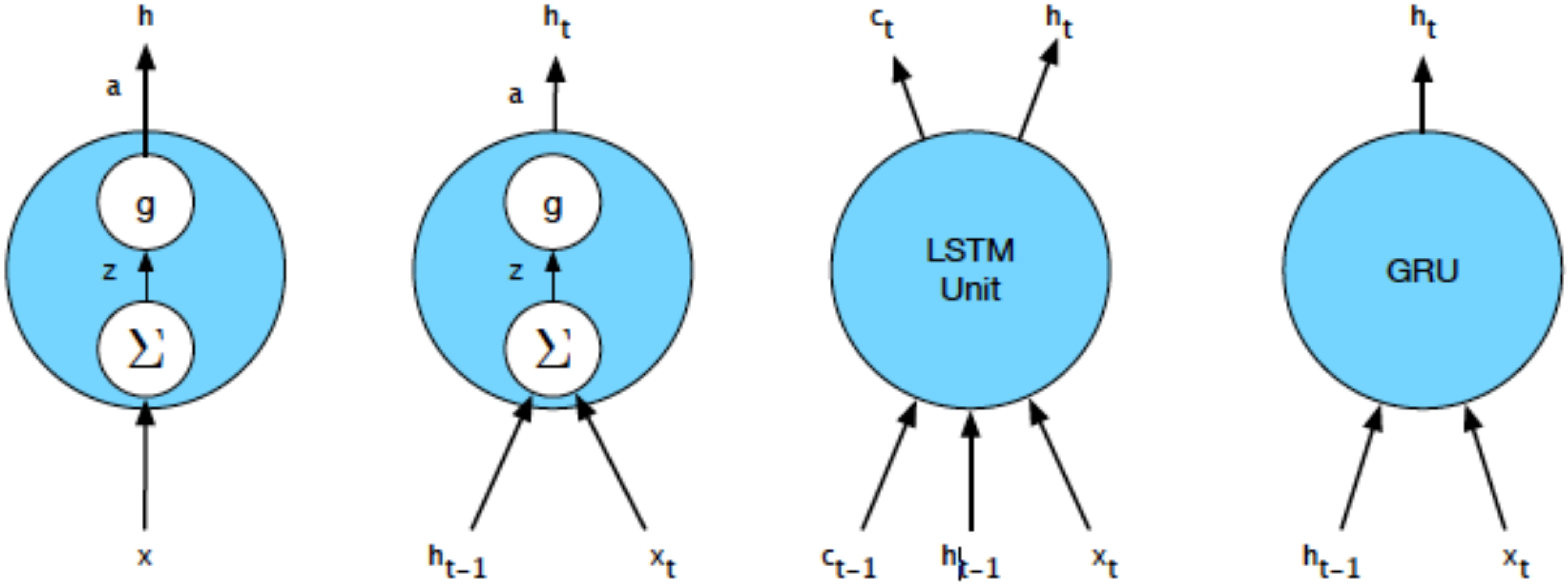
RECURRENT LANGUAGE MODELS



Digital Design

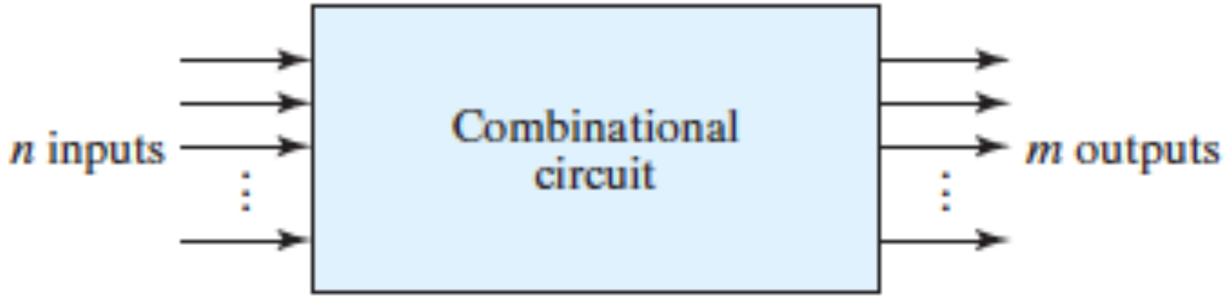


FIGURE 4.1
Block diagram of combinational circuit

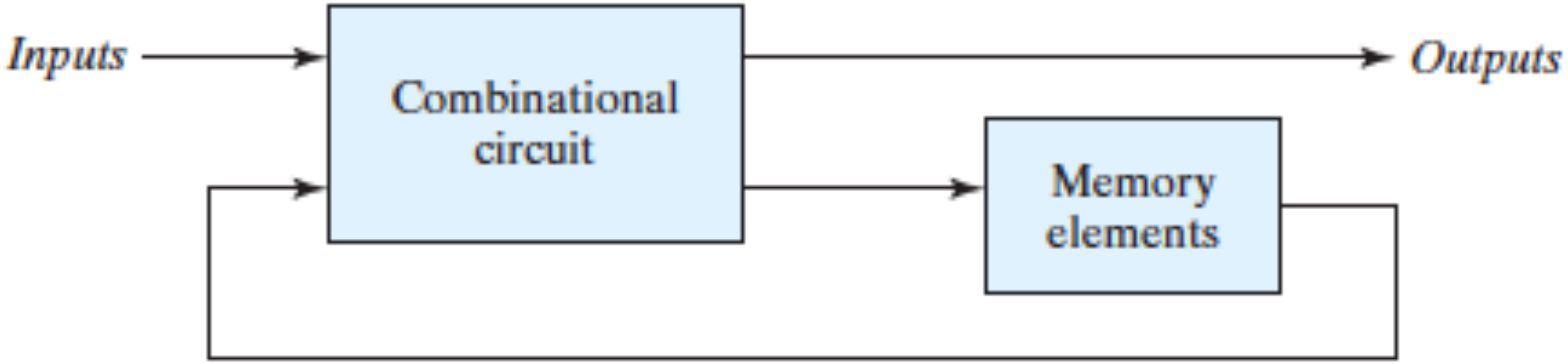
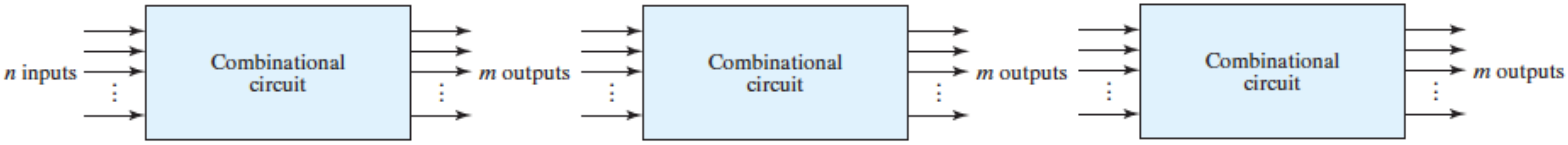
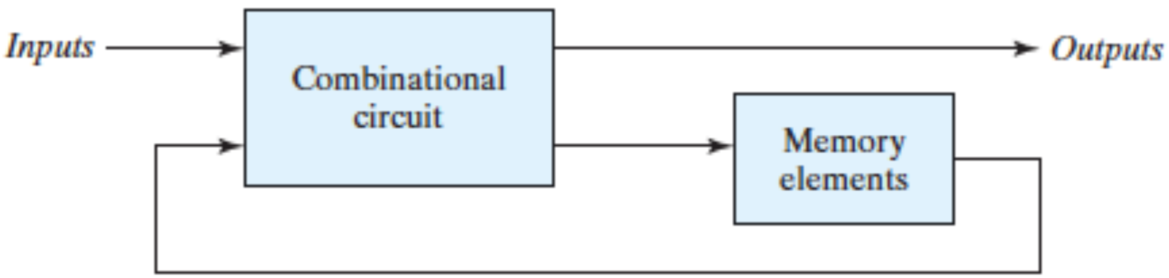


FIGURE 5.1
Block diagram of sequential circuit

Counter: 0 → 1 → 2 → ...



