
# On Finding the Structure in Time

Alexander G. Ororbia II

Introduction to Machine Learning

CSCI-635

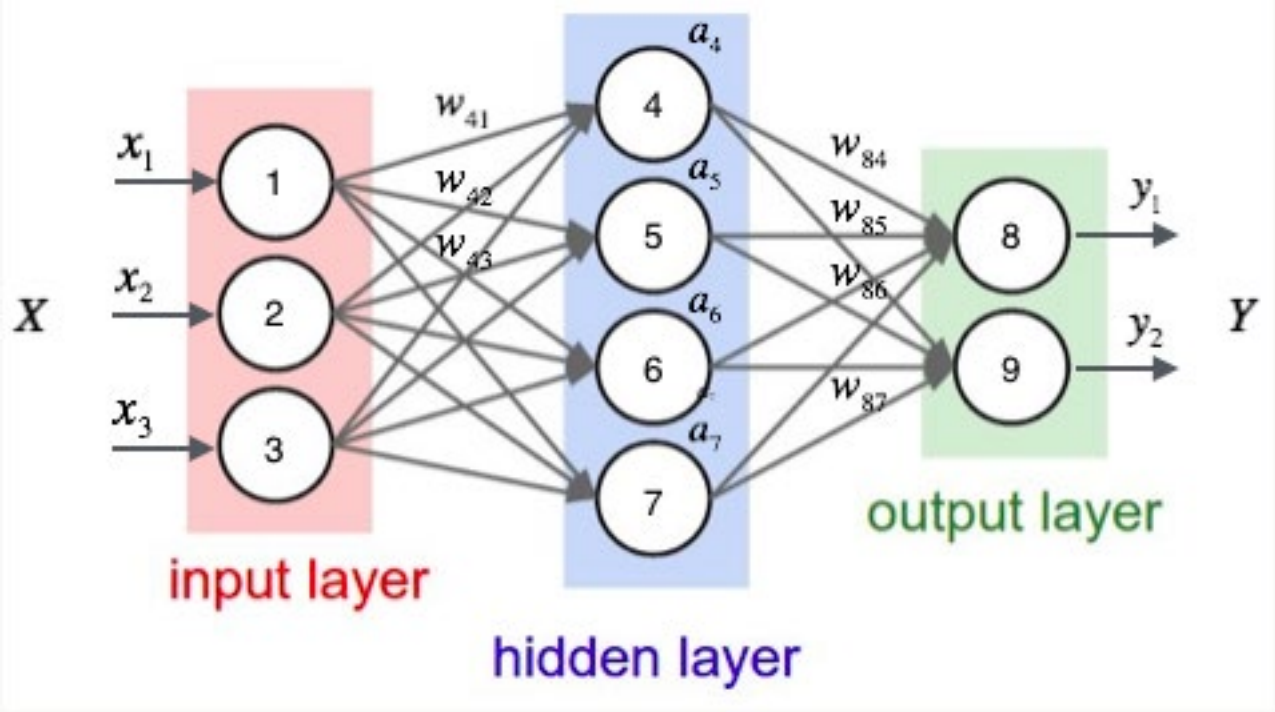
2/11/2025

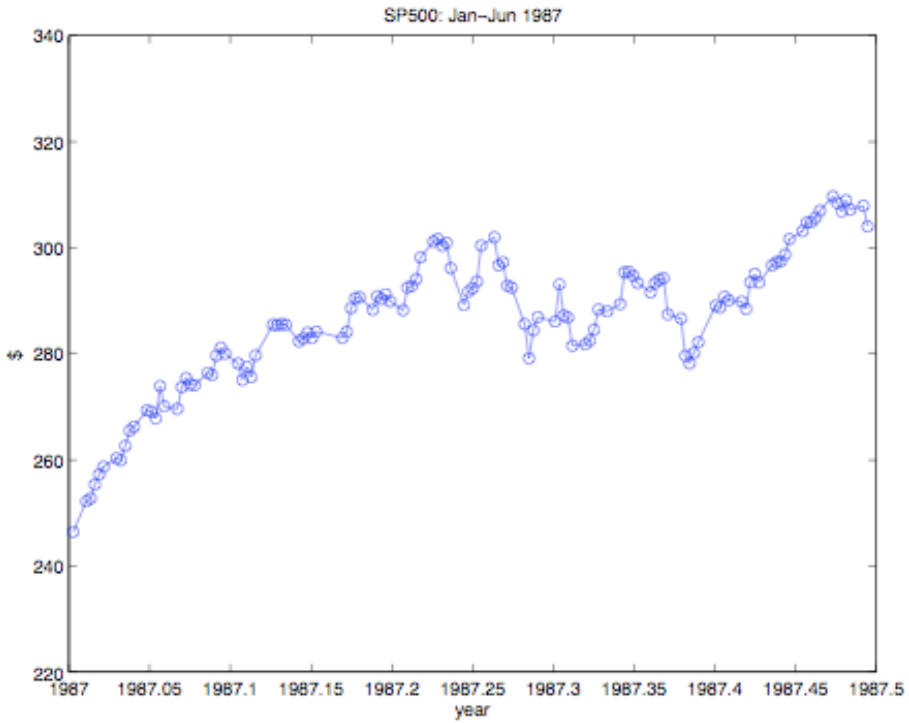
***Companion reading:***

Chapter 10 of Deep Learning textbook

Without “time funnels” or “temporal queues”…

# Feedforward Neural Networks (FNNs)



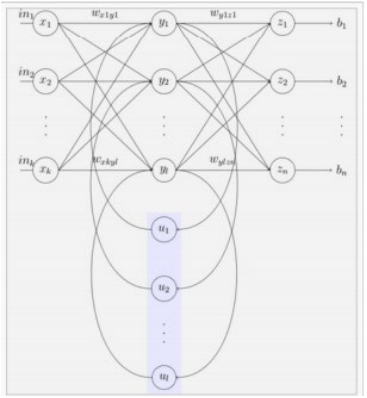



# Why might we need something other than FNNs?

# RNNs process sequential data

- Recurrent Neural Networks are a family of neural networks for processing sequential data
- RNN and CNN are both specialized architectures
  - RNN is specialized for processing a sequence of values $x^{(1)},..,x^{(\tau)}$

    Just as CNN is specialized for processing a grid of values such as an image
  - RNNs can scale to much longer sequences than would be impractical for networks without sequence-based specialization
  - RNNs can also process variable-length sequences



Just as a CNN can scale to images with large width/height and process variable size images

# The Problem of Long-Term Dependencies

One appeal of RNNs is  idea that they might be able to connect **previous information** to **present task**, e.g., using previous video frames might inform the understanding of present frame.
If RNNs could do this, they would be extremely useful!

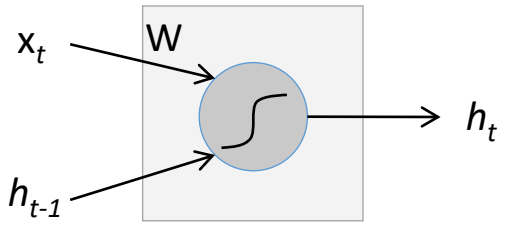
But *can* they? **It depends.**

# The Problem of Long-Term Dependencies

Sometimes, only need to look at recent information to perform present task.

