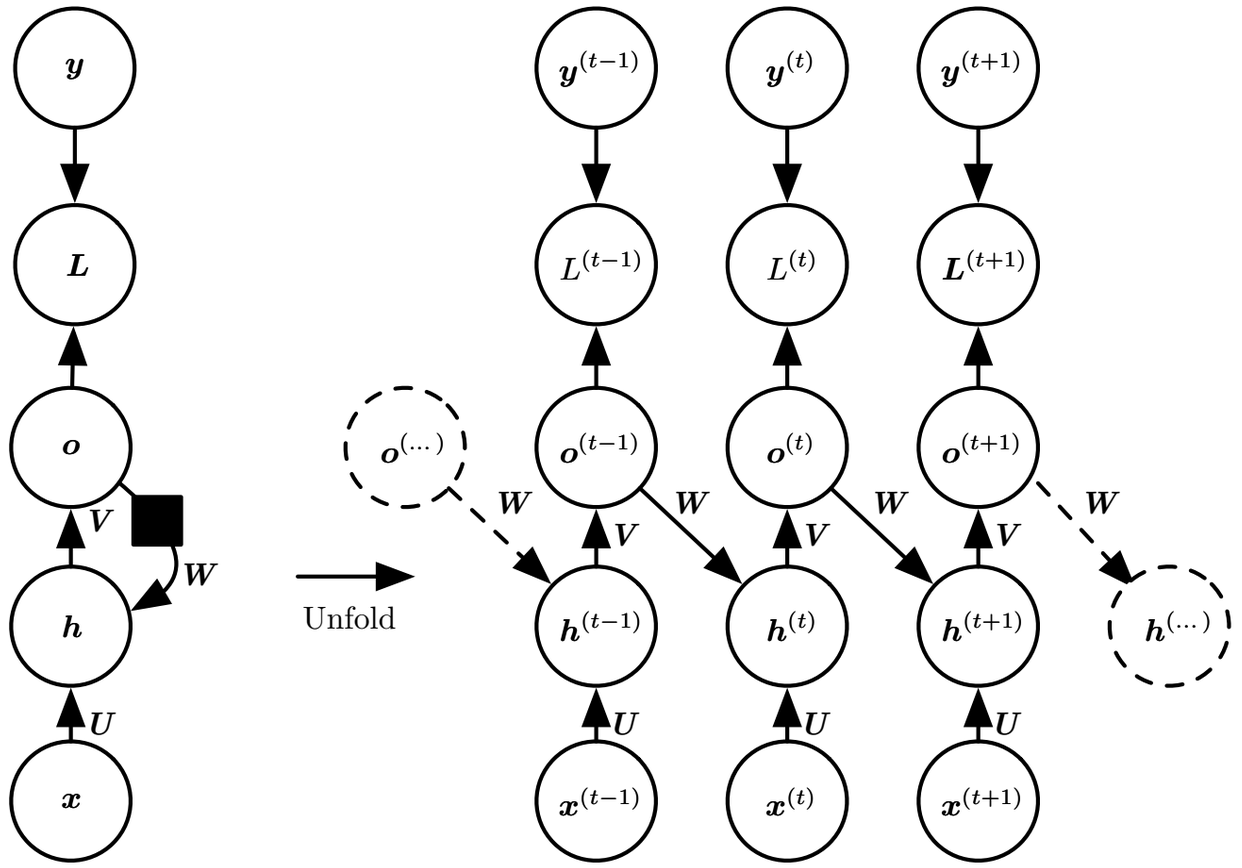


Figure 10.4

- Feedback connection from output to hidden
- Not as strong as hidden to hidden connection

(Goodfellow 2016)

Output to Hidden Connection

- Cannot simulate Universal Turing Machine
- Requires output to capture past information which is not the case
- Advantage is training can be parallelized
- Can be trained using **teacher forcing**