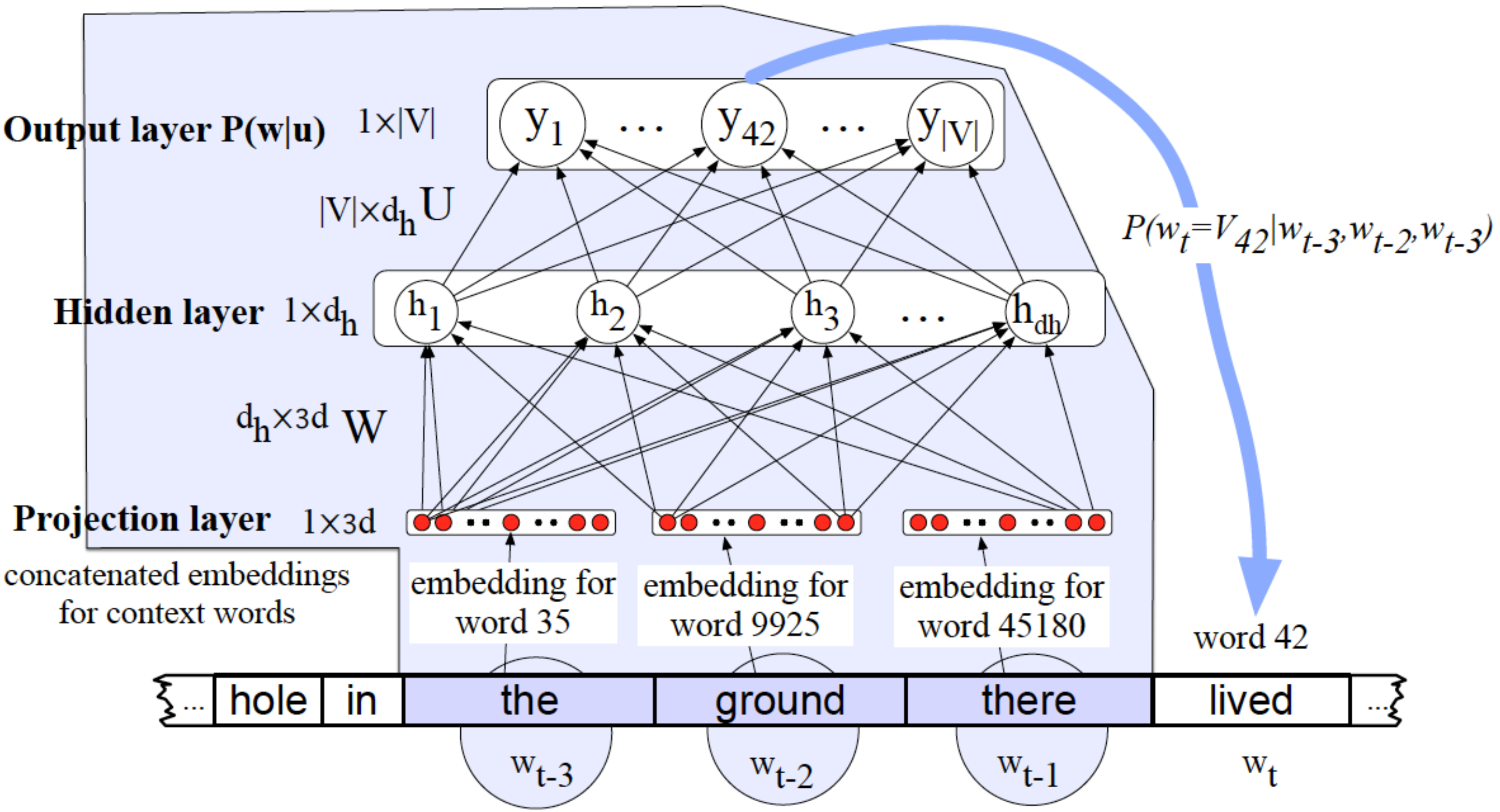
n inputs
Combinational circuit
m outputs
Combinational circuit
m outputs
Combinational circuit
m outputs



Inputs
Combinational circuit
Outputs
Memory elements

Neural Language Model

Bengio, Yoshua, et al. "A neural probabilistic language model." The journal of machine learning research 3 (2003): 1137-1155.



Language is Temporal

We already saw that previous context matters!

Unigram → Bigram → Trigram → …

Conversations

Social Timelines

Social Feeds

Language is Temporal

We already saw that previous context matters!

Unigram → Bigram → Trigram → …

Neural LM

- The same in this respect!
- Only different method (approach)
- But …

Language is Temporal

We already saw that previous context matters!

Unigram → Bigram → Trigram → …

Neural LM

- The same in this respect!
- Only different method (approach)
- But it offers a modular structure!

Neural LM

Neural LM
- But it offers a modular structure!
- We can
  - Stack them
  - Connect different parts
  - Create loopback connection

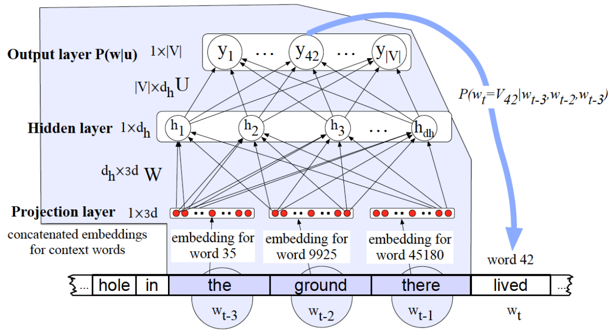
Recurrent Neural LM

Bengio's Neural LM

- Fixed Length Context Window (n-gram)
- Longer Stream
  - Sliding the Context Window

"we are in natural language processing class"

[we are in]
[are in natural]
[in natural language]
[natural language processing]
[language processing class]



Recurrent Neural LM

Bengio's Neural LM

Fixed Length Context Window (n-gram)

Longer Stream

Sliding the Context Window

"we are in natural language processing class"

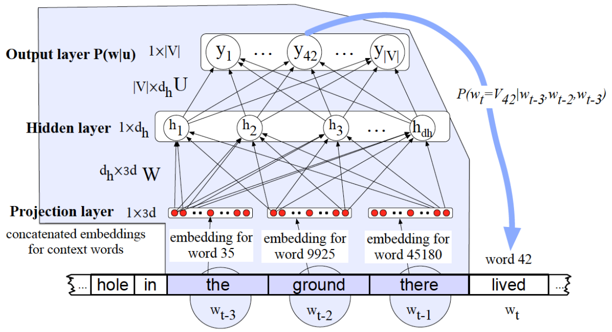
[we are in]

