CS60010: Deep Learning

Recurrent Neural Network

Sudeshna Sarkar

Spring 2018

8 Feb 2018

RNN 3: hidden2hidden, single output.

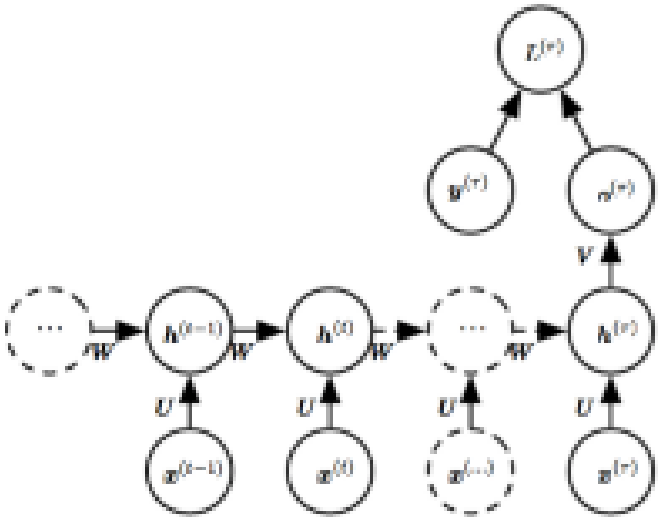


Figure 10.5: Time-unfolded recurrent neural network with a single output at the end of the sequence. Such a network can be used to summarize a sequence and produce a fixed-size representation used as input for further processing. There might be a target

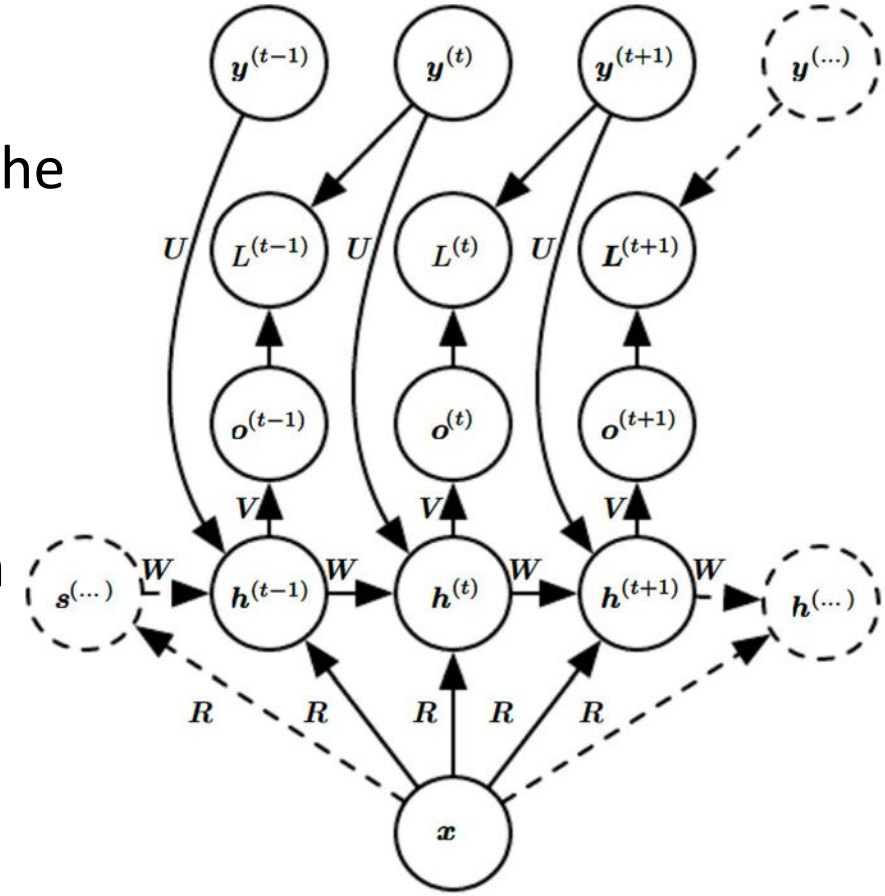
Such a network can be used to summarize a sequence and produce a fixed-size representation used as input for further processing.
There might be a target right at the end or the gradient on the output $o^{(t)}$ can be obtained by backpropagation from further downstream modules