Teacher Forcing

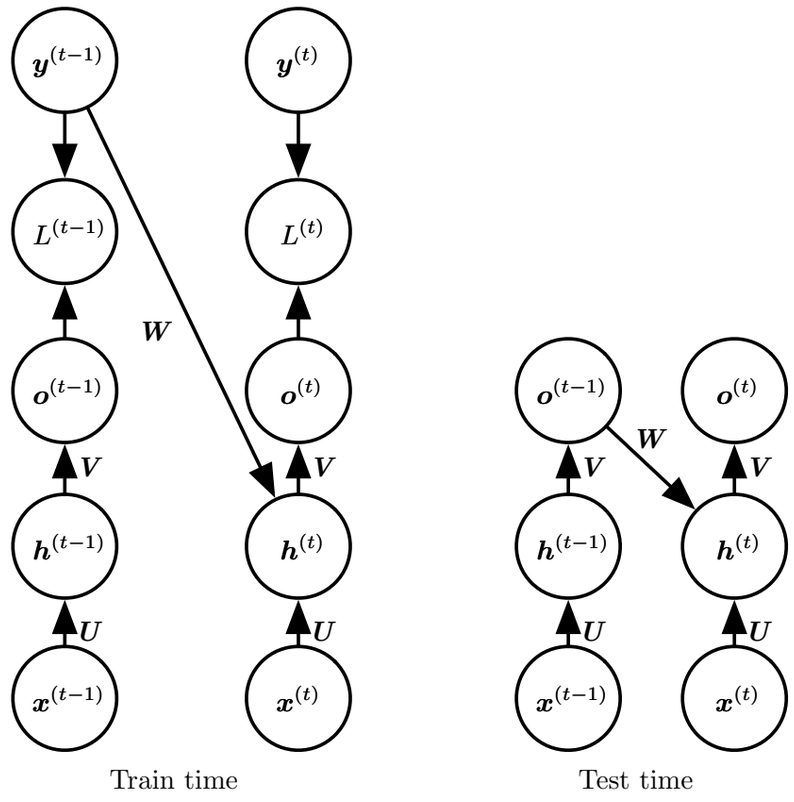


Figure 10.6: Illustration of teacher forcing. Teacher forcing is a training technique that is applicable to RNNs that have connections from their output to their hidden states at the next time step. *(Left)*At train time, we feed the *correct output* $\boldsymbol{y}^{(t)}$ drawn from the train set as input to $\boldsymbol{h}^{(t+1)}$. *(Right)*When the model is deployed, the true output is generally not known. In this case, we approximate the correct output $\boldsymbol{y}^{(t)}$ with the model's output $\boldsymbol{o}^{(t)}$, and feed the output back into the model.

$$\log p\left(\boldsymbol{y}^{(1)}, \boldsymbol{y}^{(2)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right)$$
$$= \log p\left(\boldsymbol{y}^{(2)} \mid \boldsymbol{y}^{(1)}, \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right) + \log p\left(\boldsymbol{y}^{(1)} \mid \boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}\right)$$

(Goodfellow 2016)