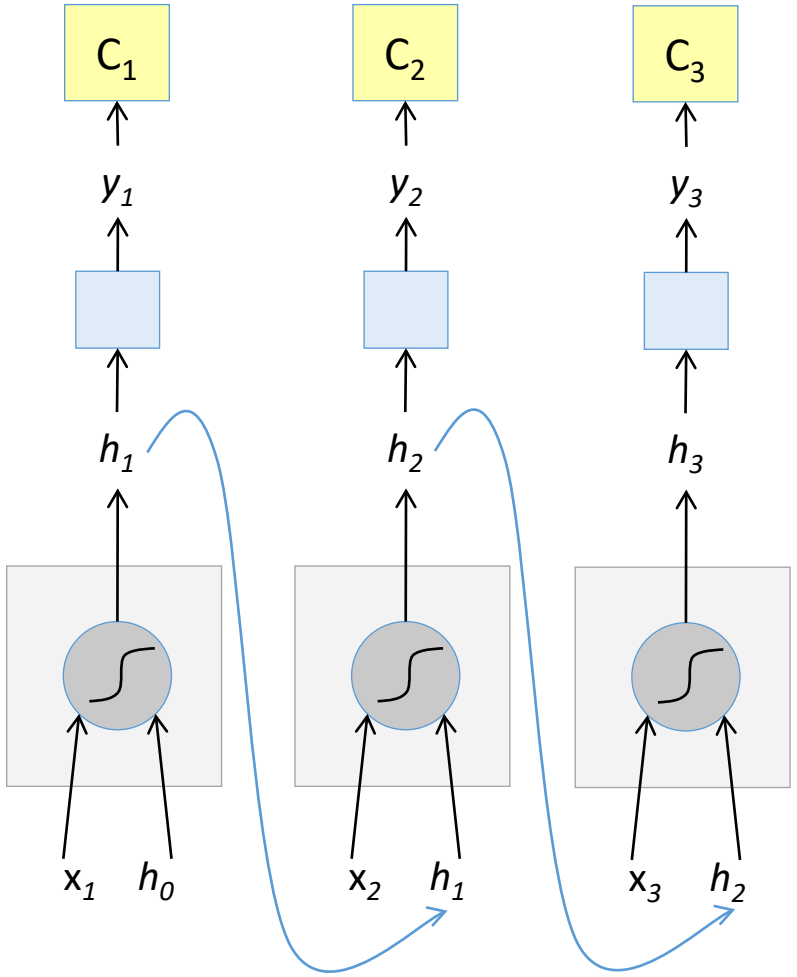
For example, consider a language model trying to predict the next word based on the previous ones.
If we are trying to predict the last word in "*the clouds are in the sky*" we do not need any further context – next word is going to be sky. In these cases, where gap between **relevant information and place that prediction is required is small**, RNNs can learn to use past information

# The RNN Building Block




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

# The RNN, Unrolled




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

$$y_t = \mathrm{F}(h_t)$$

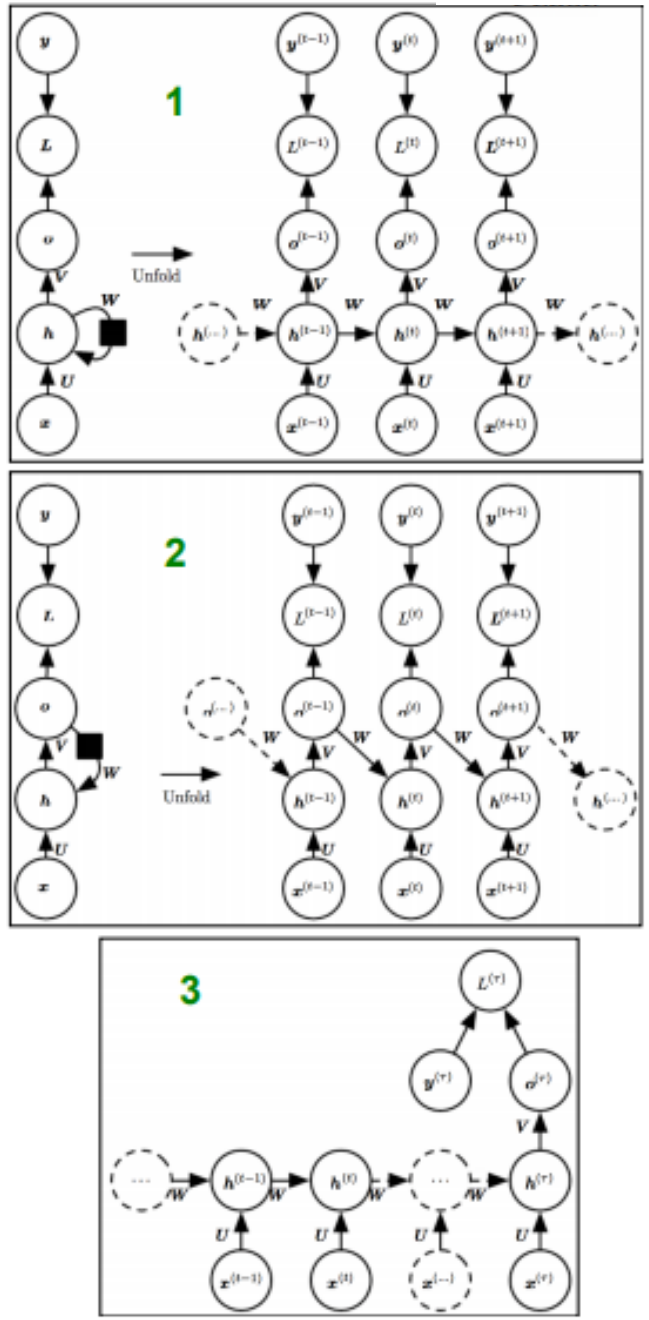
$$C_t = \mathrm{Loss}(y_t, \mathrm{GT}_t)$$

# RNN operating on a sequence

- RNNs operate on a sequence that contain vector $\boldsymbol{x}^{(t)}$ with time step index $t$, ranging from $1$ to $\tau$
  - Sequence: $\boldsymbol{x}^{(1)},\ldots,\boldsymbol{x}^{(\tau)}$
  - RNNs operate on minibatches of sequences of length $\tau$
- Some remarks about sequences
  - The steps need not refer to passage of time in the real world
  - RNNs can be applied in two-dimensions across spatial data such as image
  - Even when applied to time sequences, network may have connections going backwards in time, provided entire sequence is observed before it is provided to network

# Three design patterns of RNNs

1. Produce output at each time step and have recurrent connections between hidden units
2. Produce output at each time step and have recurrent connections only from output at one time step to hidden units at next time step
   - Less powerful than (1) but easier to train
     - Each step can be trained in isolation
     - Greater parallelization during training
3. Recurrent connections between hidden units that read an entire input sequence and produce a single output
   - Can summarize a sequence and produce a fixed size representation for further processing



