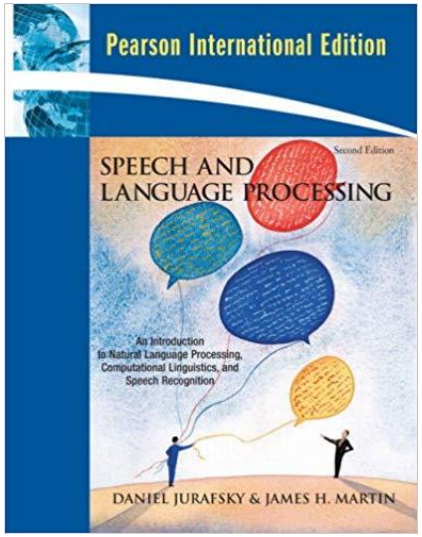
NLP & IR



Chapter 9

Deep Learning Architectures for Sequence Processing

Dell Zhang
Birkbeck, University of London

Sliding Window

- The sliding window approach that we use in feed-forward network based neural language models is problematic.
  - First, it shares the primary weakness of Markov approaches in that it limits the context from which information can be extracted; anything outside the context window has no impact on the decision being made.
  - Second, the use of windows makes it difficult for networks to learn systematic patterns arising from phenomena like constituency.

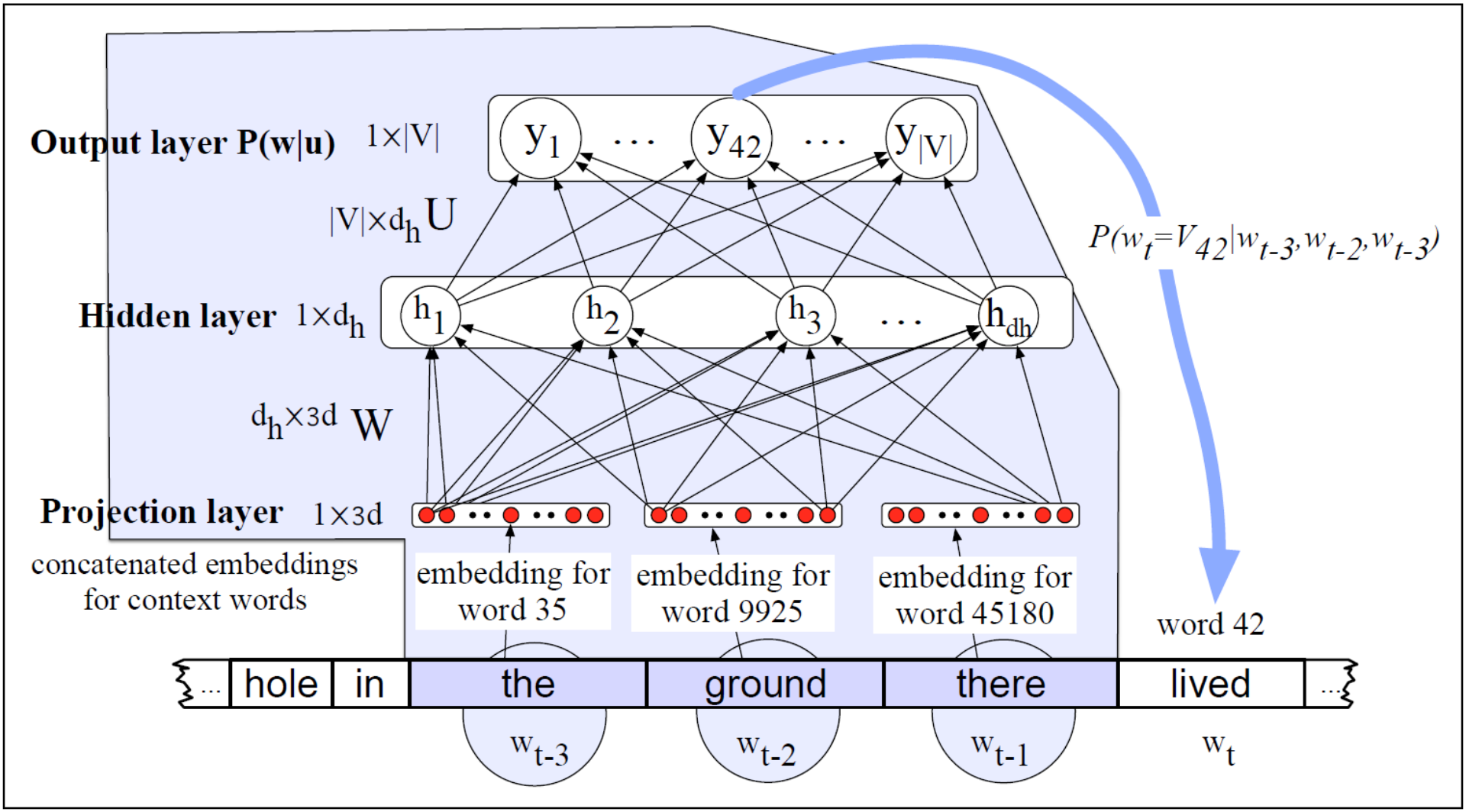


Figure 9.1 A simplified view of a feedforward neural language model moving through a text. At each timestep t the network takes the 3 context words, converts each to a d-dimensional embeddings, and concatenates the 3 embeddings together to get the $1 \times Nd$ unit input layer x for the network.

For example, the phrase "the ground" appears twice in different windows: once, as shown, in the first and second positions in the window, and in the preceding step in the second and third slots, thus forcing the network to learn two separate patterns for a single constituent.

Recurrent Neural Networks

- A recurrent neural network (RNN) is any network that contains a cycle within its network connections, i.e., any network where the value of a unit is directly, or indirectly, dependent on its own output as an input.
  - RNNs allow us to handle *variable length* input sequences explicitly as sequences without the use of arbitrary fixed-sized windows.

Simple Recurrent Networks

aka Elman Networks

The key difference from a feedforward network lies in *the recurrent link* (shown as a dashed line).