Sequence Input, Single Output

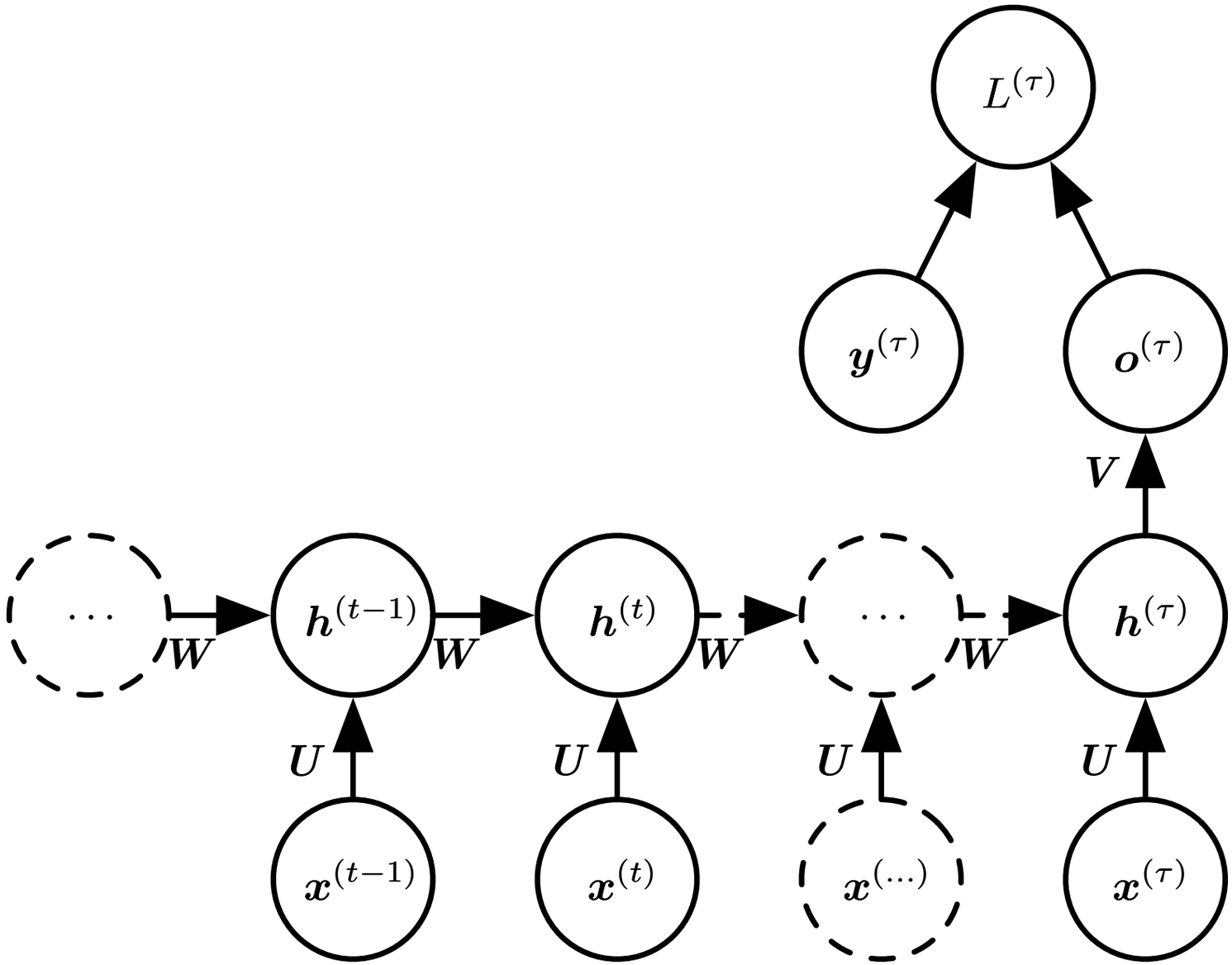


Figure 10.5

Computing Gradient in RNN

$$\nabla_{\boldsymbol{c}} L = \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} \right)^\top \nabla_{\boldsymbol{o}^{(t)}} L$$

$$\nabla_{\boldsymbol{b}} L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}} \right)^\top \nabla_{\boldsymbol{h}^{(t)}} L$$

$$\nabla_{\boldsymbol{V}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_{\boldsymbol{V}} o_i^{(t)}$$

$$\nabla_{\boldsymbol{W}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)}$$

$$\nabla_{\boldsymbol{U}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)}$$

- Need to Compute

Computing Gradient in RNN

$$\frac{\partial L}{\partial L^{(t)}} = 1.$$

$$\left(\nabla_{\boldsymbol{o}^{(t)}} L\right)_i = \frac{\partial L}{\partial o_i^{(t)}} = \frac{\partial L}{\partial L^{(t)}} \frac{\partial L^{(t)}}{\partial o_i^{(t)}} = \hat{y}_i^{(t)} - \mathbf{1}_{i,y^{(t)}}.$$

$$\nabla_{\boldsymbol{h}^{(\tau)}} L = \boldsymbol{V}^\top \nabla_{\boldsymbol{o}^{(\tau)}} L.$$

$$\begin{aligned}\nabla_{\boldsymbol{h}^{(t)}} L &= \left(\frac{\partial \boldsymbol{h}^{(t+1)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top \left(\nabla_{\boldsymbol{h}^{(t+1)}} L\right) + \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{h}^{(t)}}\right)^\top \left(\nabla_{\boldsymbol{o}^{(t)}} L\right) \\ &= \boldsymbol{W}^\top \left(\nabla_{\boldsymbol{h}^{(t+1)}} L\right) \operatorname{diag}\left(1 - \left(\boldsymbol{h}^{(t+1)}\right)^2\right) + \boldsymbol{V}^\top \left(\nabla_{\boldsymbol{o}^{(t)}} L\right)\end{aligned}$$