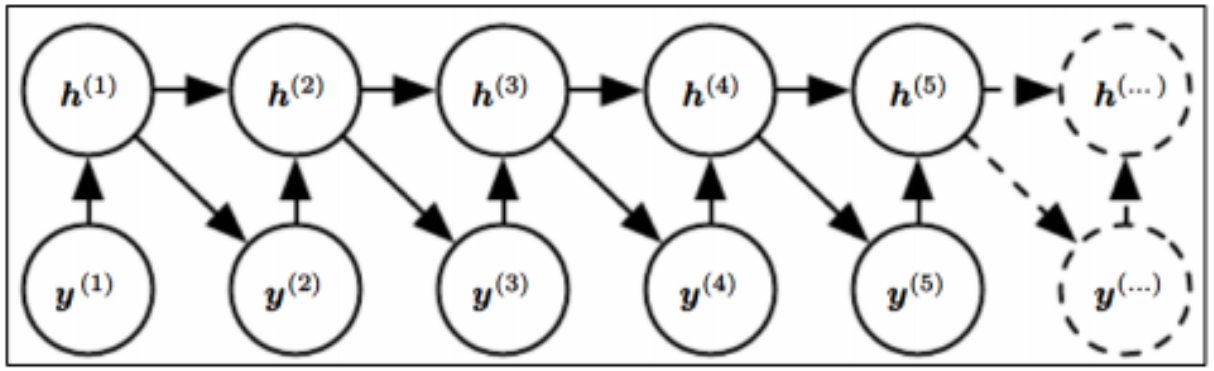
# Computing gradient in RNN using BPTT

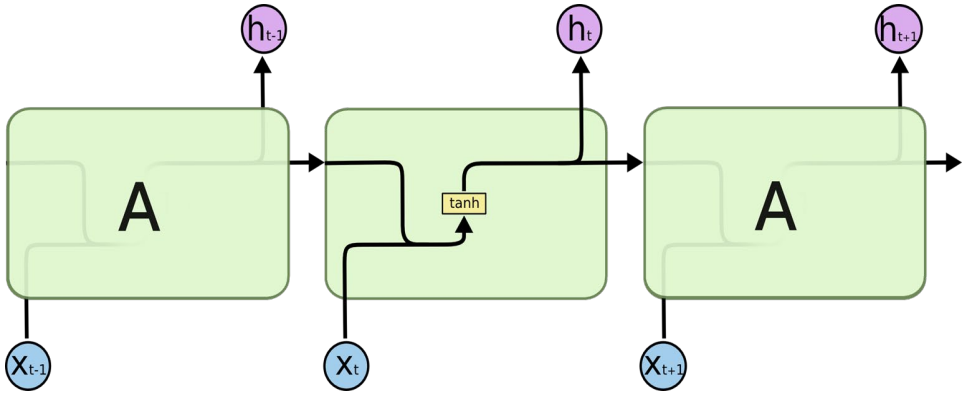
- Computing the gradient through an RNN is straightforward
- One simply applies the generalized back-propagation algorithm to the unrolled computational graph.
- No specialized algorithms are necessary.
- Gradients obtained by back-propagation may then be used with any general-purpose gradient-based techniques to train an RNN.

# White Board Time!

(Turning MLPs into RNNs)

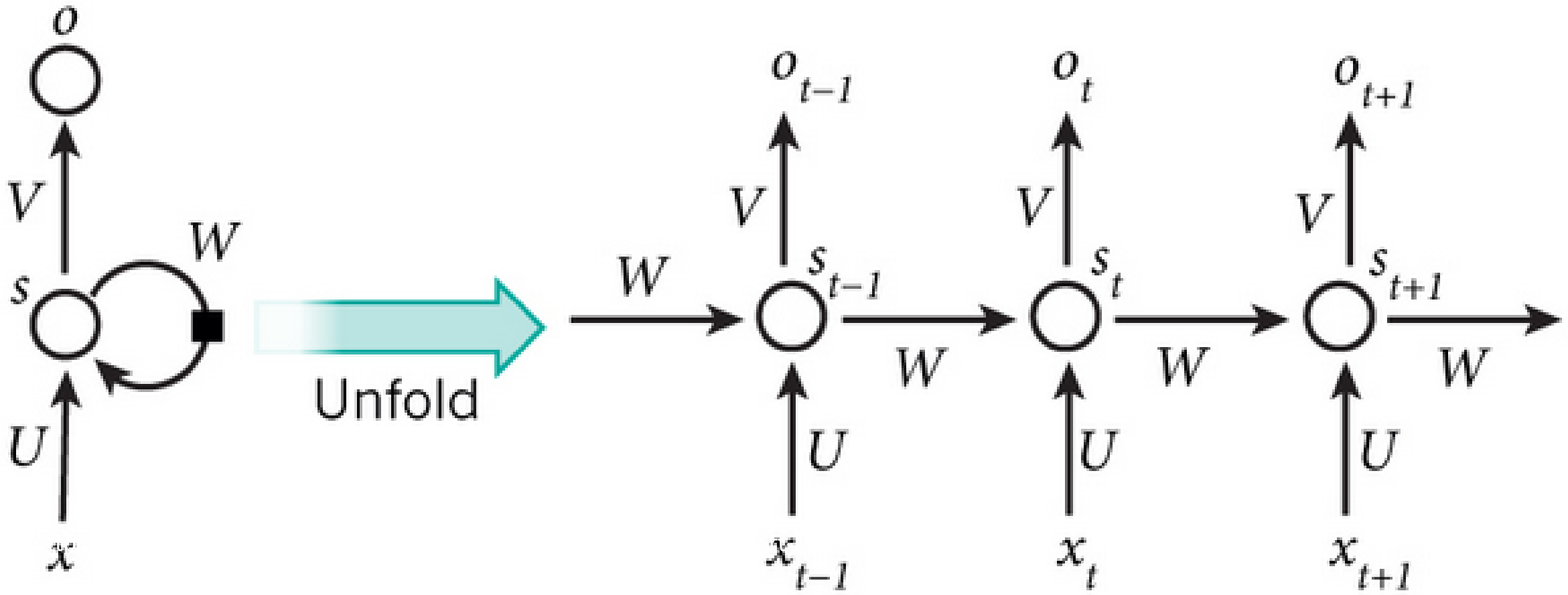

Efficient parameterization based on $h^{(t)}=f(h^{(t-1)},x^{(t)};\theta)$



# Unfolding over Time

Recurrence similar to
- Hidden Markov Model (HMM)
- Kalman Filter (KF, EKF, UKF)

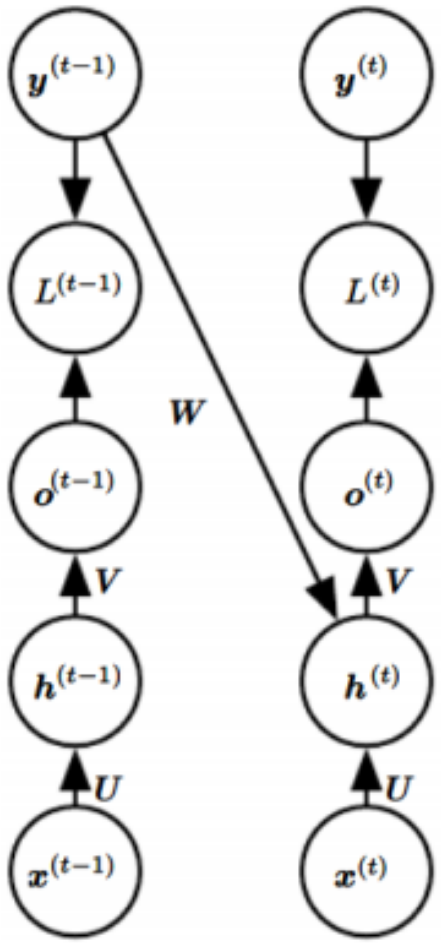


# Illustration of Teacher Forcing

- Teacher Forcing is a training technique applicable to RNNs that have connections from output to hidden states at next time step

## Train time:

We feed the correct output $\boldsymbol{y}^{(t)}$ (from teacher) drawn from the training set as input to $\boldsymbol{h}^{(t+1)}$

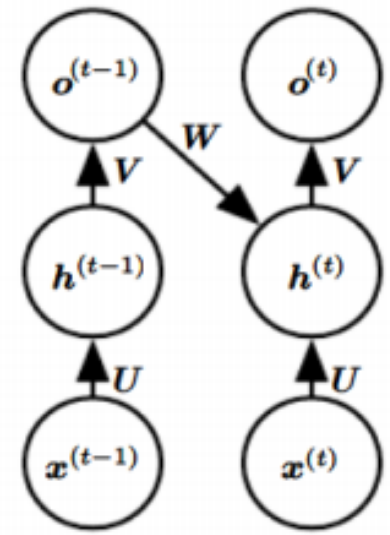



Train time

## Test time:

True output is not known.