[are in natural]

[in natural language]

[natural language processing]

[language processing class]



Overlapping sliding carries some context from history

Recurrent Neural LM

Bengio's Neural LM

Fixed Length Context Window (n-gram)

Longer Stream

Sliding the Context Window

"we are in natural language processing class"

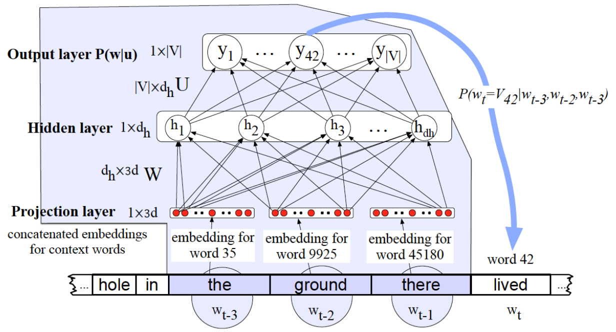
[we are in]

[are in natural]

[in natural language]

[natural language processing]

[language processing class]



But not from far distances!

Recurrent Neural LM

Bengio's Neural LM

Fixed Length Context Window (n-gram)

Longer Stream

Sliding the Context Window

"we are in natural language processing class"

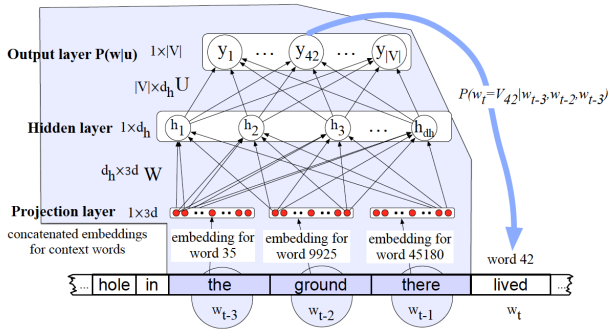
[language processing class]

[are in natural]

[we are in]

[in natural language]

[natural language processing]



Also, inputs are independent!

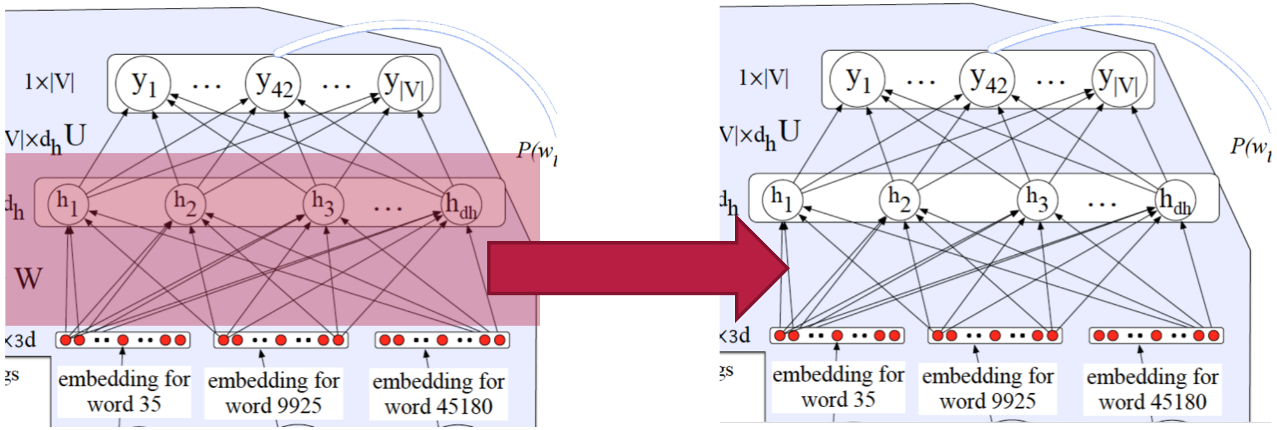
Recurrent Neural LM

Bengio's Neural LM

- Fixed Length Context Window (n-gram)
- Longer Stream
  - Sliding the Context Window

"we are in natural language processing class"

[we are in]
[are in natural]
[in natural language]
[natural language processing]
[language processing class]