- This is what we have

Computing Gradient in RNN

$$\nabla_{\boldsymbol{c}} L = \sum_t \left(\frac{\partial \boldsymbol{o}^{(t)}}{\partial \boldsymbol{c}} \right)^\top \nabla_{\boldsymbol{o}^{(t)}} L = \sum_t \nabla_{\boldsymbol{o}^{(t)}} L \tag{10.22}$$

$$\nabla_{\boldsymbol{b}} L = \sum_t \left(\frac{\partial \boldsymbol{h}^{(t)}}{\partial \boldsymbol{b}^{(t)}} \right)^\top \nabla_{\boldsymbol{h}^{(t)}} L = \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)} \right)^2 \right) \nabla_{\boldsymbol{h}^{(t)}} L \tag{10.23}$$

$$\nabla_{\boldsymbol{V}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial o_i^{(t)}} \right) \nabla_{\boldsymbol{V}} o_i^{(t)} = \sum_t \left(\nabla_{\boldsymbol{o}^{(t)}} L \right) \boldsymbol{h}^{(t)\top} \tag{10.24}$$

$$\nabla_{\boldsymbol{W}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{W}^{(t)}} h_i^{(t)} \tag{10.25}$$

$$= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)} \right)^2 \right) \left(\nabla_{\boldsymbol{h}^{(t)}} L \right) \boldsymbol{h}^{(t-1)\top} \tag{10.26}$$

$$\nabla_{\boldsymbol{U}} L = \sum_t \sum_i \left(\frac{\partial L}{\partial h_i^{(t)}} \right) \nabla_{\boldsymbol{U}^{(t)}} h_i^{(t)} \tag{10.27}$$

$$= \sum_t \mathrm{diag}\left(1 - \left(\boldsymbol{h}^{(t)} \right)^2 \right) \left(\nabla_{\boldsymbol{h}^{(t)}} L \right) \boldsymbol{x}^{(t)\top} \tag{10.28}$$

- This is what we get

Fully Connected Graphical Model

- Chain Rule of Conditional Probability

$$P(\mathbb{Y}) = P(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(\tau)}) = \prod_{t=1}^{\tau} P(\mathbf{y}^{(t)} \mid \mathbf{y}^{(t-1)}, \mathbf{y}^{(t-2)}, \ldots, \mathbf{y}^{(1)})$$

- The negative log-likelihood according to this model is

$$L = \sum_t L^{(t)}$$

where

$$L^{(t)} = -\log P(\mathrm{y}^{(t)} = y^{(t)} \mid y^{(t-1)}, y^{(t-2)}, \ldots, y^{(1)}).$$