We approximate the correct output $\boldsymbol{y}^{(t)}$ with the model's output $\boldsymbol{o}^{(t)}$ and feed the output back to the model

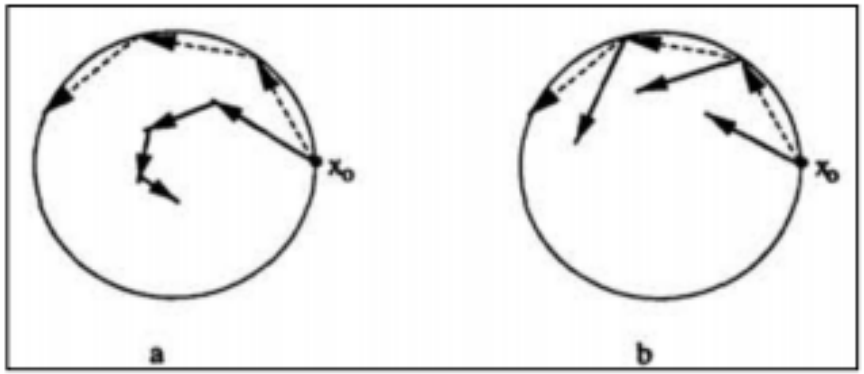



Test time

# Teacher Forcing

Srihari

- Models that have recurrent connections from their outputs leading back to the model may be trained with *teacher forcing*
- Teacher forcing during training means
  - Instead of summing activations from incoming units (possibly erroneous)
  - Each unit sums correct teacher activations as input for the next iteration
- Visualizing effect of teaching forcing
  - imagine that the network is learning to follow a trajectory; it goes astray (because the weights are wrong) but teacher forcing puts the net back on its trajectory by setting the state of all the units to that of teacher's.

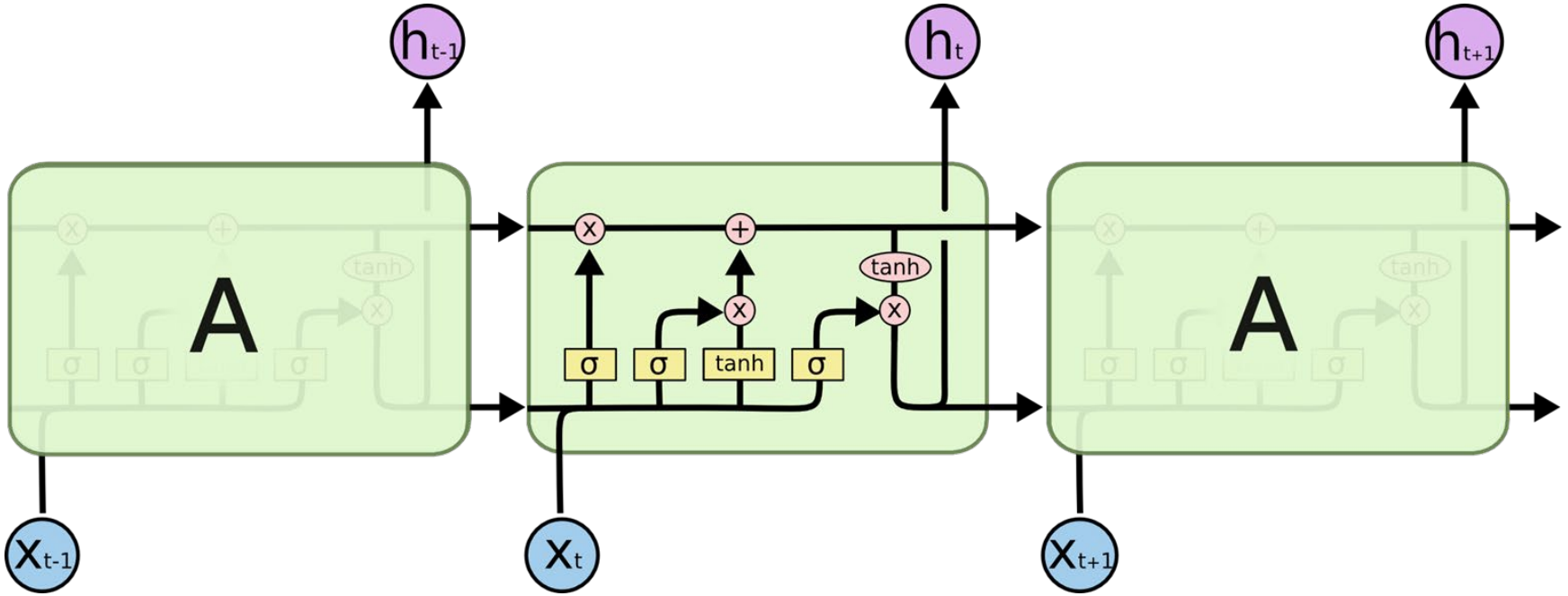



(a) Without teacher forcing, trajectory runs astray (solid lines) while the correct trajectory are the dotted lines

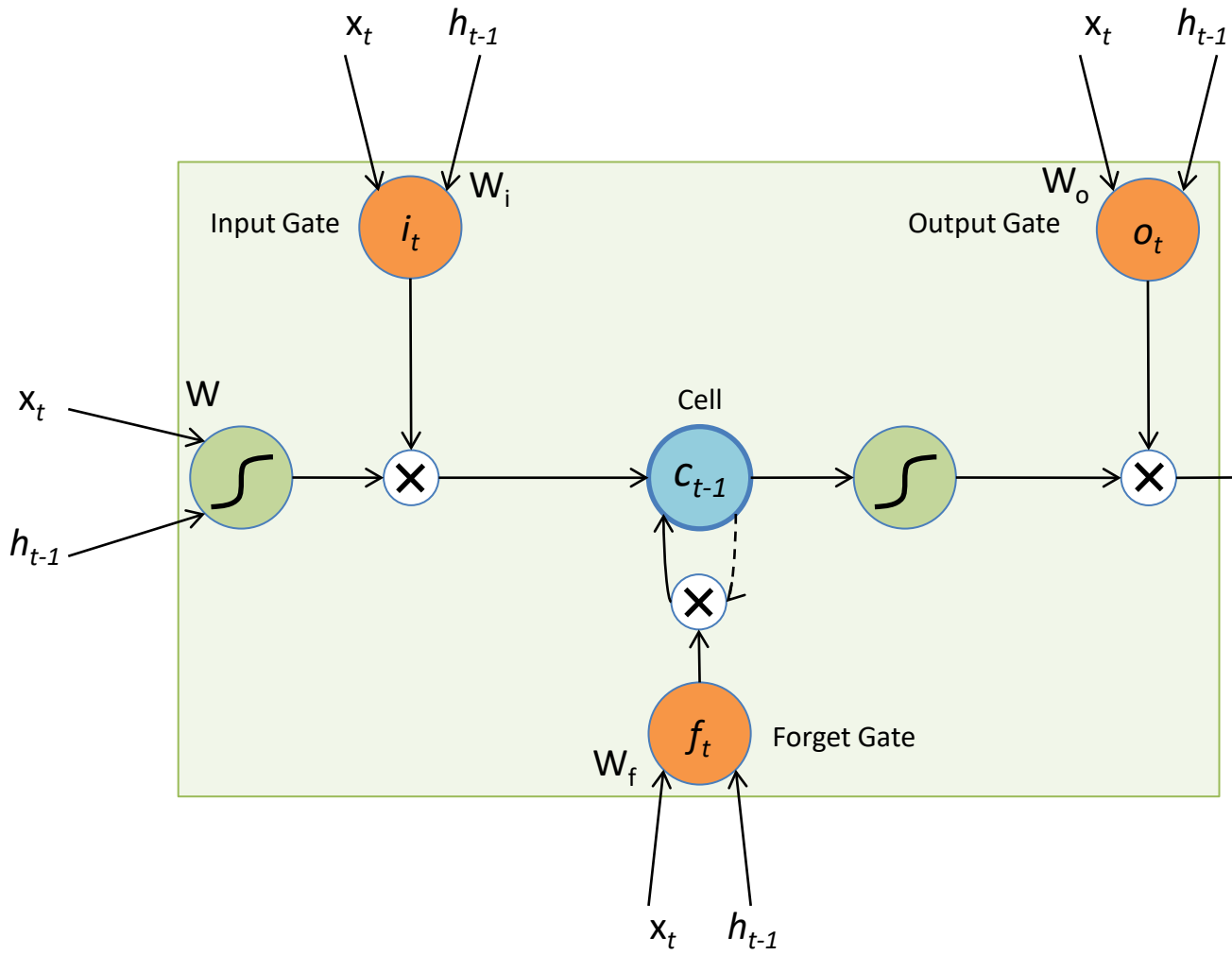
(b) With teacher forcing trajectory corrected at each step