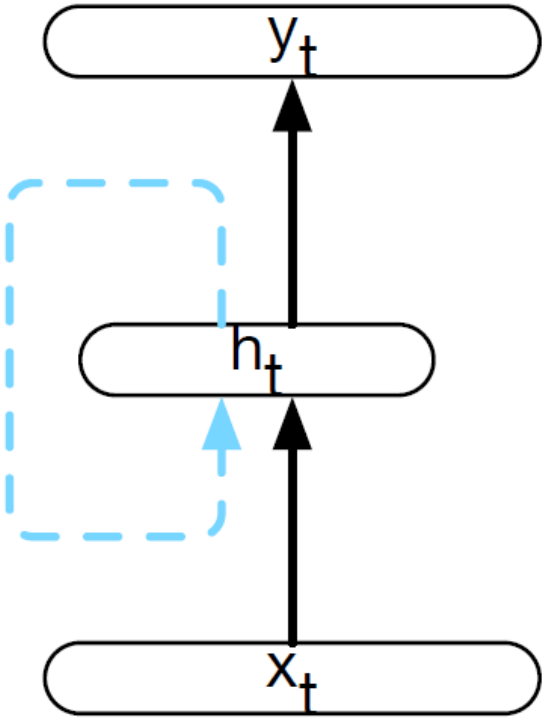


Figure 9.2 Simple recurrent neural network after Elman (Elman, 1990). The hidden layer includes a recurrent connection as part of its input. That is, the activation value of the hidden layer depends on the current input as well as the activation value of the hidden layer from the previous timestep.

Simple Recurrent Networks

- The recurrent link augments the input to the hidden layer with the activation value of the hidden layer *from the preceding point in time*.
  - The hidden layer from the previous timestep provides *a form of memory, or context*, that encodes earlier processing and informs the decisions to be made at later points in time.
  - Importantly, the architecture does not impose a fixed-length limit on this prior context; the context embodied in the previous hidden layer includes *information extending back to the beginning* of the sequence.

Simple Recurrent Networks

The most significant addition lies in the new set of weights, U, that connect the hidden layer from the previous timestep to the current hidden layer.

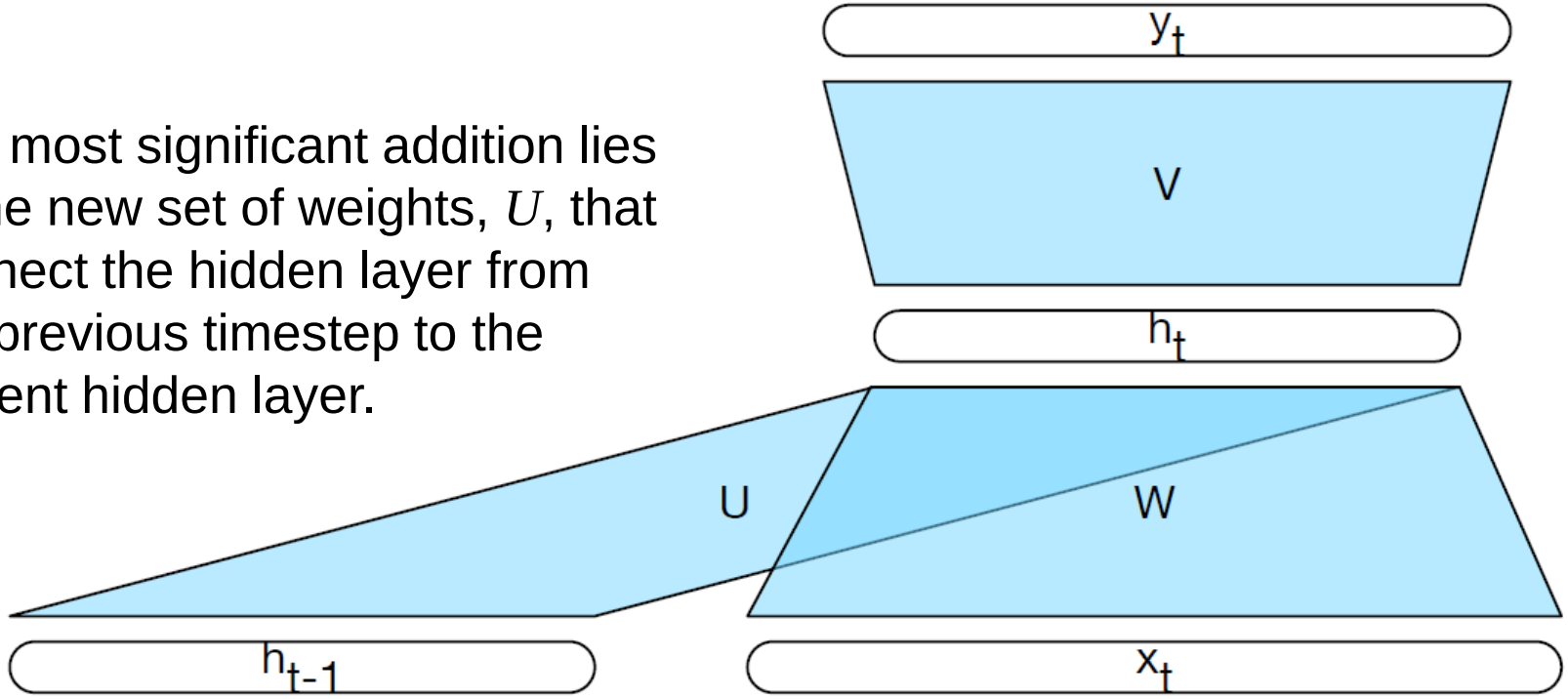


Figure 9.3 Simple recurrent neural network illustrated as a feed-forward network.

Simple Recurrent Networks

- Forward inference (mapping a sequence of inputs to a sequence of outputs) in an SRN is nearly identical to what we've already seen with feed-forward networks.

$$h_t = g(Uh_{t-1} + Wx_t)$$
$$y_t = f(Vh_t) \qquad \text{e.g.,} \quad y_t = softmax(Vh_t)$$

Simple Recurrent Networks

- The fact that the computation at time t requires the value of the hidden layer from time t-1 mandates an *incremental* inference algorithm that proceeds from the start of the sequence to the end.

function FORWARDRNN(x, *network*) **returns** output sequence y

$h_0 \leftarrow 0$
for $i \leftarrow 1$ **to** LENGTH(x) **do**
  $h_i \leftarrow g(U\ h_{i-1} + W\ x_i)$
  $y_i \leftarrow f(V\ h_i)$
return y

Figure 9.5 Forward inference in a simple recurrent network.