Fully Connected Graphical Model

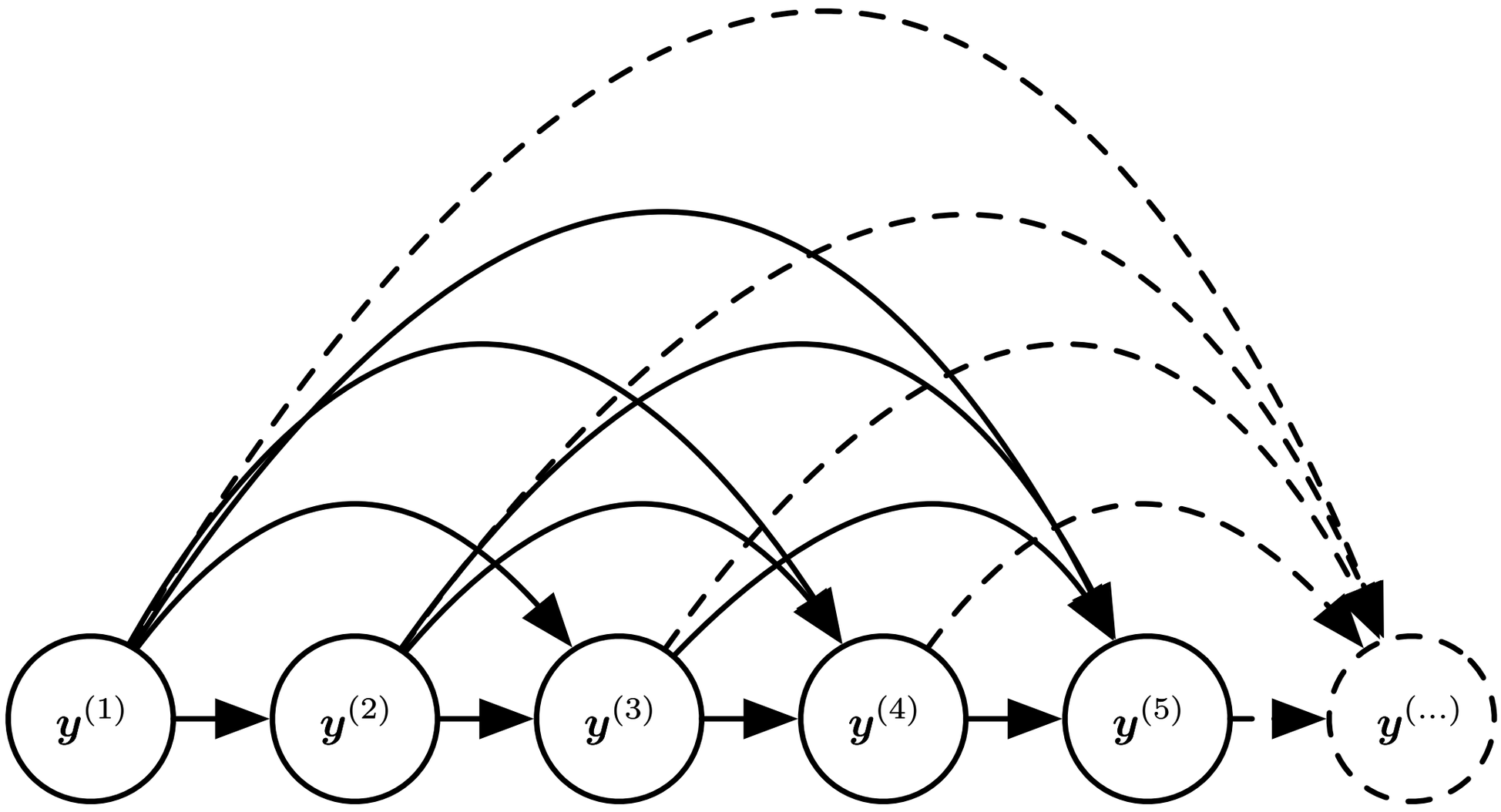
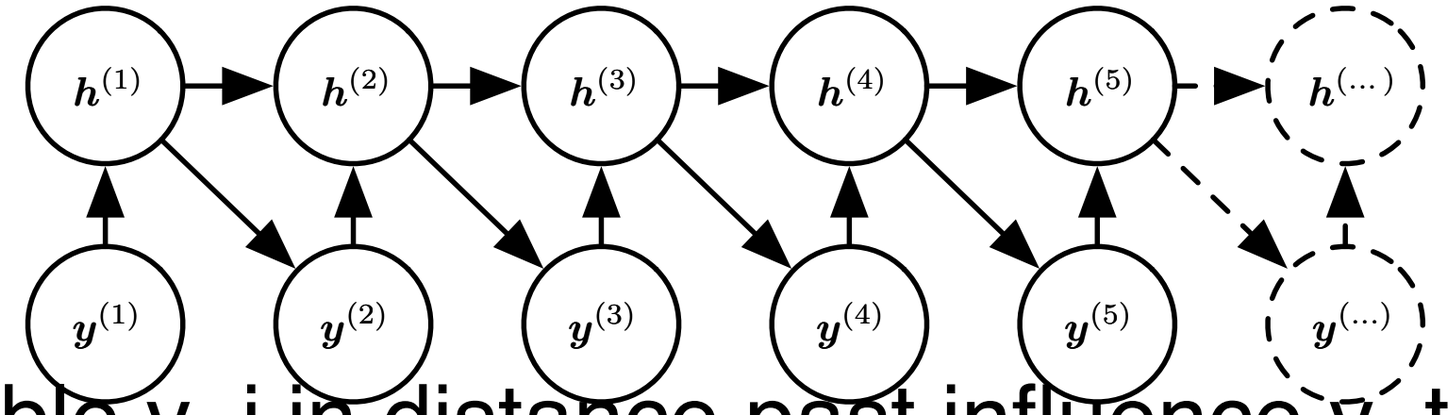