Vector to sequence RNN

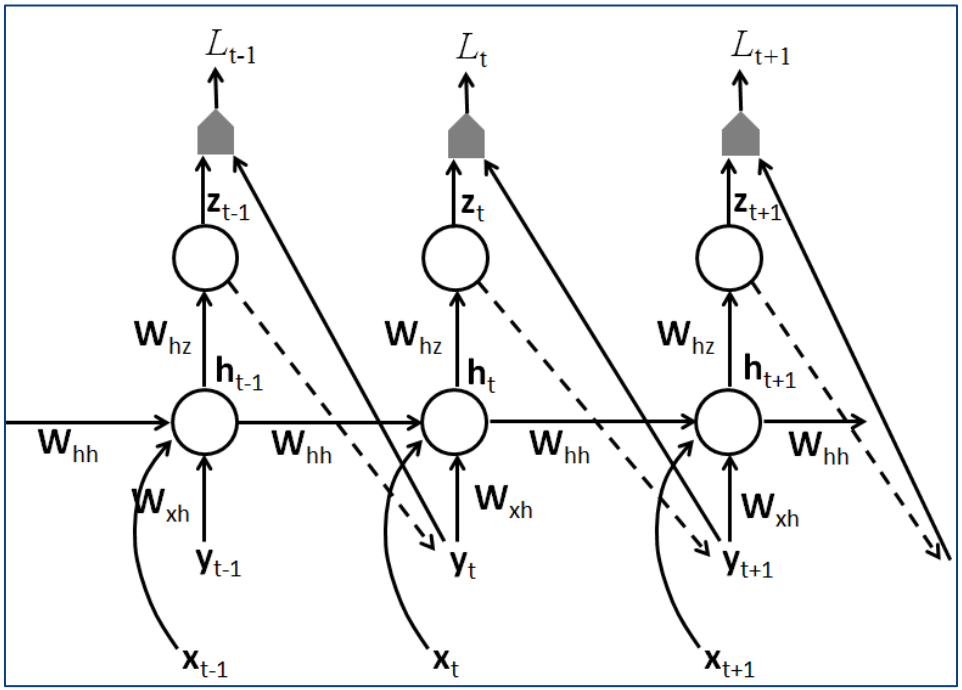
If **x** is a fixed-sized vector, we can make it an extra input of the RNN that generates the **y** sequence. Some common ways of providing the extra input

- as an extra input at each time step,
- as the initial state $\mathbf{h}_0$
- both

- Example: generate caption for an image



- The input **x** is a sequence of the same length as the output sequence **y**



- Removing the dash lines, it assumes $\mathbf{y}_t$'s are independent of each other when the past input sequence is given, i.e.
  $P(\mathbf{y}_t \mid \mathbf{y}_{t-1}, \ldots, \mathbf{y}_1, \mathbf{x}_t, \ldots, \mathbf{x}_1) = P(\mathbf{y}_t \mid \mathbf{x}_t, \ldots, \mathbf{x}_1)$
- Without the conditional independence assumption, add the dash lines and the prediction of $\mathbf{y}_{t+1}$ is based on both the past **x**'s and past **y**'s