# LSTM Networks

LSTMs also have a chain-like structure, but repeating module has a different structure; **instead of having a single neural network layer, there are four** which interact in a particular way



# The LSTM Building Block




$$f_t = \sigma\left(W_f \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f\right)$$

Similarly for $i_t$, $o_t$

$$c_t = f_t \otimes c_{t-1} + i_t \otimes \tanh W \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix}$$

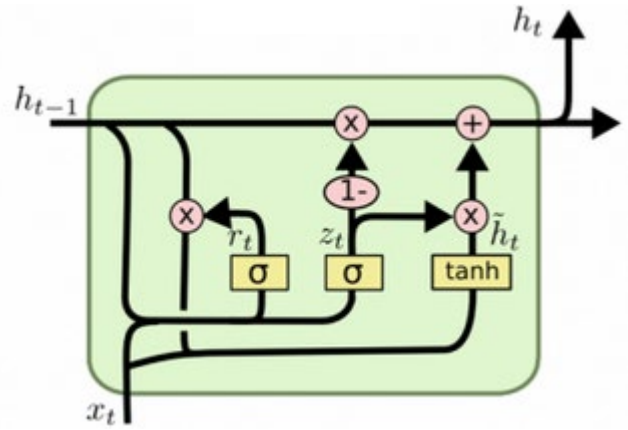
$$h_t = o_t \otimes \tanh c_t$$

* Dashed line indicates time-lag

# Gated Recurrent Unit (GRU)

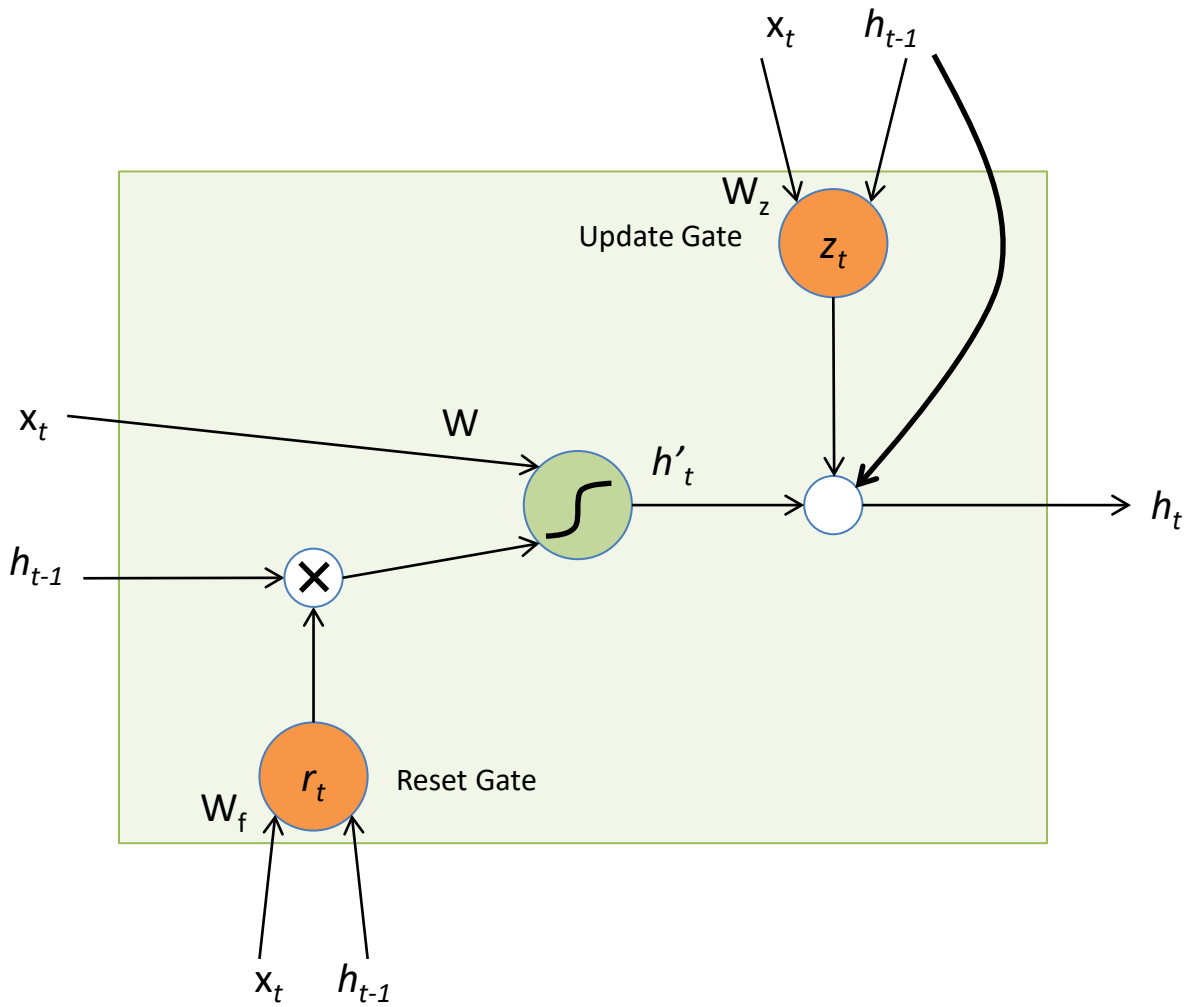
- A simplified version of the LSTM
  - Merges forget and input gate into a single “update” gate
  - Merges cell with hidden state

- Has fewer parameters than an LSTM and was shown to outperform LSTM on *some* tasks



# The GRU Building Block

⊗ : ***Hadamard product***




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

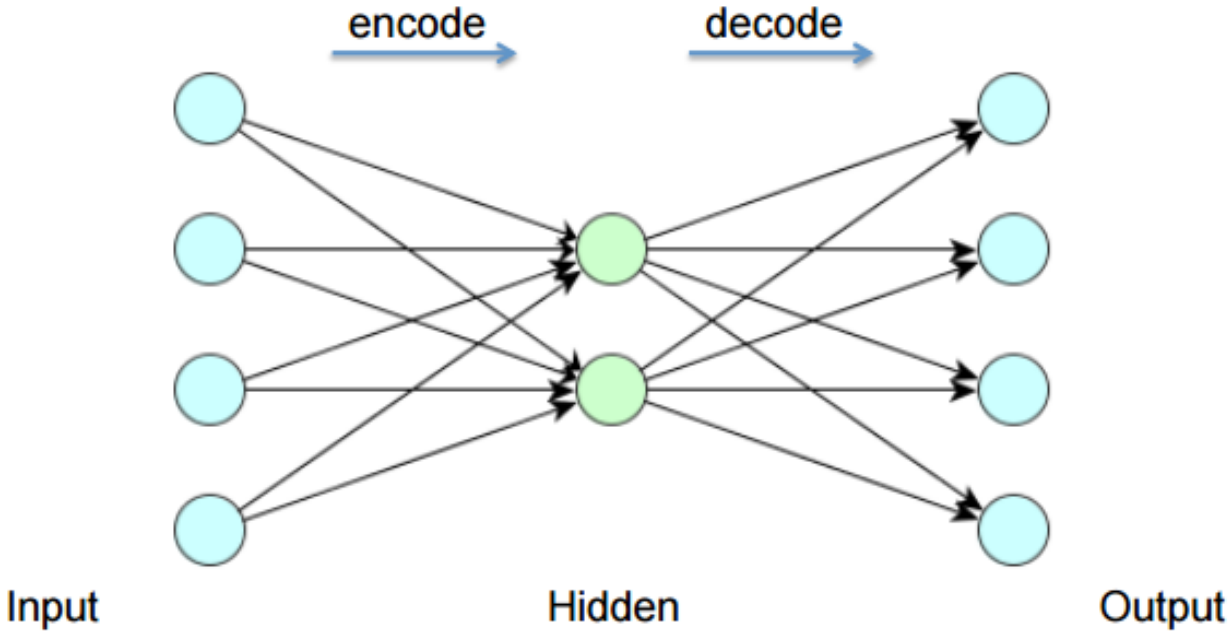
$$h'_t = \tanh W \begin{pmatrix} x_t \\ r_t \otimes h_{t-1} \end{pmatrix}$$