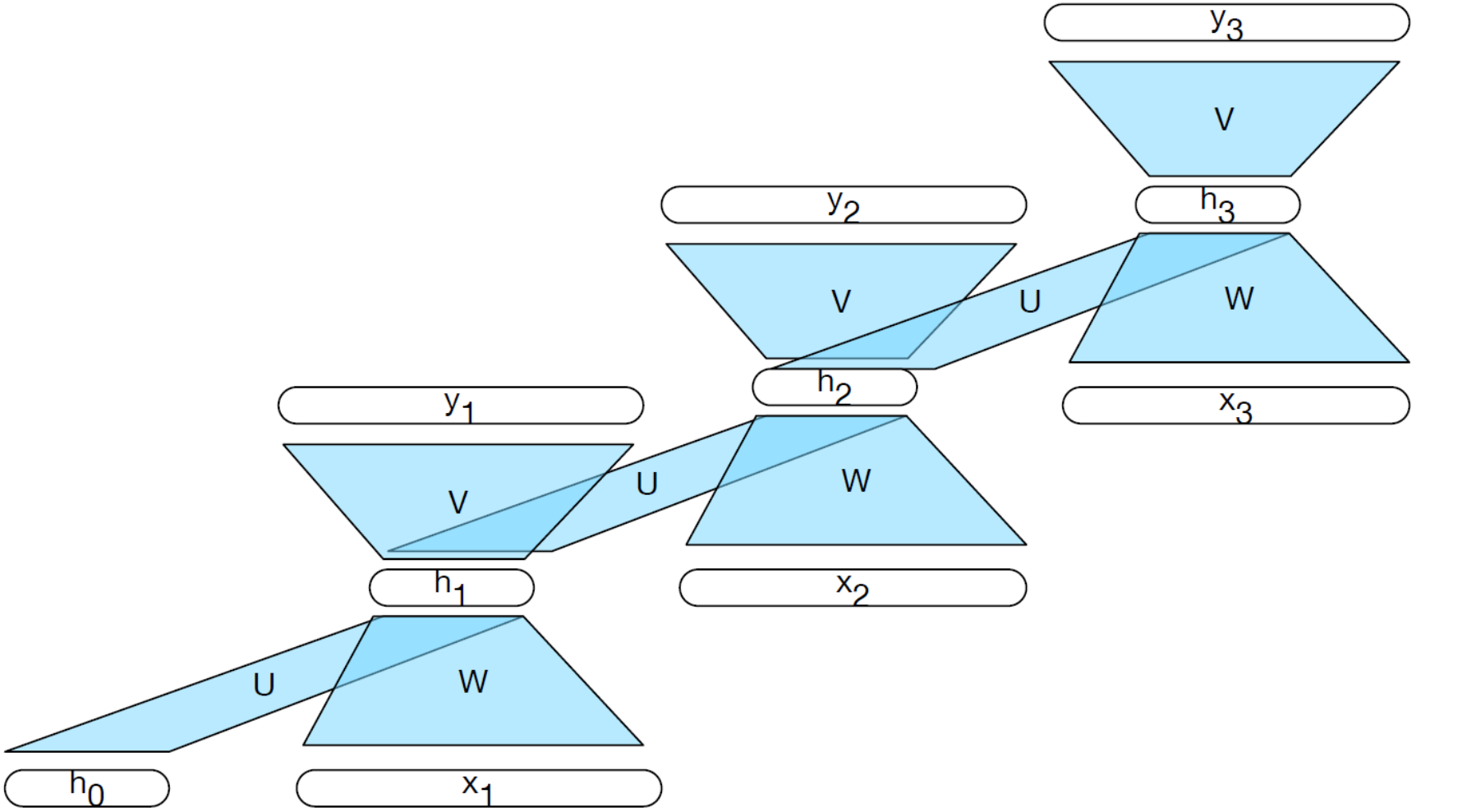


Figure 9.4 A simple recurrent neural network shown unrolled in time. Network layers are copied for each timestep, while the weights U, V and W are shared in common across all timesteps.

Applications of RNNs in NLP

- Sequence **Labelling**
  - Part-Of-Speech (POS) Tagging
  - Named Entity Recognition (NER)
- Sequence **Classification**
  - Sentiment Analysis
  - Topic Categorization
  - Spam Filtering
  - Message Routing
- Sequence **Generation**
  - *We'll come back to this later.*
- *……*