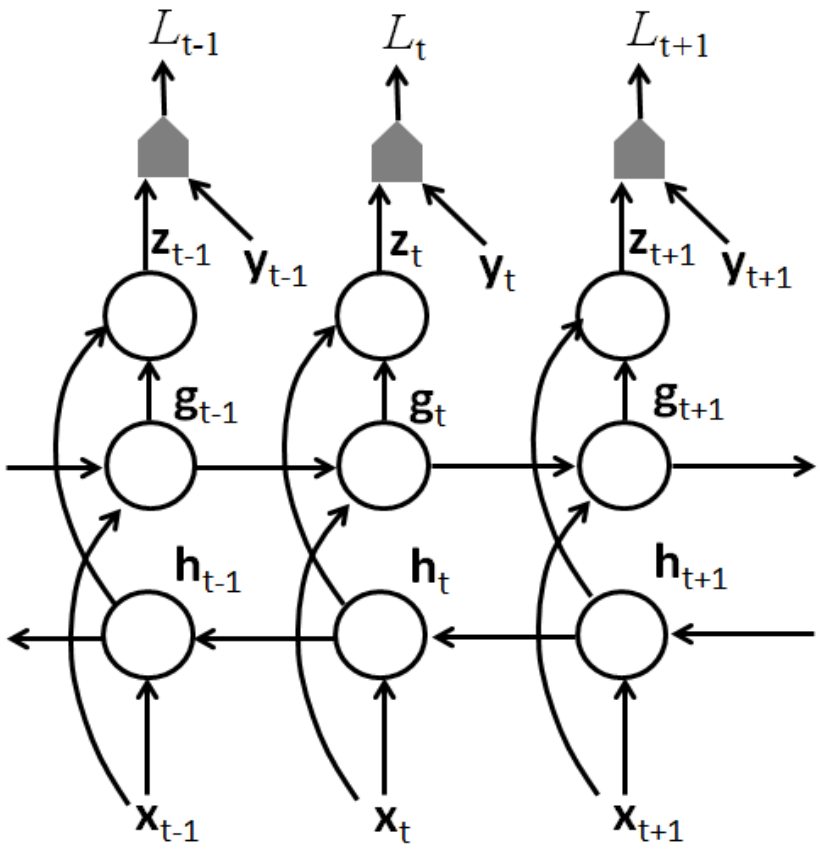
RNNs share same weights across Time Steps

- To go from multi-layer networks to RNNs:
  - Need to share parameters across different parts of a model
  - Separate parameters for each value of cannot generalize to sequence lengths not seen during training
  - Share statistical strength across different sequence lengths and across different positions in time
- Sharing important when information can occur at multiple positions in the sequence
  - Given "*I went to Nepal in* 1999 " and "*In* 1999*, I went to Nepal* ", an ML method to extract year, should extract 1999 whether in position 6 or 2
  - A feed-forward network that processes sentences of fixed length would have to learn all of the rules of language separately at each position
  - An RNN shares the same weights across several time steps

Bidirectional RNNs

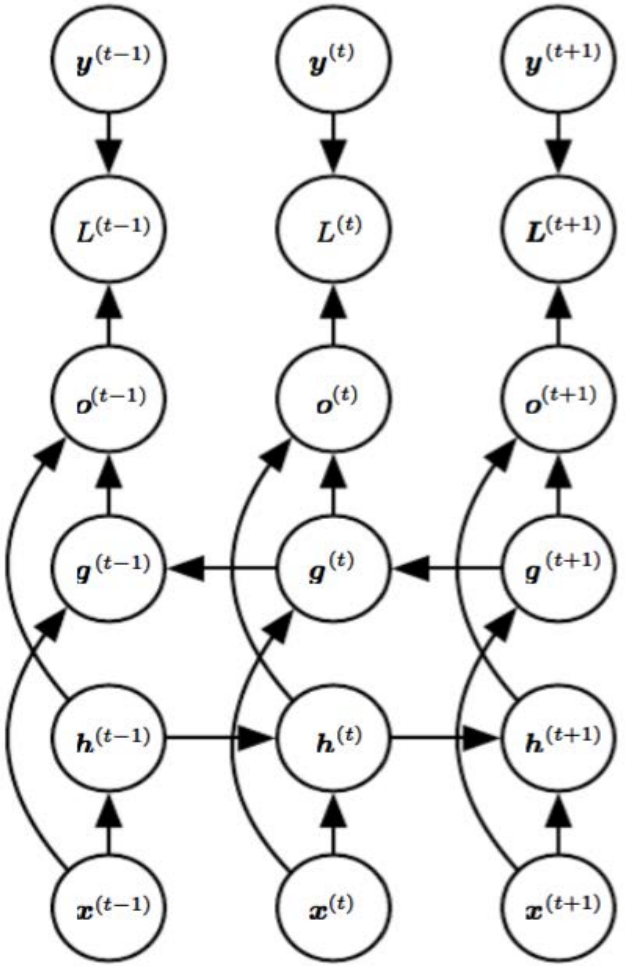
- In some applications, we want to output at time *t* a prediction regarding an output which may depend on the whole input sequence, e.g., speech recognition, MT
- **Bidirectional** recurrent neural network combines a forward-going RNN and a backward-going RNN