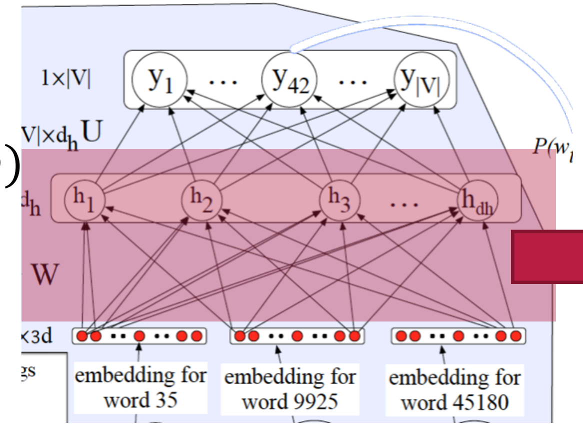
Recurrent Neural LM

$$Y = softmax(hU)$$

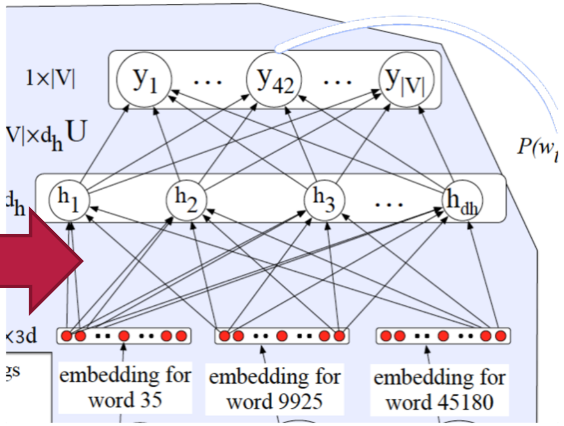
$$\text{h} = \tanh(X\,W + b)$$

$$X\,W + b$$

$$X$$



+ hV



$$Y' = softmax(h'U)$$

$$\text{h}' = \tanh(X\,W + b + hV)$$

$$X'\,W + b$$

$$X'$$

$$f(h): what\ part\ of\ history\ and\ how\ to\ transfer\ to\ future!$$

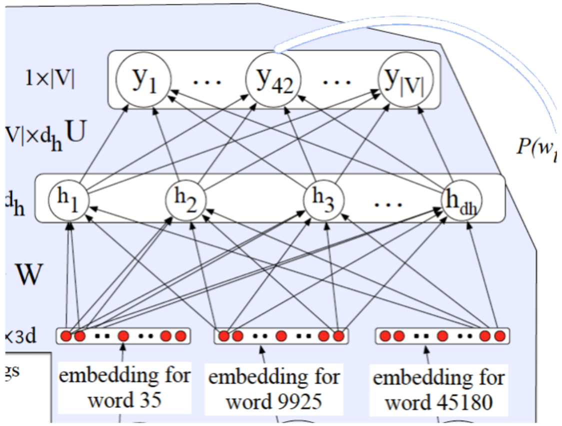
Recurrent Neural LM

$$Y_n = softmax(hnU)$$

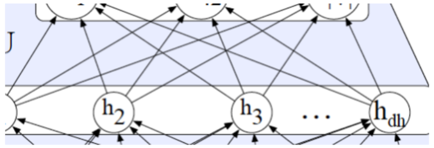
$$h_n = \tanh(Xn\,W + b + h_{n-1}V)$$

$$X_n\,W + b$$

$$X_n$$



$h_{n-1}V$



Recurrent Neural LM

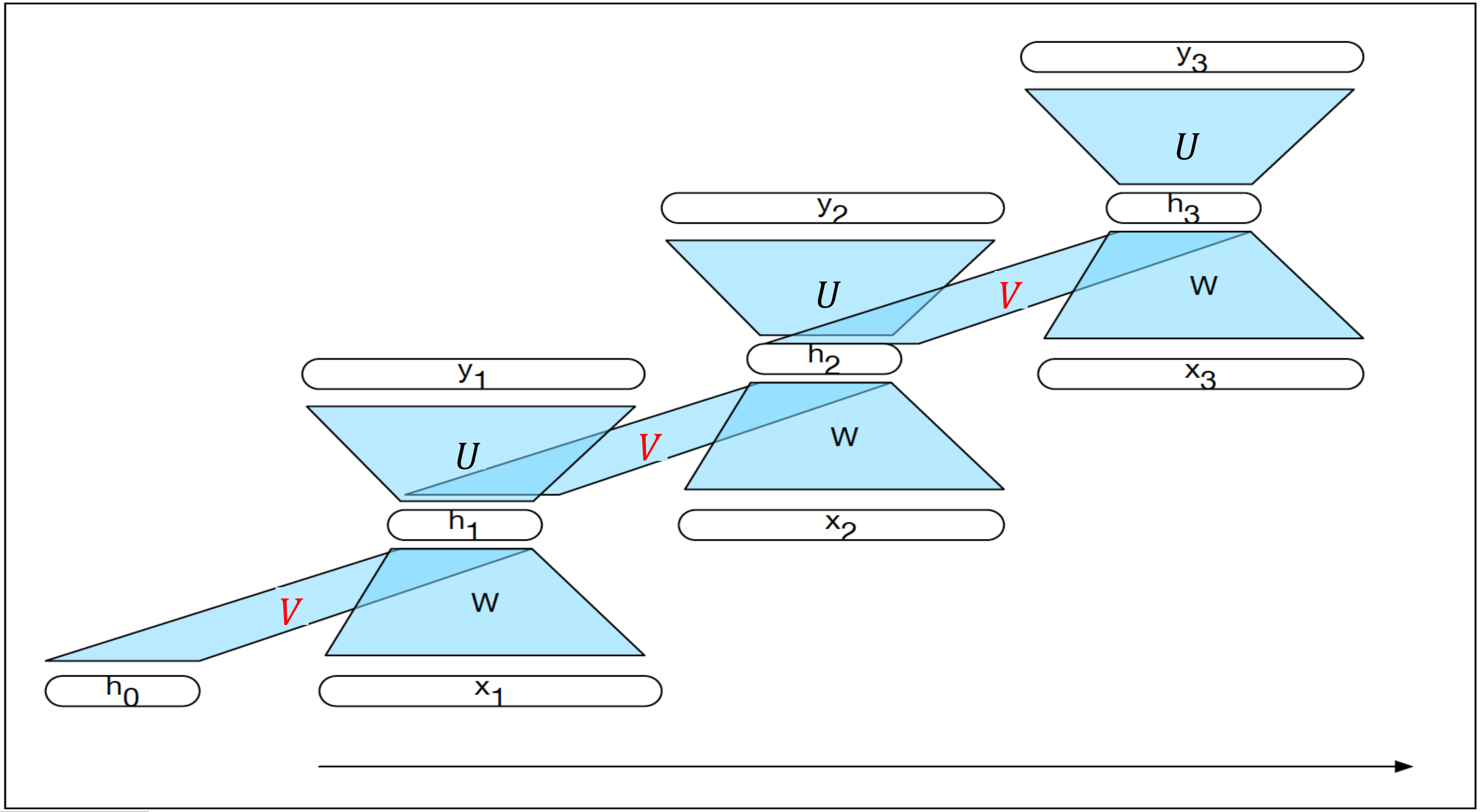


Figure 9.5 A simple recurrent neural network shown unrolled in time. Network layers are copied for each time step, while the weights U, V and W are shared in common across all time steps.

Recurrent Neural LM: Training

$Y_n = \sigma'(h_n U + b);\;\; hn = \sigma(x_n W + b + hn\_{}_1 V)$

L = distance (Y_n – Y*)

$$\frac{\partial L}{\partial v \in V} = ?$$

$$\frac{\partial L}{\partial w \in W} = ?$$

$$\frac{\partial L}{\partial u \in U} = ?$$

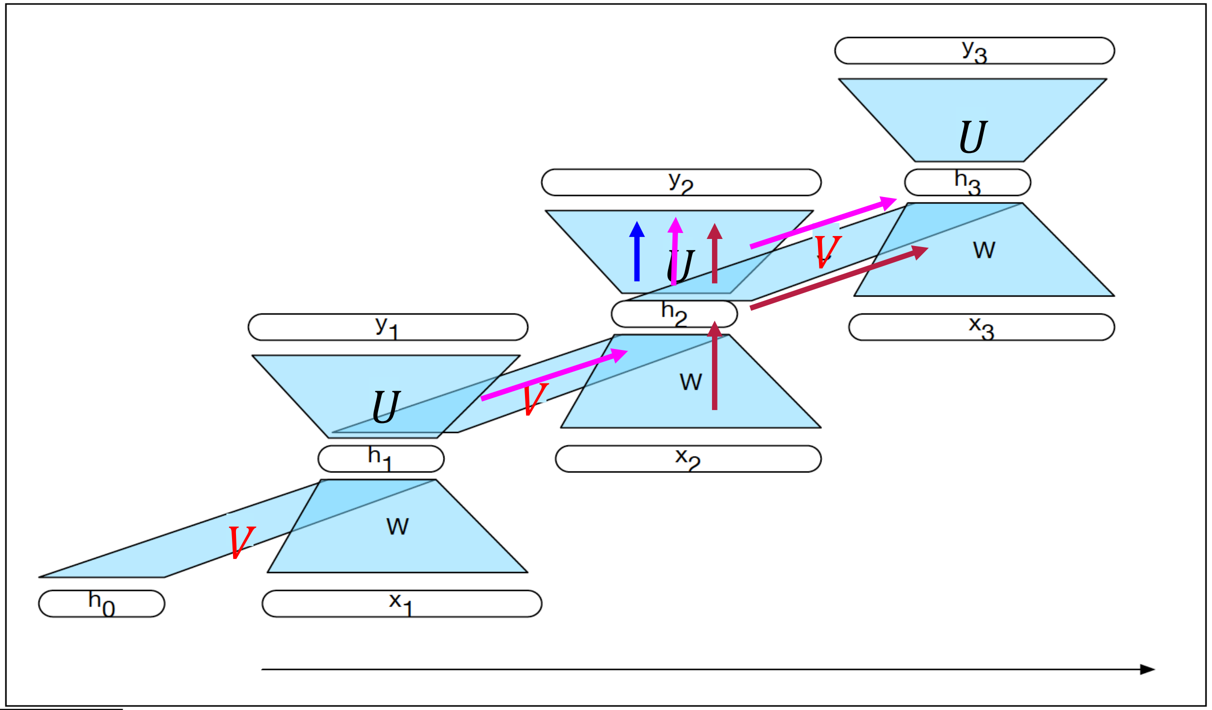


Figure 9.5 A simple recurrent neural network shown unrolled in time. Network layers are copied for each time step, while the weights U, V and W are shared in common across all time steps.