Part-Of-Speech (POS) Tagging

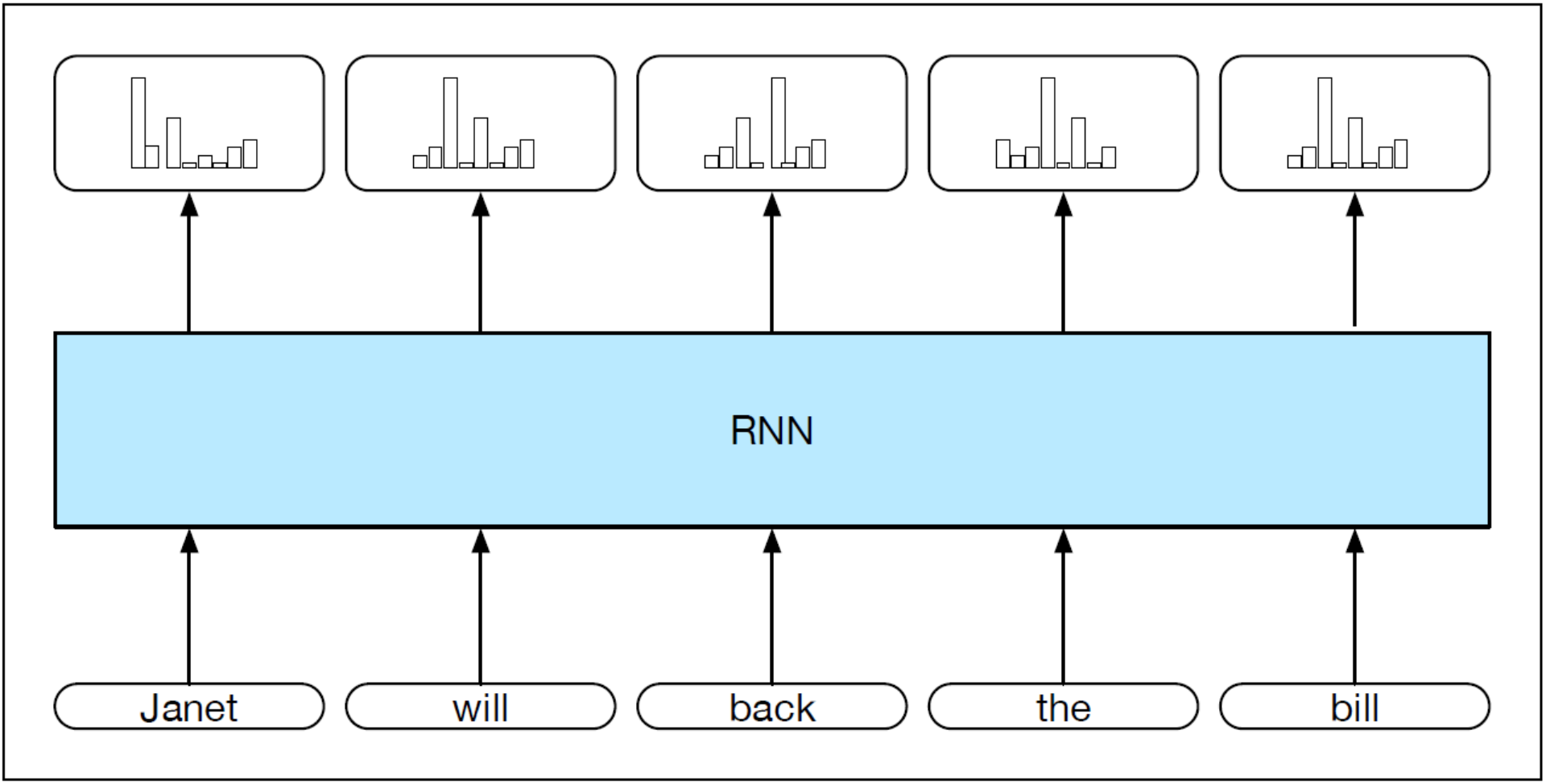


Figure 9.8 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

The Penn TreeBank Tagset

Tag	Description	Example	Tag	Description	Example
CC	coordin. conjunction	*and, but, or*	SYM	symbol	+,%, &
CD	cardinal number	*one, two*	TO	"to"	*to*
DT	determiner	*a, the*	UH	interjection	*ah, oops*
EX	existential 'there'	*there*	VB	verb base form	*eat*
FW	foreign word	*mea culpa*	VBD	verb past tense	*ate*
IN	preposition/sub-conj	*of, in, by*	VBG	verb gerund	*eating*
JJ	adjective	*yellow*	VBN	verb past participle	*eaten*
JJR	adj., comparative	*bigger*	VBP	verb non-3sg pres	*eat*
JJS	adj., superlative	*wildest*	VBZ	verb 3sg pres	*eats*
LS	list item marker	*1, 2, One*	WDT	wh-determiner	*which, that*
MD	modal	*can, should*	WP	wh-pronoun	*what, who*
NN	noun, sing. or mass	*llama*	WP$	possessive wh-	*whose*
NNS	noun, plural	*llamas*	WRB	wh-adverb	*how, where*
NNP	proper noun, sing.	*IBM*	$	dollar sign	*$*
NNPS	proper noun, plural	*Carolinas*	#	pound sign	*#*
PDT	predeterminer	*all, both*	"	left quote	' or "
POS	possessive ending	*'s*	"	right quote	' or "
PRP	personal pronoun	*I, you, he*	(	left parenthesis	[, (, {, <
PRP$	possessive pronoun	*your, one's*	)	right parenthesis	],), }, >
RB	adverb	*quickly, never*	,	comma	,
RBR	adverb, comparative	*faster*	.	sentence-final punc	. ! ?
RBS	adverb, superlative	*fastest*	:	mid-sentence punc	: ; ... – -
RP	particle	*up, off*			

Named Entity Recognition (NER)

- Finding all the spans in a text that correspond to names of people, places or organizations etc.
- To turn a problem like this into a per-word sequence labeling task, we'll use **IOB encoding**:
  - I: tokens *inside* a span
  - O: tokens *outside* of any span of interest
  - B: tokens that *begin* a span of interest

In applications where we are interested in more than one class of entity, we can specialize the B and I tags to represent each of the more specific classes.

United	*cancelled*	*the*	*flight*	*from*	*Denver*	*to*	*San*	*Francisco.*
B-ORG	O	O	O	O	B-LOC	O	B-LOC	I-LOC

Sequence Classification

- To apply RNNs in this setting, the hidden layer from the final state of the network is taken to constitute a compressed representation of the entire sequence.
- This compressed sequence representation can then in turn serve as the input to a feed-forward network trained to select the correct class.
- We use the output from the softmax layer from the *final* classifier along with a cross-entropy loss function to drive our network training.