Figure 10.7

RNN Graphical Model



- A variable y_i in distance past influence y_t via h

- The model can be efficiently parameterized by using same conditional probability in each time step

- When the variables are observed the joint distribution can be evaluated efficiently

- Hard to predict missing values and optimizing the parameters

Sampling from Model

- Simply sample from the conditional distribution
- But length of the sequence is not known!!
- If output is a symbol from a vocabulary then add an eos symbol to stop the sampling process
- Generate Bernouli output to determine whether to continue generation or not

Vector to Sequence

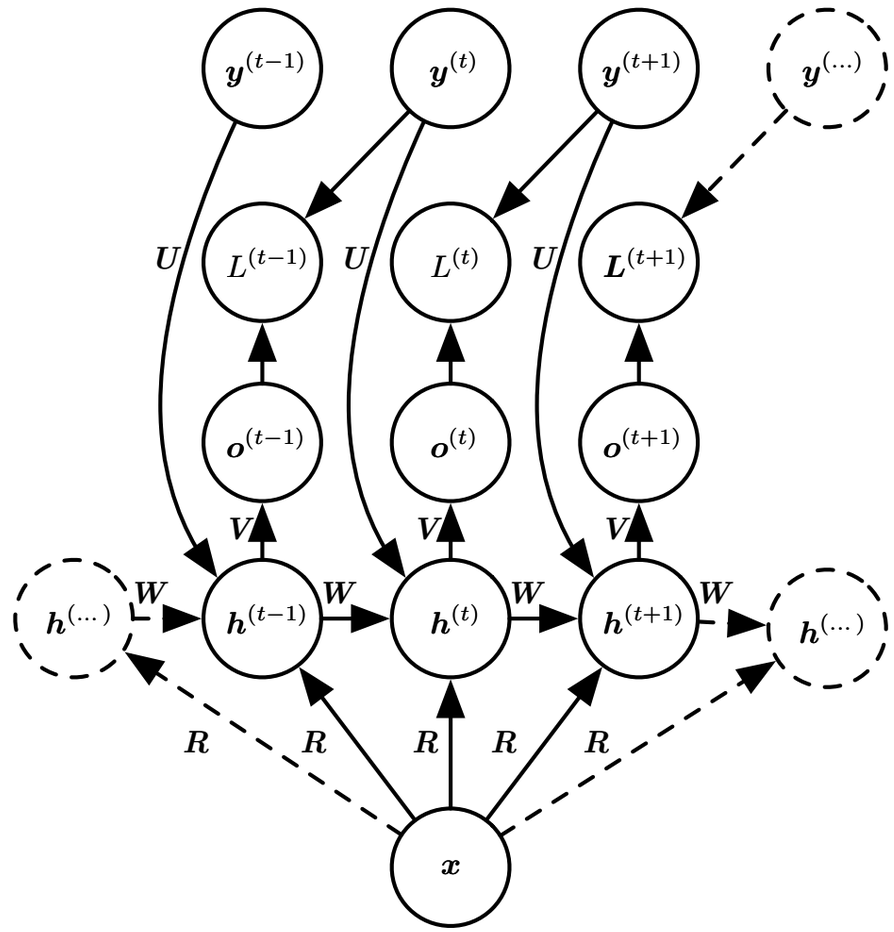


Figure 10.9: An RNN that maps a fixed-length vector $\boldsymbol{x}$ into a distribution over sequences $\mathbf{Y}$. This RNN is appropriate for tasks such as image captioning, where a single image is used as input to a model that then produces a sequence of words describing the image. Each element $\boldsymbol{y}^{(t)}$ of the observed output sequence serves both as input (for the current time step) and, during training, as target (for the previous time step).