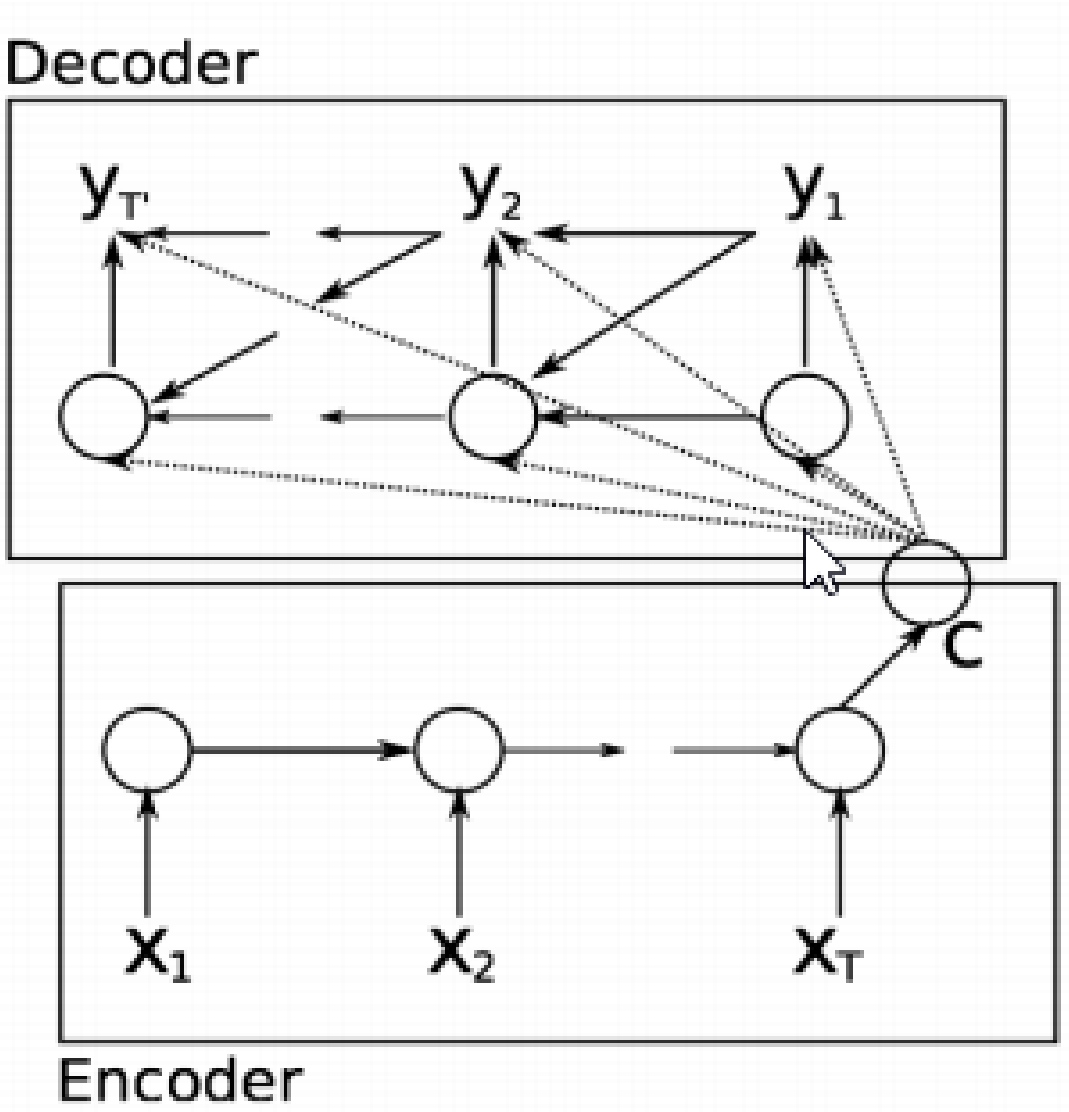
$$z_t = \sigma\left( W_z \begin{pmatrix} x_t \\ h_{t-1} \end{pmatrix} + b_f \right)$$