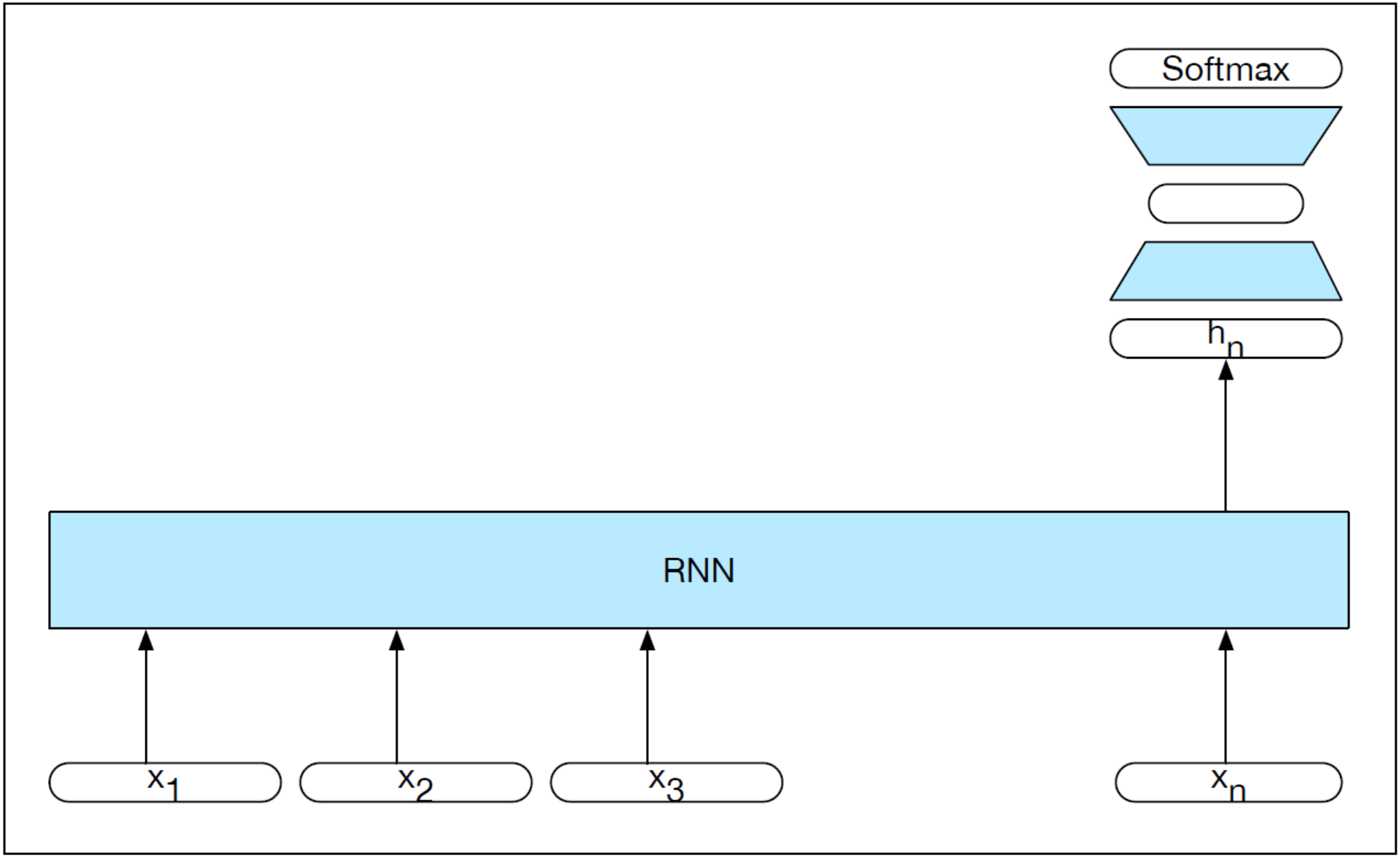


Figure 9.9 Sequence classification using a simple RNN combined with a feedforward network.

Deep Networks

- A **Stacked RNN** consists of multiple networks where the output of one layer serves as the input to a subsequent layer.
- It has been demonstrated across numerous tasks that stacked RNNs can outperform single-layer networks.
  - One reason for this success has to do with the networks ability to induce representations at differing levels of abstraction across layers.

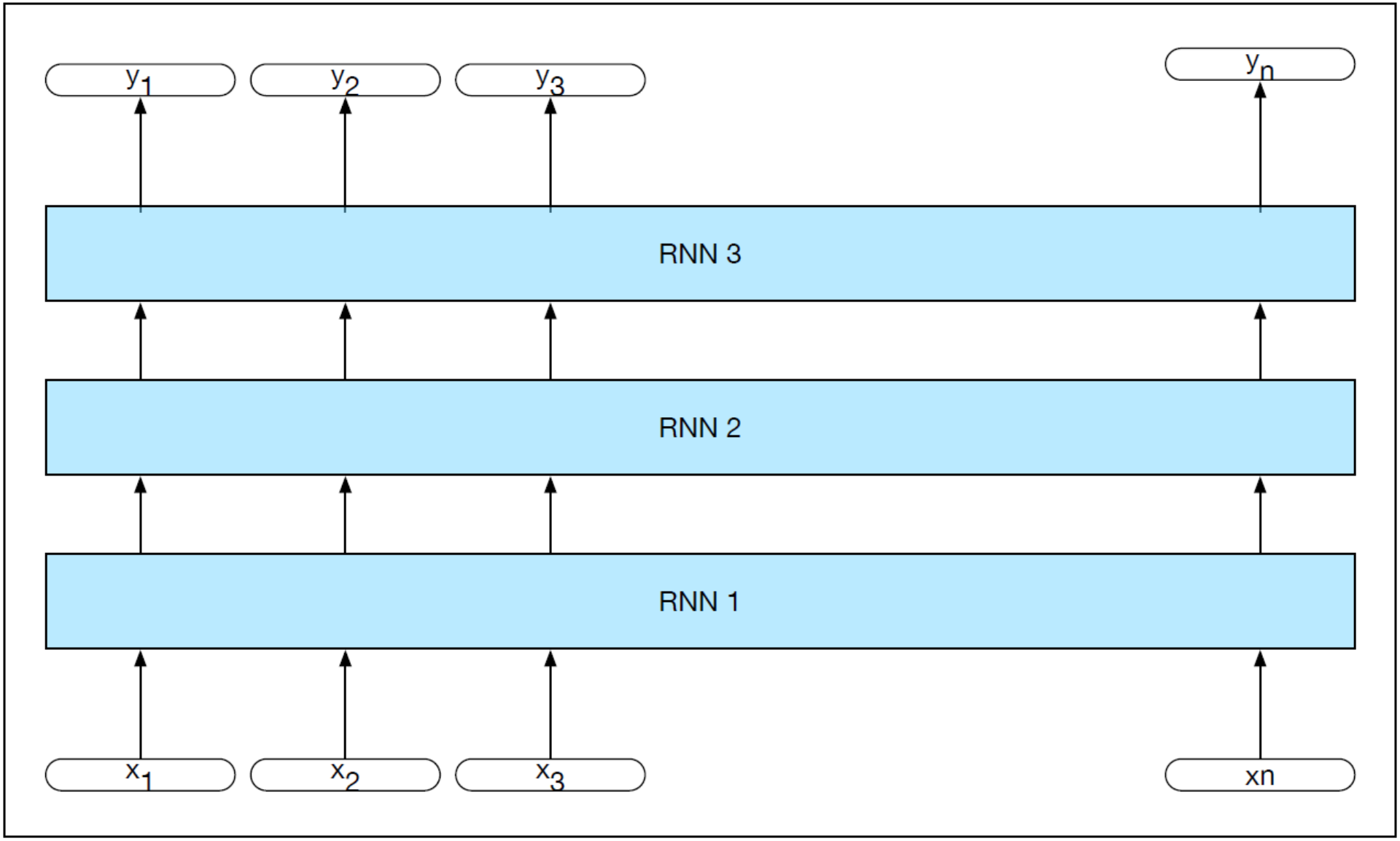


Figure 9.10 Stacked recurrent networks. The output of a lower level serves as the input to higher levels with the output of the last network serving as the final output.