The idea can be extended to 2D input with four RNN going in four directions



Bidirectional RNNs

- $h^{(t)}$ summaries the information from the past sequence, and
- $g^{(t)}$ summaries the information from the future sequence



Encoder-Decoder Sequence to Sequence RNN

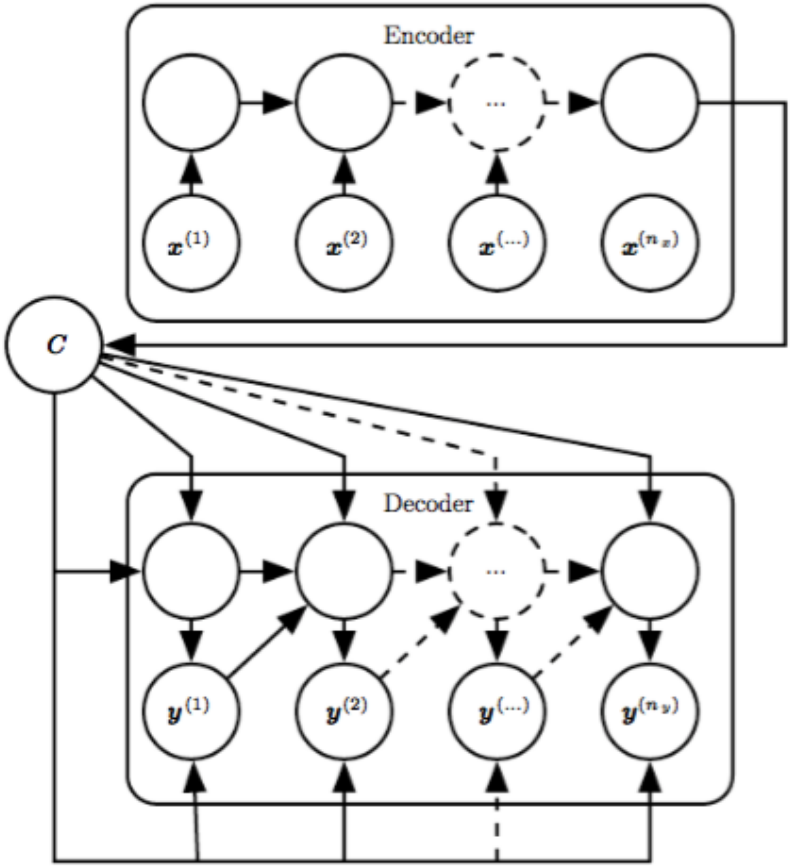
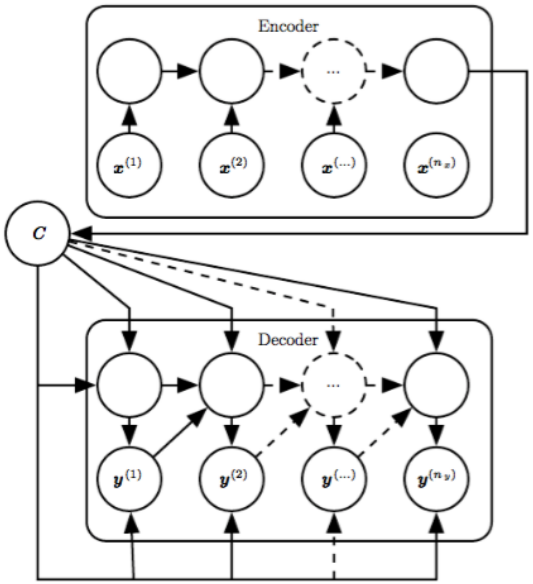