Backpropagation Through Time (Werbos 1974, Rumelhart et al. 1986, Werbos 1990).

Recurrent Neural LM: Training

Mikolov, T. et al (2010). Recurrent neural network based language model. In INTERSPEECH 2010, 1045–1048.

The limited context constraint inherent in both N-gram models and sliding window approaches is avoided since the hidden state embodies information about all of the preceding words all the way back to the beginning of the sequence.

True Markovian!

Recurrent Neural LM: Autoregression

Generate a sentence by the trained language model:

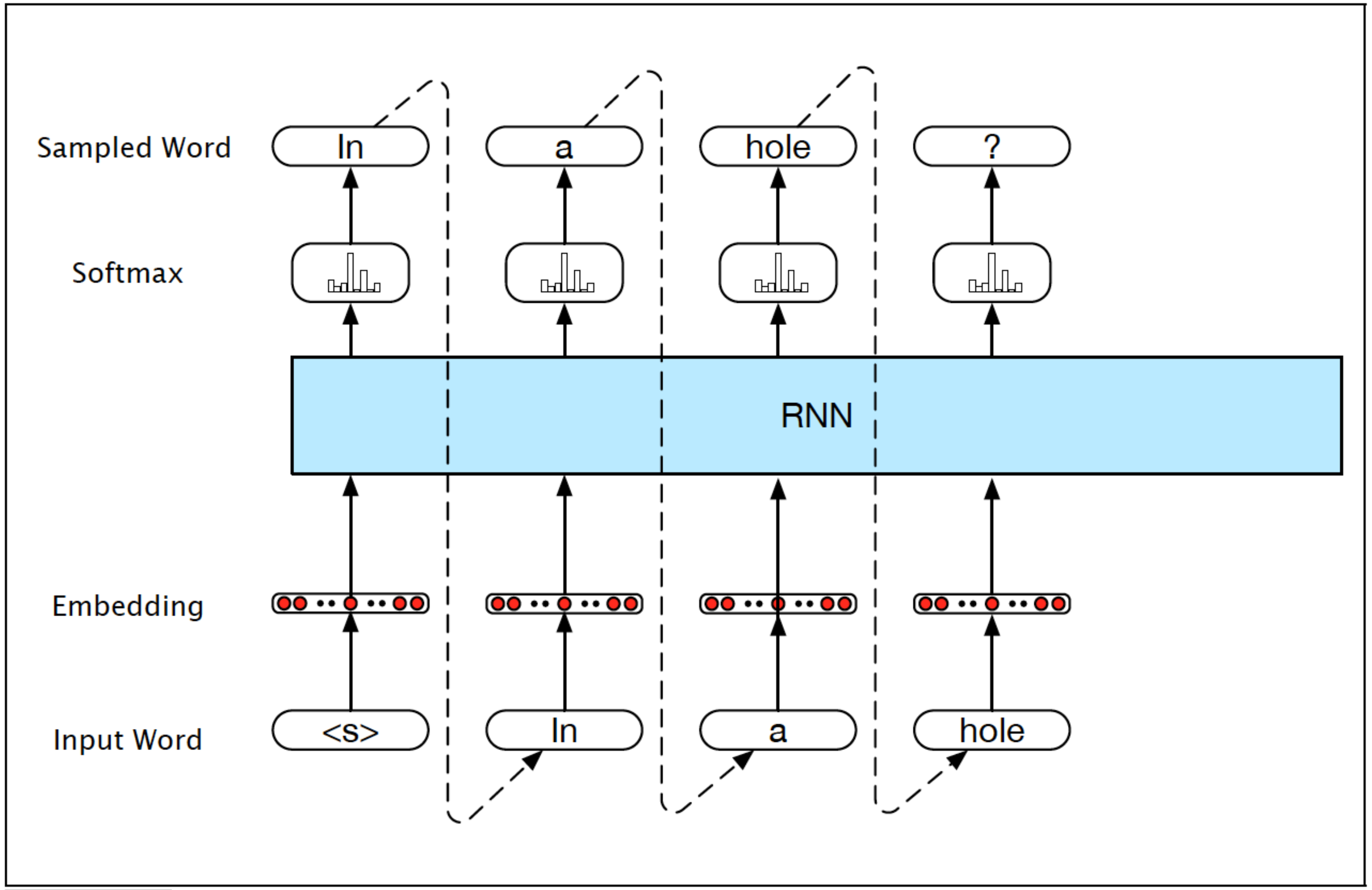


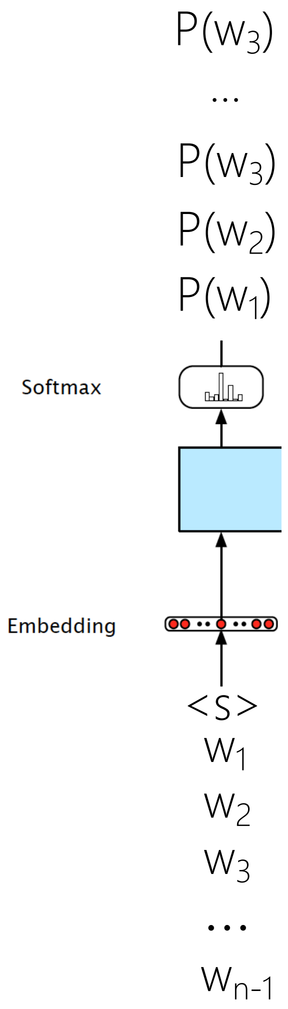
Figure 9.7 Autoregressive generation with an RNN-based neural language model.

Recurrent Neural LM: Evaluation

Likelihood of the test text stream:

$P(w_1 w_2 w_3 .. w_n) = P(w_1)P(w_2|w_1)P(w_3|w_1 w_2) \ldots P(w_n|w_1 w_2 \ldots w_{n-1})$

$$\text{Perplexity}(w_1 w_2 w_3 .. w_n) = \sqrt[n]{\prod_{i=1}^{n} \frac{1}{P(w_1 w_2 w_3 .. wn)}}$$