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

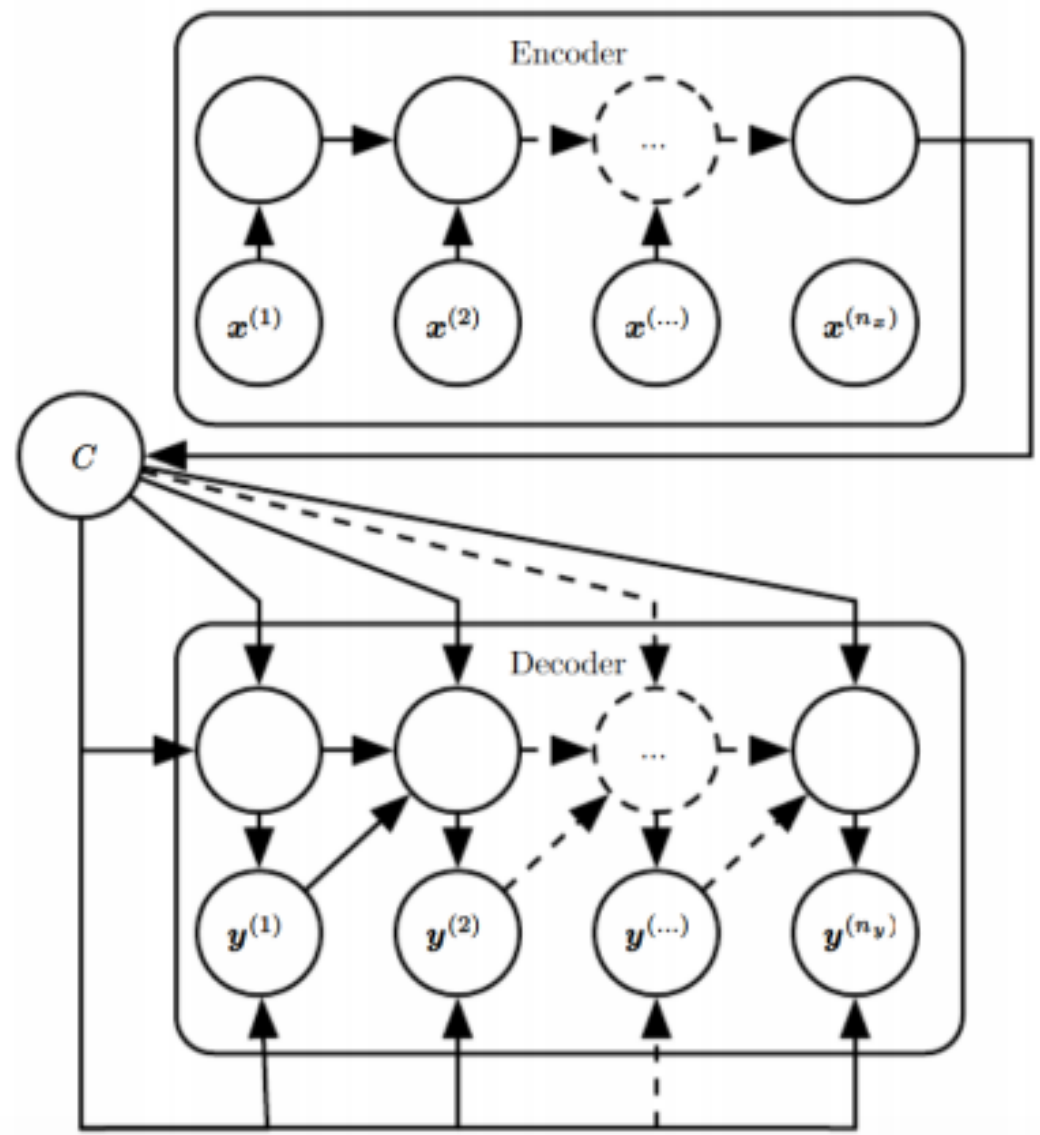
# The Encoder-Decoder Framework

- Auto-association (auto-encoding)
  - Learn a compressed representation of the input (think of word2vec, except simpler)
  - Bottleneck layer = meaningful latent space
- Can de-couple encoder & decoder
  - Each can be complex, different functions (like RNNs)



# An Encoder-Decoder or Sequence-to-Sequence RNN




Learns to generate an output sequence

$(y^{(1)},..,y^{(n_y)})$

given an input sequence

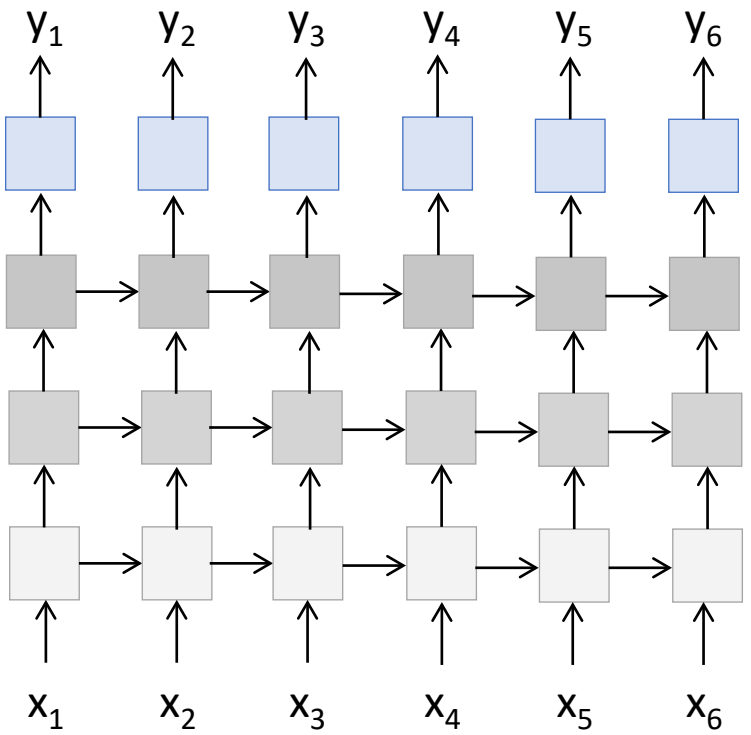
$(x^{(1)},..,x^{(n_x)})$

It consists of an encoder RNN that reads an input sequence and a decoder RNN that generates the output sequence or computes the probability of a given output sequence)

The final hidden state of the encoder RNN Is used to compute a fixed size context $C$ which represents a semantic summary of the input sequence and is given as input to the decoder

# Multi-layer RNNs

- We can design RNNs with multiple hidden layers:




- Might need to consider exotic modifications: Skip connections across layers, across time, …