Figure 10.12: Example of an encoder-decoder or sequence-to-sequence RNN architecture, for learning to generate an output sequence $(\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(n_y)})$ given an input sequence $(\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(n_x)})$. It is composed of an encoder RNN that reads the 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 generally fixed-size context variable C which represents a semantic summary of the input sequence and is given as input to the decoder RNN.

Encoder-Decoder Sequence to Sequence RNN



- An encoder or reader or input RNN processes the input sequence. The encoder emits the context C , usually as a simple function of its final hidden state.
- A decoder or writer or output RNN is conditioned on that fixed-length vector to generate the output sequence Y = (y(1) , . . . , y(ny)).
- **Training**: two RNNs are trained jointly to maximize the average of logP(y(1),…,y(ny) |x(1),…,x(nx)) over all the pairs of x and y sequences in the training set.

Deep Recurrent Networks

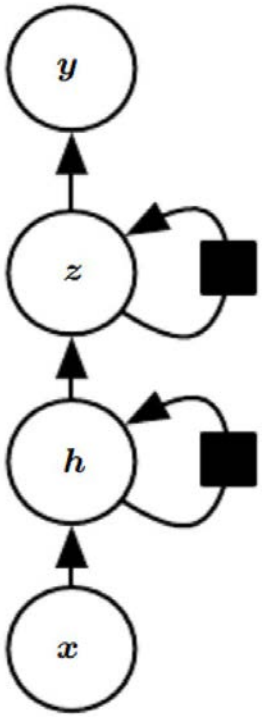
- The computation in most RNNs can be decomposed into three blocks of parameters and associated transformations
  1. From the input to the hidden state
  2. From the previous hidden state to the next hidden state
  3. From the hidden state to the output
- introduce depth in each of these operations?

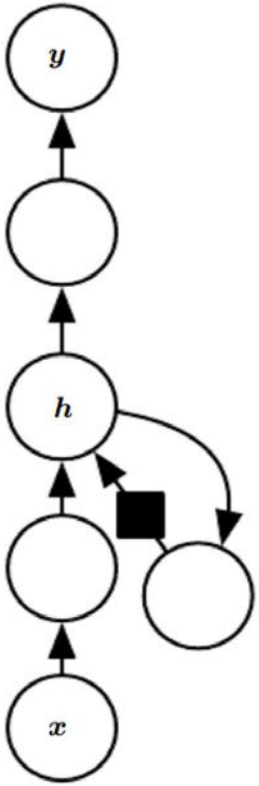
Ways of making an RNN deep

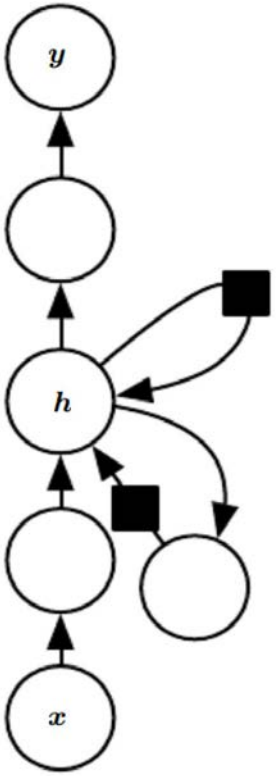
1. Hidden recurrent state can be broken down into groups organized hierarchically