Hidden and Output Recurrence

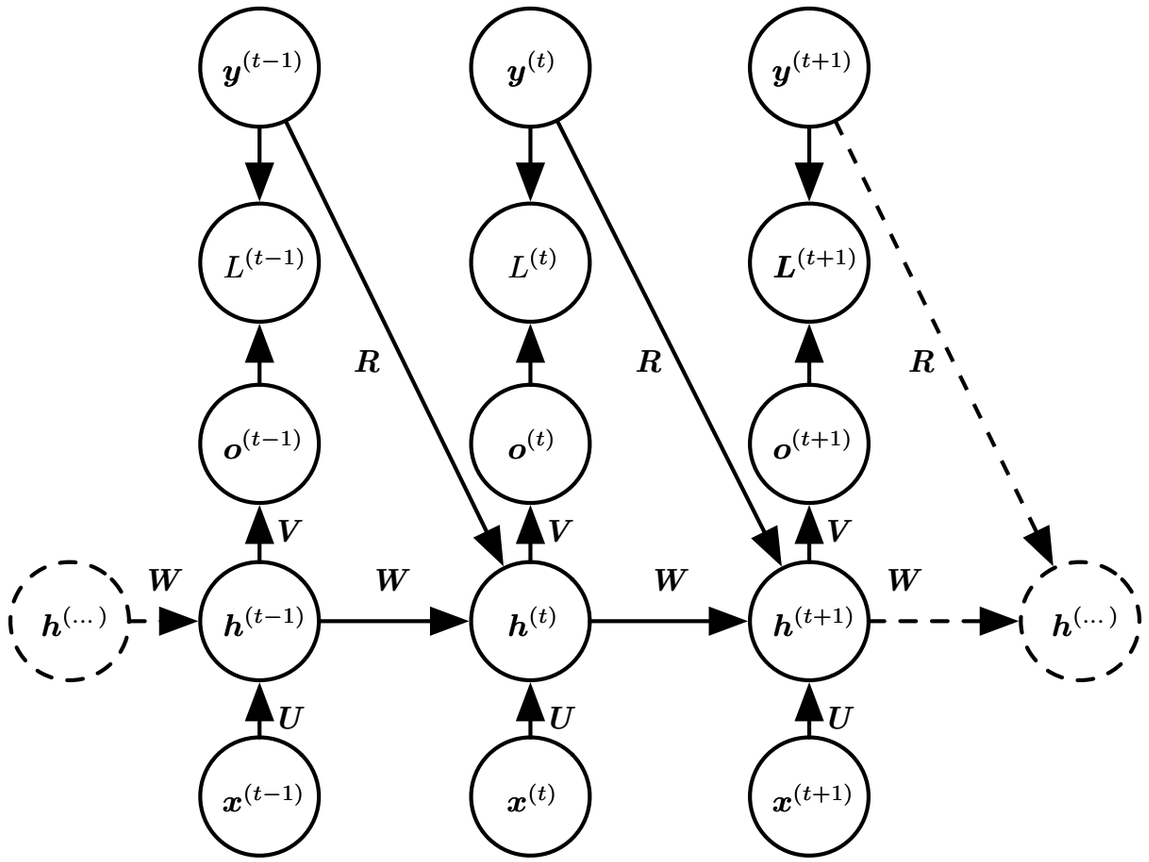


Figure 10.10: A conditional recurrent neural network mapping a variable-length sequence of $\boldsymbol{x}$ values into a distribution over sequences of $\boldsymbol{y}$ values of the same length. Compared to figure 10.3, this RNN contains connections from the previous output to the current state. These connections allow this RNN to model an arbitrary distribution over sequences of $\boldsymbol{y}$ given sequences of $\boldsymbol{x}$ of the same length. The RNN of figure 10.3 is only able to represent distributions in which the $\boldsymbol{y}$ values are conditionally independent from each other given the $\boldsymbol{x}$ values.

(Goodfellow 2016)