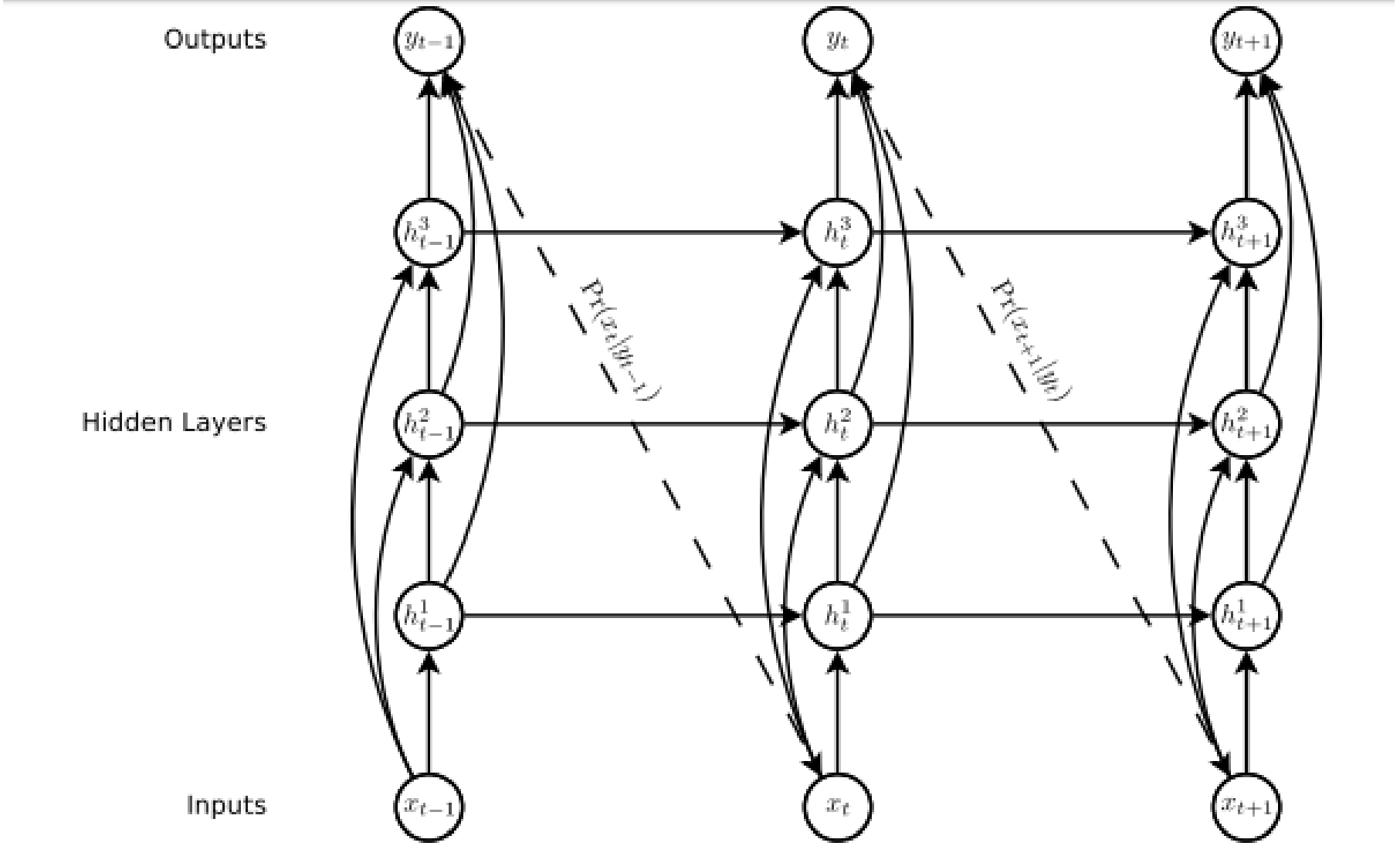
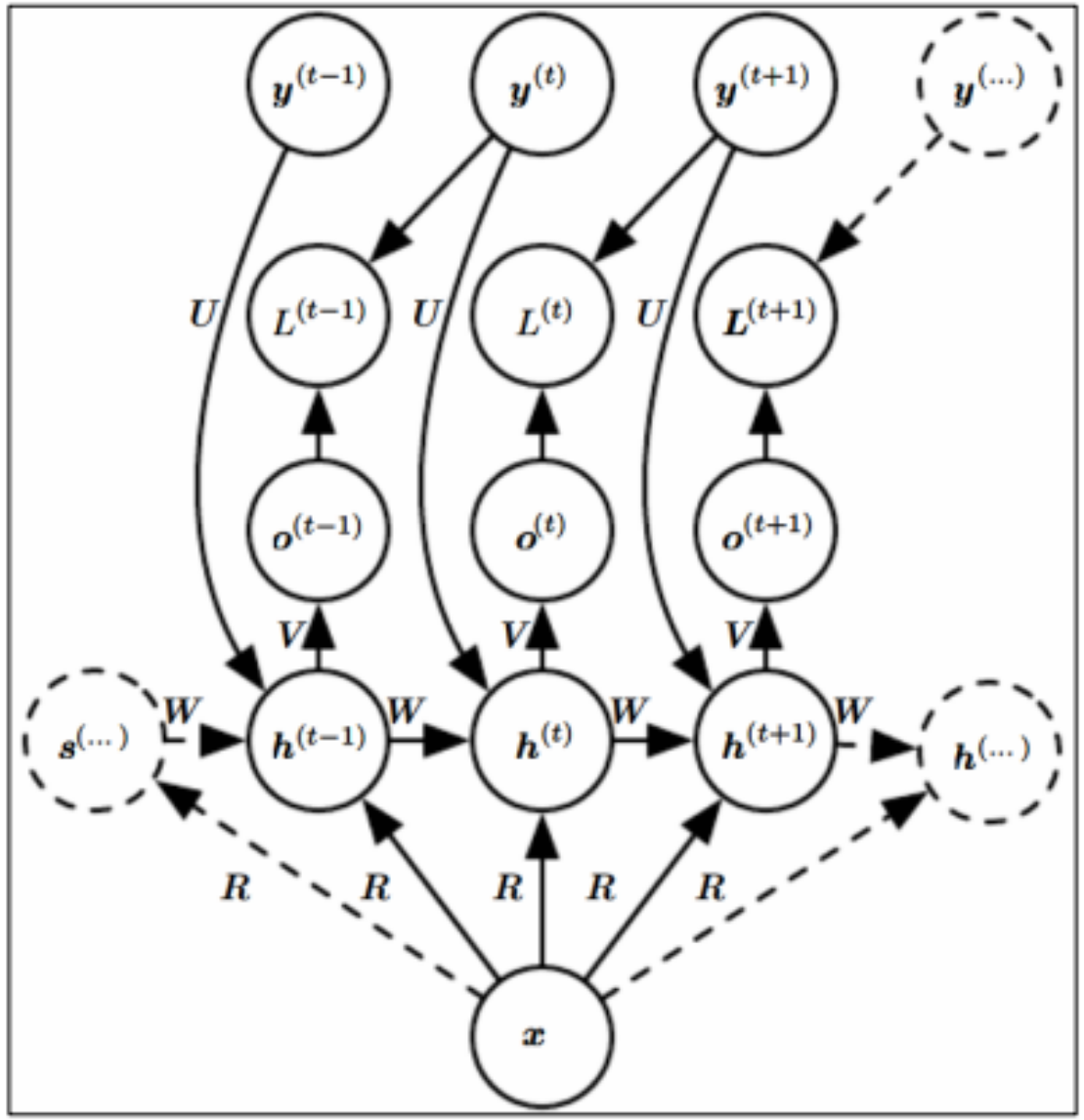
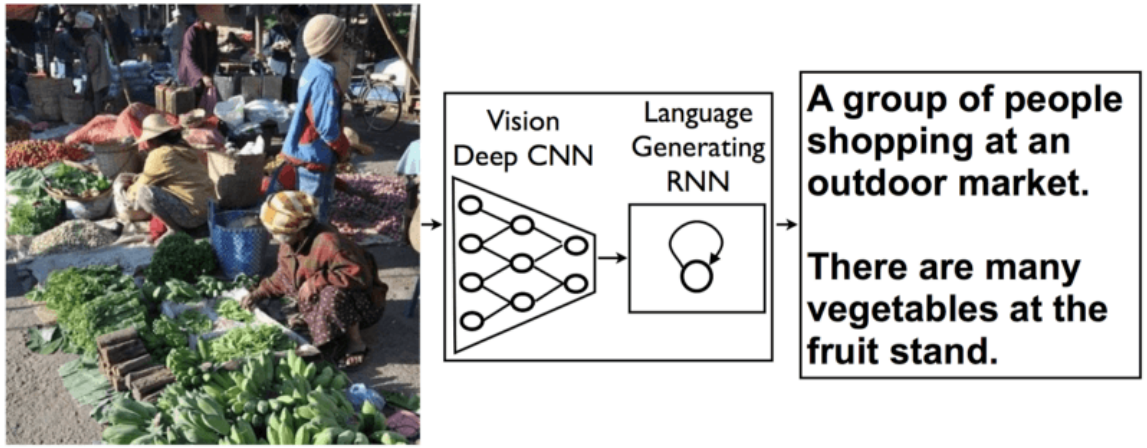



Figure 1: **Deep recurrent neural network prediction architecture.** The circles represent network layers, the solid lines represent weighted connections and the dashed lines represent predictions.

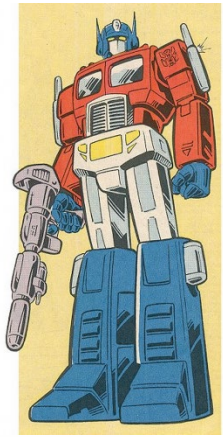
# RNN to map a fixed length vector $\boldsymbol{x}$ over sequences $\mathbf{Y}$



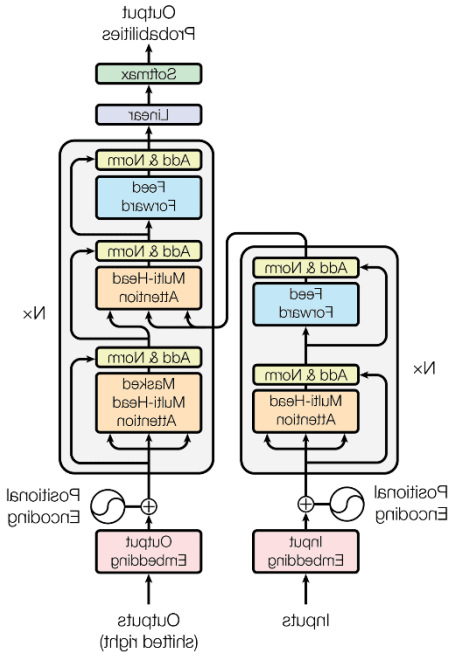

Appropriate for tasks such as image captioning
where a single image is input
which produces a sequence of words describing the image.
Each element of the observed output $y^{(t)}$ of the observed output sequence serves both as input (for the current time step) and during training as target



# Neural Transformers



- Beyond the RNN?
- Source https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html




**Attention Is All You Need**


**Ashish Vaswani*** Google Brain avaswani@google.com

**Noam Shazeer*** Google Brain noam@google.com

**Niki Parmar*** Google Research nikip@google.com

**Jakob Uszkoreit*** Google Research usz@google.com

**Llion Jones*** Google Research llion@google.com

**Aidan N. Gomez*** † University of Toronto aidan@cs.toronto.edu

**Łukasz Kaiser*** Google Brain lukaszkaiser@google.com

**Illia Polosukhin*** ‡ illia.polosukhin@gmail.com


*FNNs in disguise!*

# QUESTIONS?