Deep Networks

- A **Bidirectional RNN** consists of two independent recurrent networks, one where the input is processed *from the start to the end*, and the other *from the end to the start*.
- We can then combine the outputs of the two networks into a single representation that captures the both the left and right contexts of an input at each point in time.

$$h_t = h_t^{forward} \oplus h_t^{backward} \quad \begin{cases} h_t^{forward} = SRN_{forward}(x_1 : x_t) \\ h_t^{backward} = SRN_{backward}(x_n : x_t) \end{cases}$$

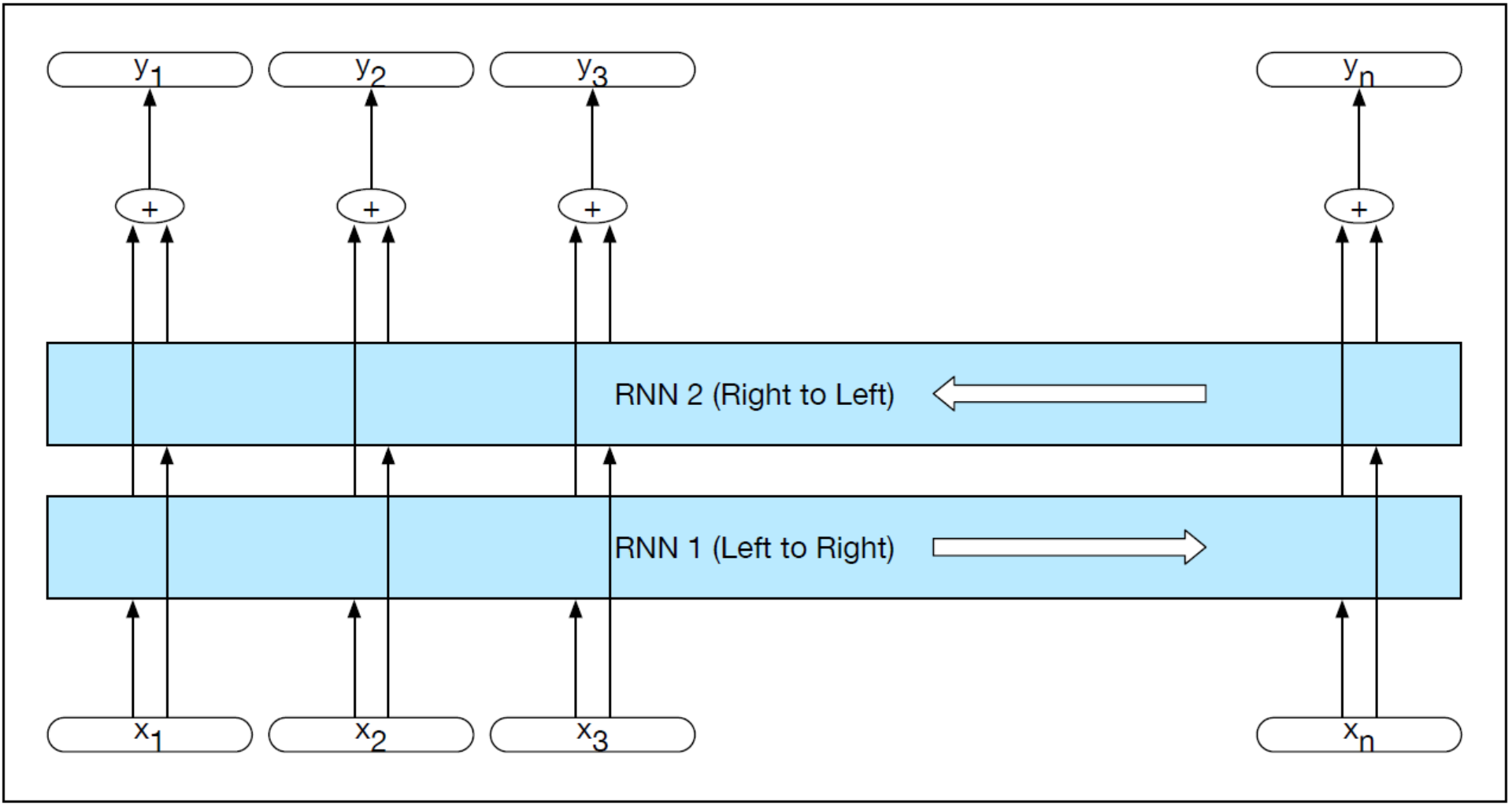


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions with the output of each model at each time point concatenated to represent the state of affairs at that point in time. The box wrapped around the forward and backward network emphasizes the modular nature of this architecture.

Deep Networks

- Bidirectional RNNs have also proven to be quite effective for sequence classification.
  - For sequence classification we used the final hidden state of the RNN as the input to a subsequent feed-forward classifier. A difficulty with this approach is that the final state naturally reflects more information about the end of the sentence than its beginning.
  - Bidirectional RNNs provide a simple solution to this problem: we simply combine the final hidden states from the forward and backward passes and use that as input for follow on processing.