2. Deeper computation can be introduced in the input-hidden, hidden-hidden and hidden-output parts. This may lengthen the shortest path linking different time steps

3. The path-lengthening effect can be mitigated by introducing skip connections.

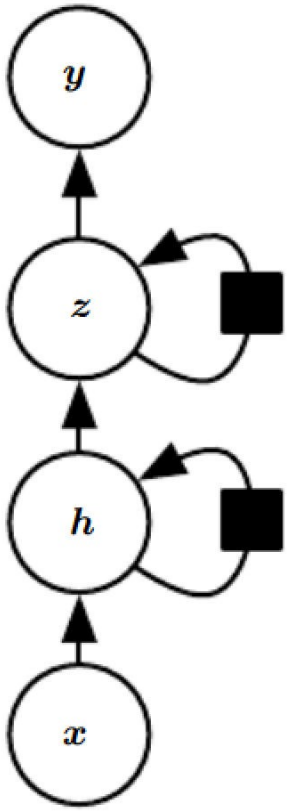






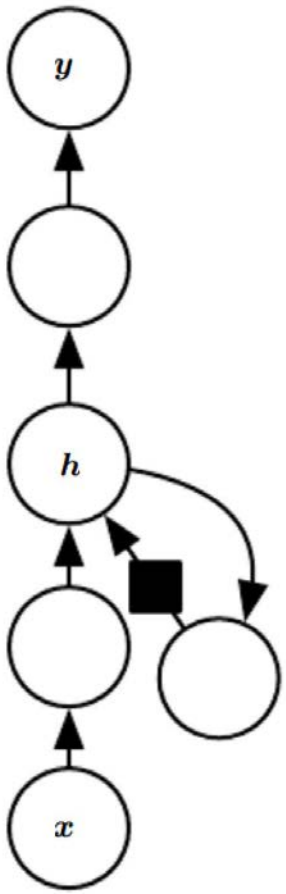
Recurrent states broken down into groups

- We can think of lower levels of the hierarchy play a role of transforming the raw input into a representation that is more appropriate at the higher levels of the hidden state



Deeper computation in hidden-to-hidden

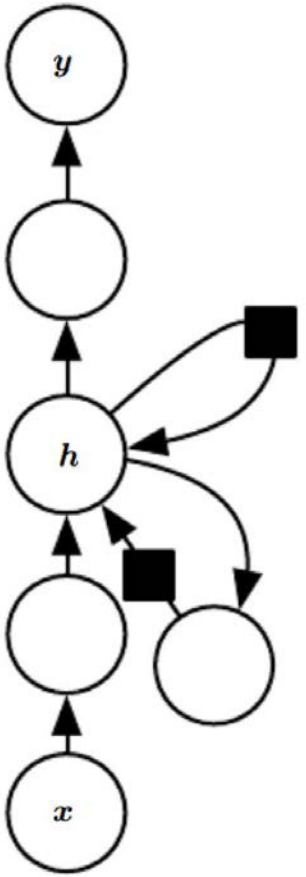
- Have a separate MLP (possibly deep) for each of the three blocks:
  - From input to hidden
  - From hidden to hidden
  - From hidden to output
- By adding depth may hurt learning by making optimization difficult
- In general it is easier to optimize shallower architectures
- Adding the extra depth makes the shortest time of a variable from time step t to a variable in time step t+1 become longer



Introducing skip connections

- For example, if an MLP with a single hidden layer is used for the state-to- state transition, we have doubled the length of the shortest path between variables in any two different time steps compared with the ordinary RNN.

- This can be mitigated by introducing skip connections in the hidden-to-hidden path.