Recurrent Neural LM: Application

- Part-of-Speech Tagging
- Sequence Classification

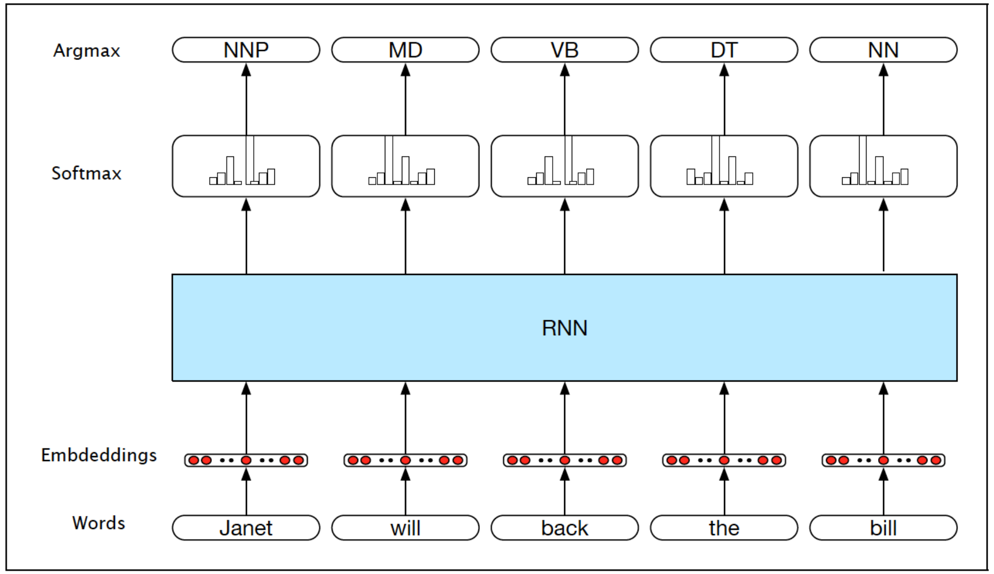


Figure 9.8 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

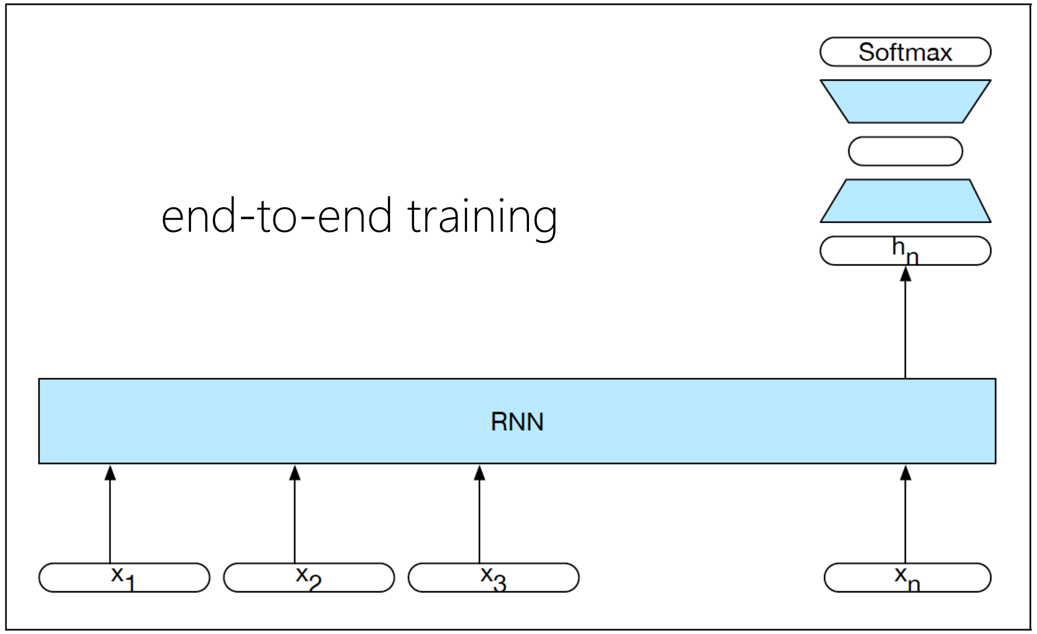


Figure 9.9 Sequence classification using a simple RNN combined with a feedforward network. The final hidden state from the RNN is used as the input to a feedforward network that performs the classification.

Recurrent Neural LM

Bidirectional

- We have the whole context
  (e.g., a sentence, short dialog, …)

- We have the future!
  From future to the past (backward)

Recurrent Neural LM: Bidirectional

$P(w_i | w_1 w_2 .. w_{i-1} \; w_{i+1} ... w_n)$

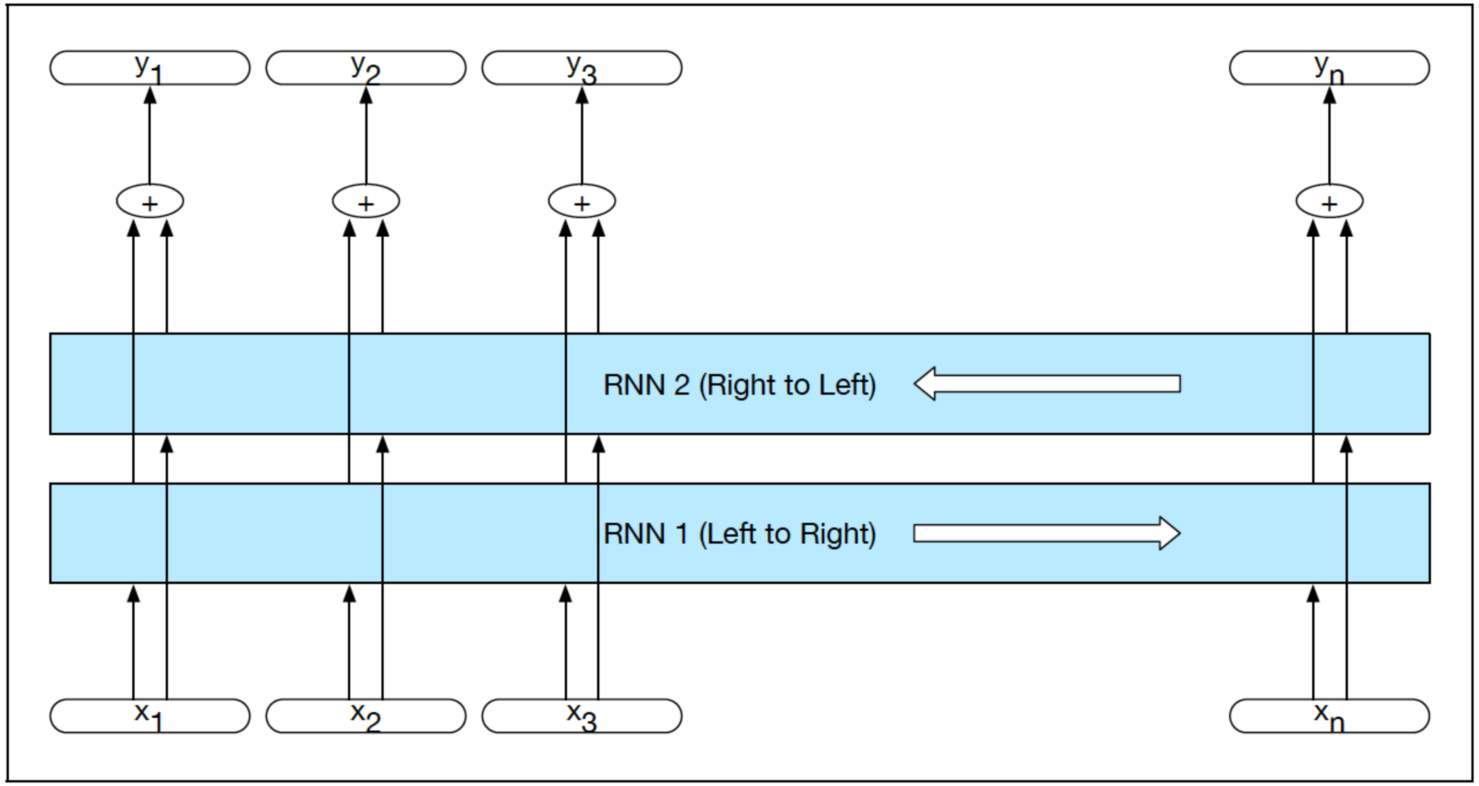


Figure 9.11 A bidirectional RNN. Separate models are trained in the forward and backward directions with the output of each model at each time point concatenated to represent the state of affairs at that point in time. The box wrapped around the forward and backward network emphasizes the modular nature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