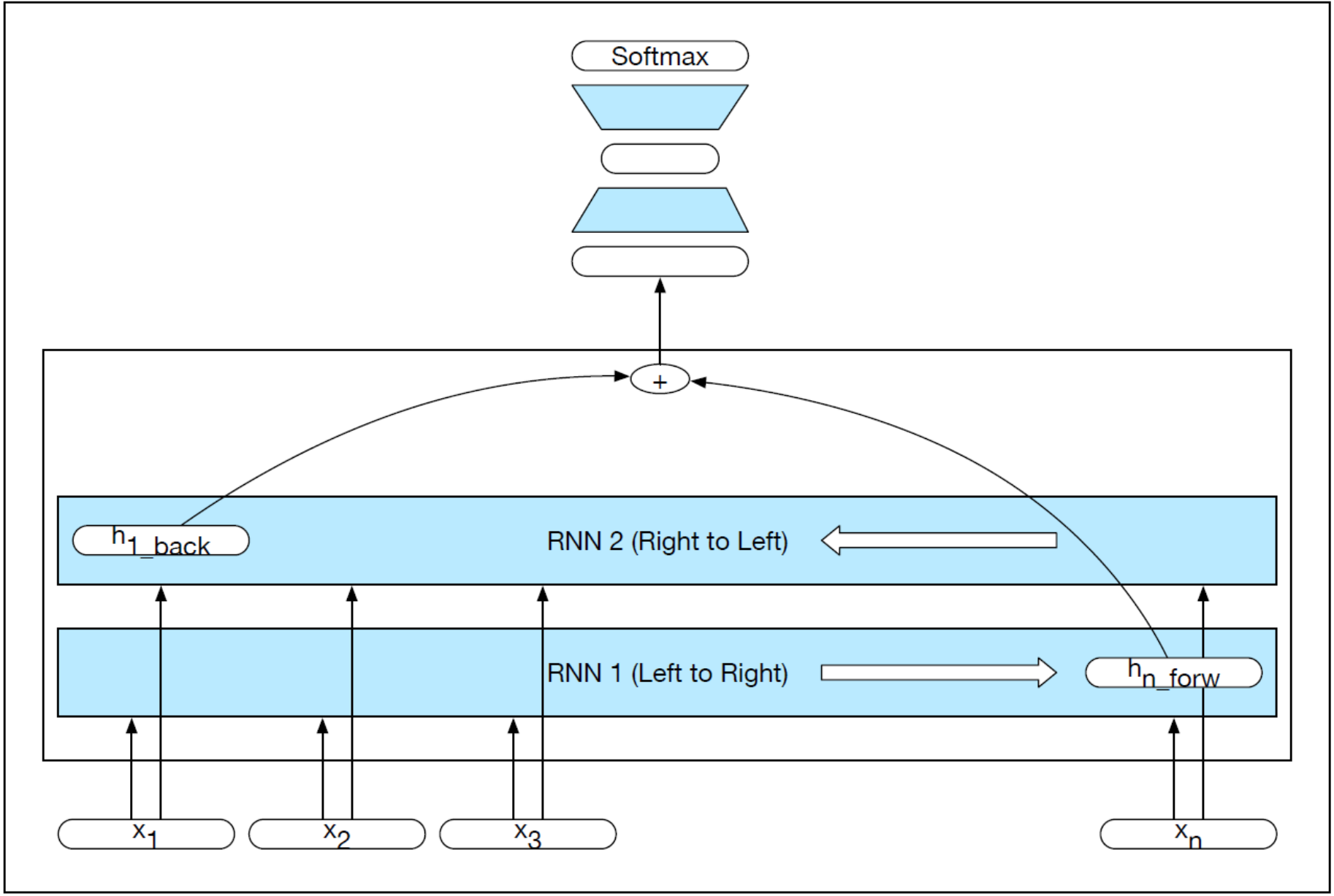


Figure 9.12 A bidirectional RNN for sequence classification. The final hidden units from the forward and backward passes are combined to represent the entire sequence. This combined representation serves as input to the subsequent classifier.

Managing Context in RNNs

- It is quite difficult to train simple RNNs for tasks that require a network to make use of information *distant* from the current point of processing.
  - Despite having access to the entire preceding sequence, the information encoded in hidden states tends to be fairly local, more relevant to the most recent parts of the input sequence and recent decisions.
  - However, it is often the case that *long-distance information* is critical to many language applications.

The flights the airline was cancelling ___ full.
(a) *was* (b) *were*

LSTM

- **Long Short-Term Memory (LSTM)** networks, divide the context management problem into two sub-problems: *removing* information no longer needed from the context, and *adding* information likely to be needed for later decision making.
- The key to the approach is to *learn* how to manage this context rather than hard-coding a strategy into the architecture.

LSTM

- LSTMs accomplish this through the use of specialized neural units that make use of ***gates*** to control the flow of information into and out of the units that comprise the network layers.
  - These gates are implemented through the use of additional sets of weights that operate sequentially on the context layer.