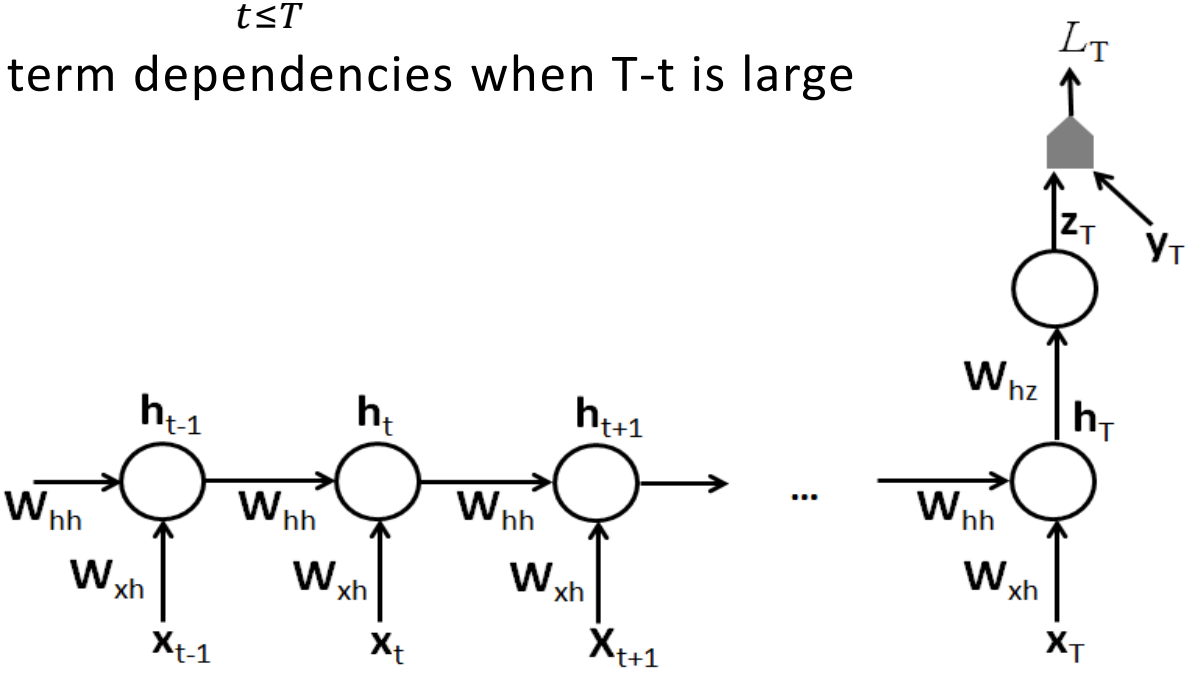
Problem of Long-Term Dependencies

- Consider the gradient of a loss L_T at time T with respect to the parameter θ of the recurrent function f_θ

$$h^{(t)} = f_\theta\left(h^{(t-1)}, x^{(t)}\right)$$

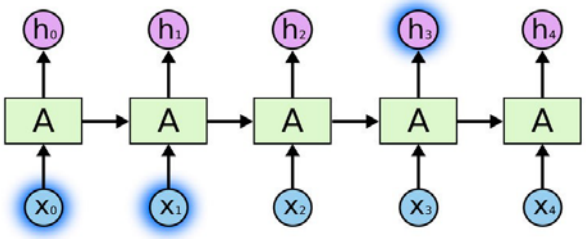
$$\frac{\partial L_T}{\partial \theta} = \sum_{t \le T} \frac{\partial L_T}{\partial h^{(t)}} \frac{\partial h^{(T)}}{\partial h^{(t)}} \frac{\partial f_\theta(h^{(t-1)}, x^{(t)})}{\partial \theta}$$

Encodes long term dependencies when T-t is large



Problem of Long-Term Dependencies

- Easy to predict last word in “the clouds are in the *sky*,”
  - When gap between relevant information and place that it’s needed is small, RNNs can learn to use the past information



- “I grew up in France… I speak fluent *French*.”
  - We need the context of France, from further back.
  - Large gap between relevant information and point where it is needed