Recurrent Neural LM: Bidirectional

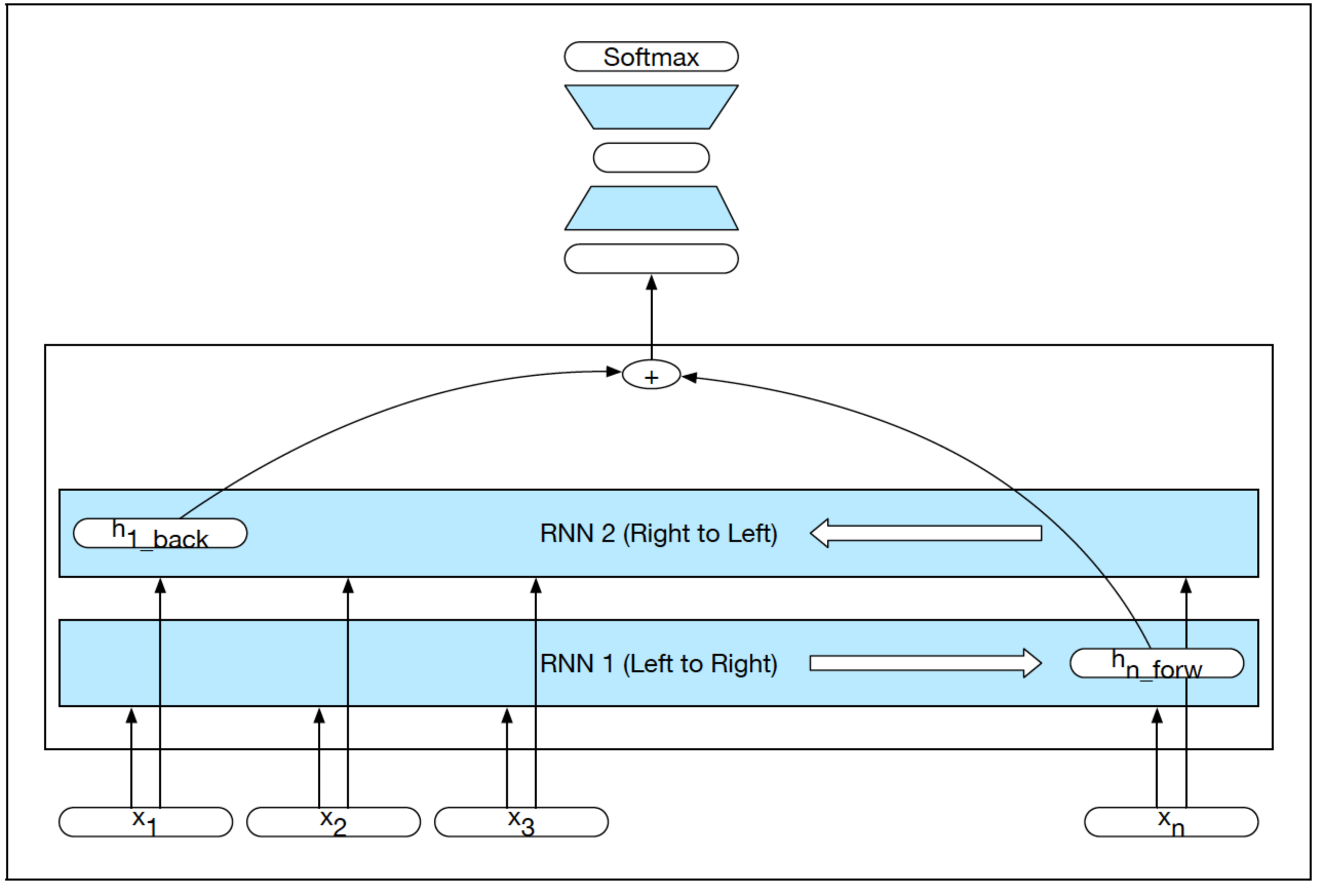


Figure 9.12 A bidirectional RNN for sequence classification. The final hidden units from the forward and backward passes are combined to represent the entire sequence. This combined representation serves as input to the subsequent classifier.