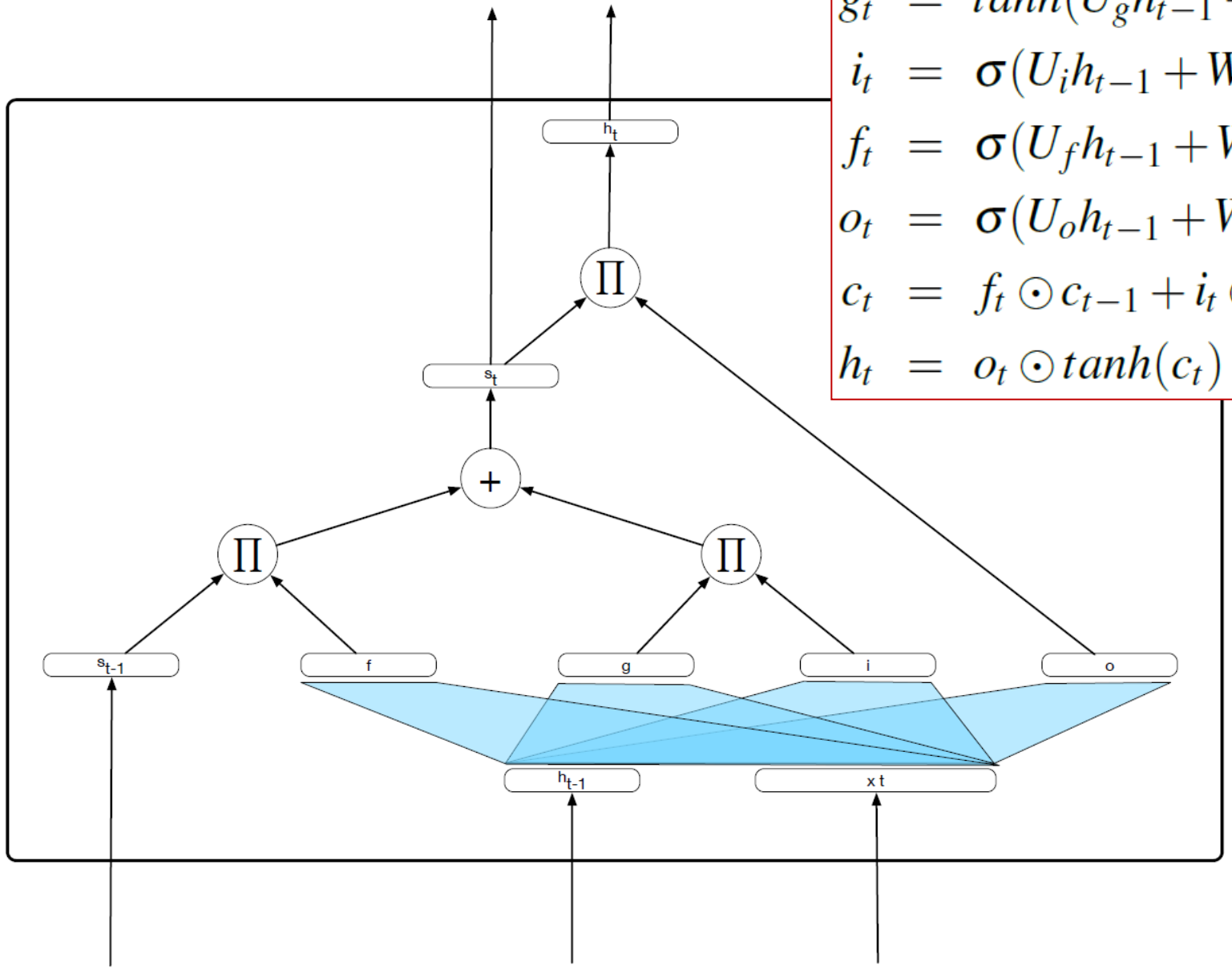


Figure 9.13 A single LSTM memory unit displayed as a computation graph.

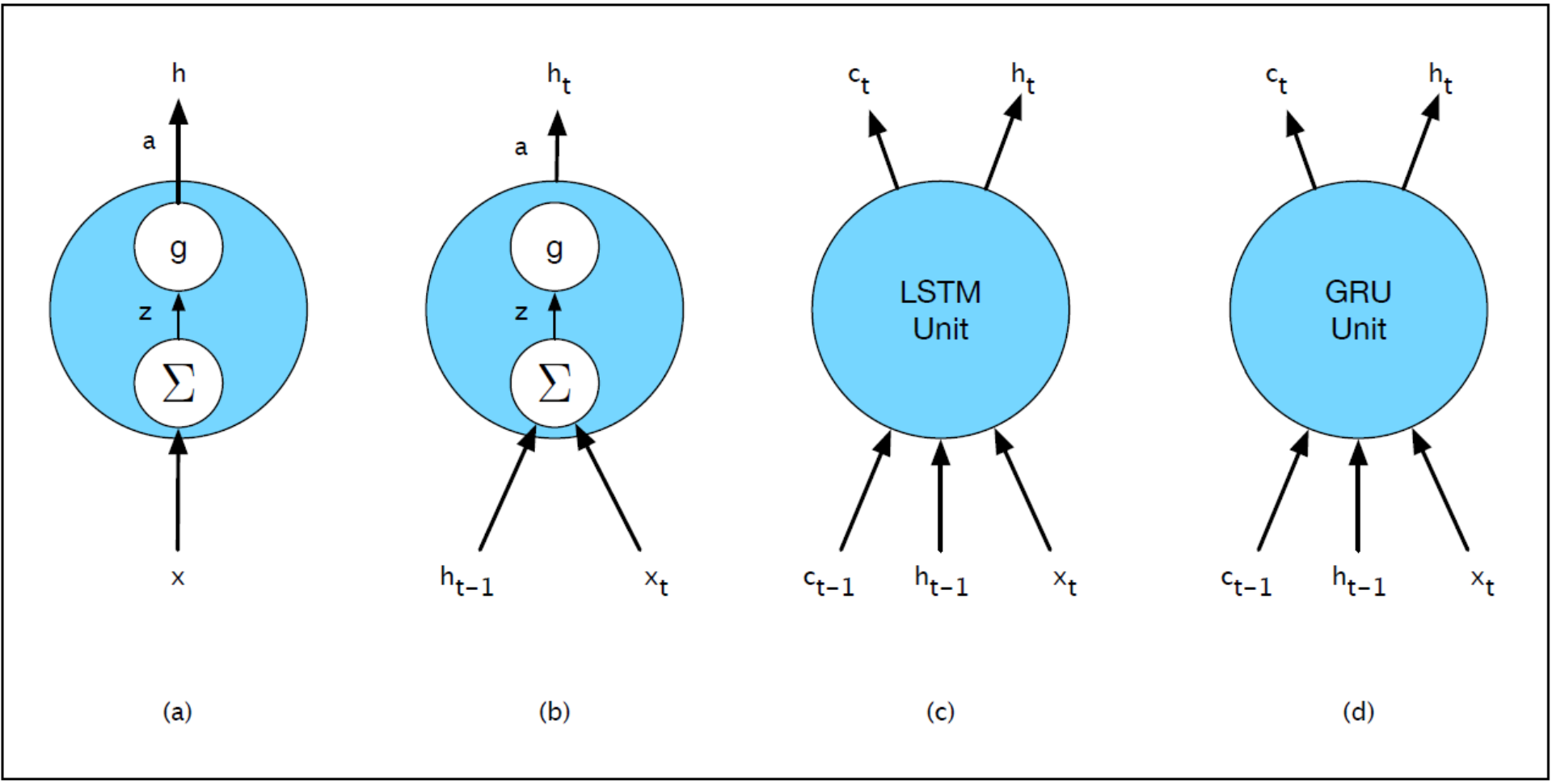


Figure 9.14 Basic neural units used in feed-forward, simple recurrent networks (SRN), long short-term memory (LSTM) and gate recurrent units.

LSTM

- The increased complexity of the LSTM and GRU units is encapsulated within the units themselves.
  - The only additional external complexity over the simple recurrent unit is the presence of the additional context vector input $c_{t\text{-}1}$ and output c_t.
  - This **modularity** is key to the power and widespread applicability of LSTM and GRU units.
  - Specifically, LSTM and GRU units can be substituted into any of the network architectures described before.

Andrej Karpathy blog

About Hacker's guide to Neural Networks

The Unreasonable Effectiveness of Recurrent Neural Networks

May 21, 2015

We'll train RNNs to generate text character by character and ponder the question "*how is that even possible?*"

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

Fun with RNNs

- Text **generation**
  using *character-level language models*
  based on *multi-layer LSTMs*.
  - Paul Graham
  - Shakespeare
  - Wikipedia
  - Algebraic Geometry (LaTeX)
  - Linux Source Code
  - Baby Names