Recurrent Neural LM: Bidirectional

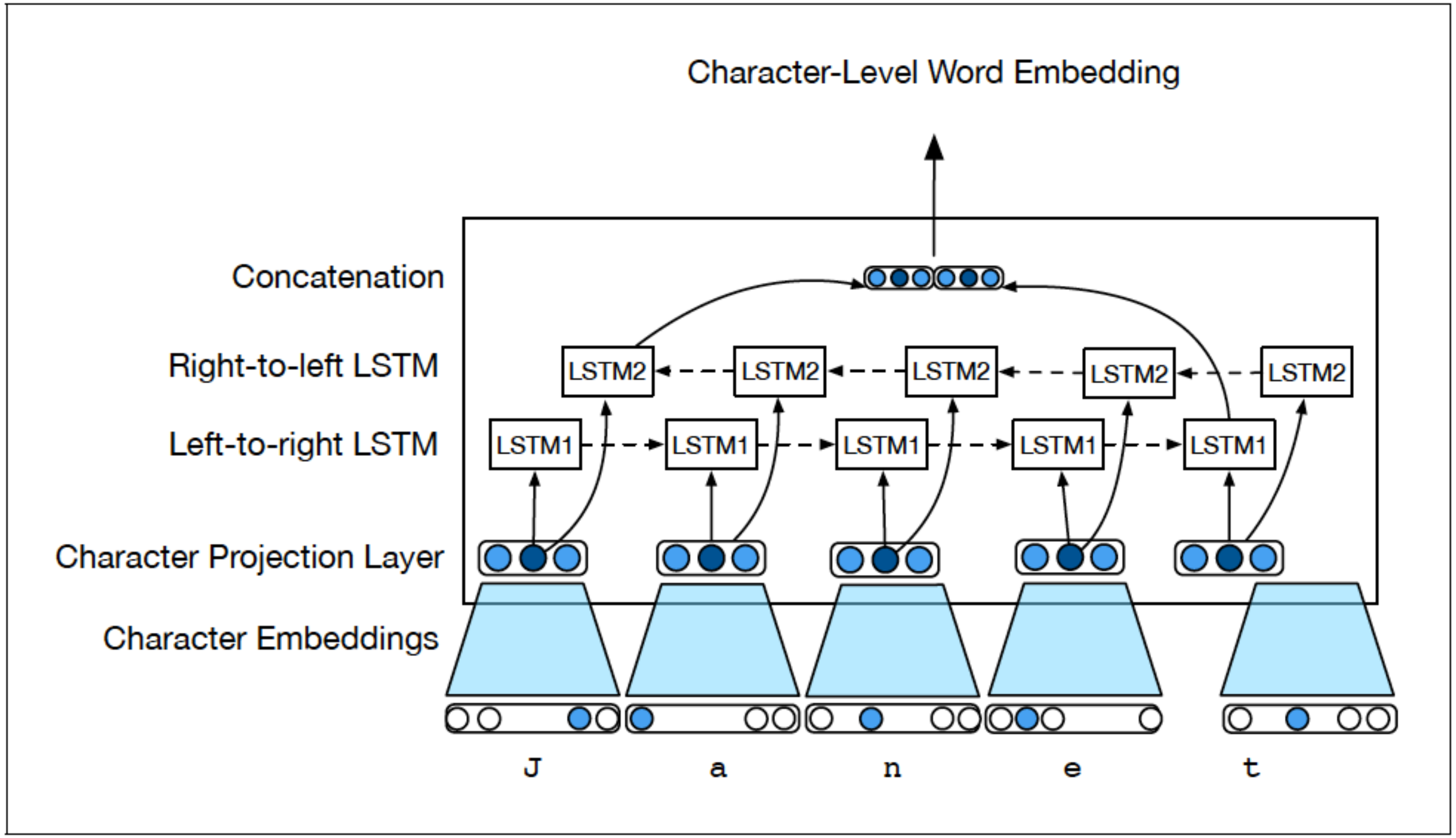


Figure 9.16 Bi-RNN accepts word character sequences and emits embeddings derived from a forward and backward pass over the sequence. The network itself is trained in the context of a larger end-application where the loss is propagated all the way through to the character vector embeddings.

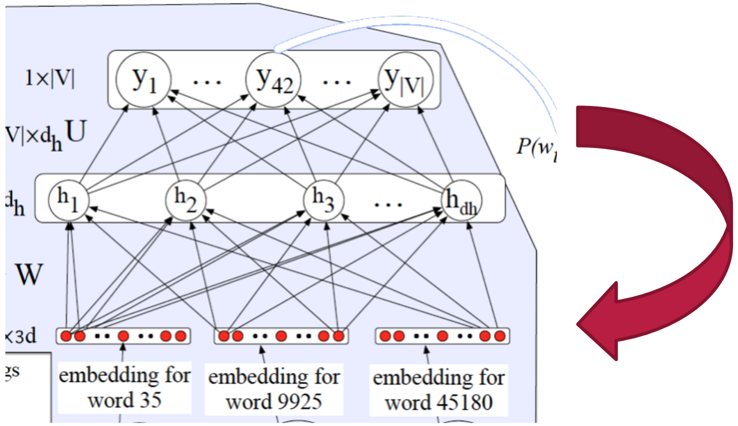
Managing Context in RNNs

$$Y_n = softmax(hnU)$$

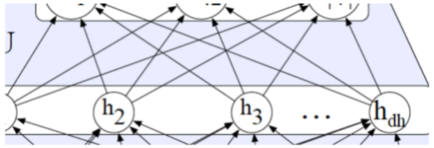
$$\mathrm{h_n} = \tanh(Xn\,W + b + h_{n\,\_\,1}V)$$

$$X_n\,W + b$$

$$X_n$$



$h_{n\,\_\,1}V$



fully consider the whole history so far

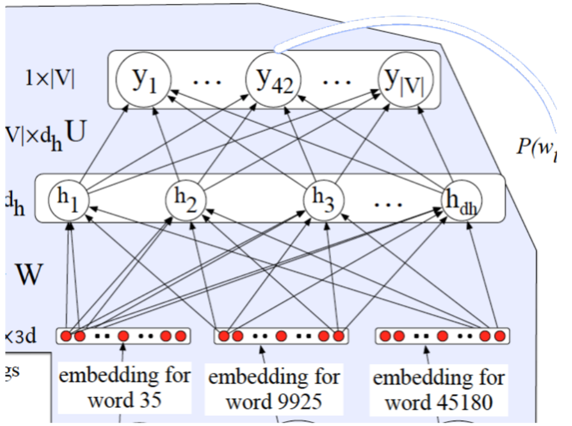
Managing Context in RNNs

$Y_n = softmax(hnU)$

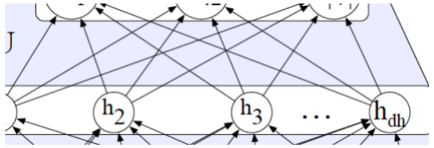
$\mathrm{h_n} = \tanh(Xn\,W + b + h_{n\,-\,1}V)$

$X_n\,W + b$

X_n



$h_{n\,-\,1}V$

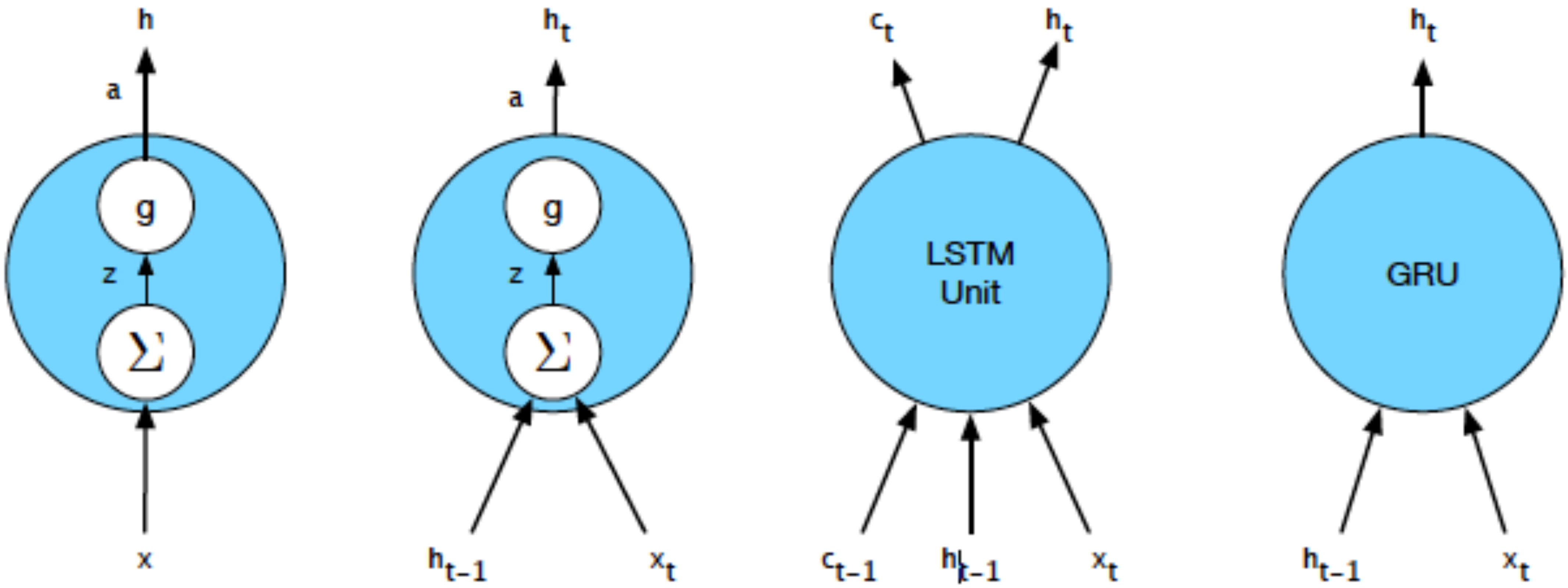


Forget: drop some part of history

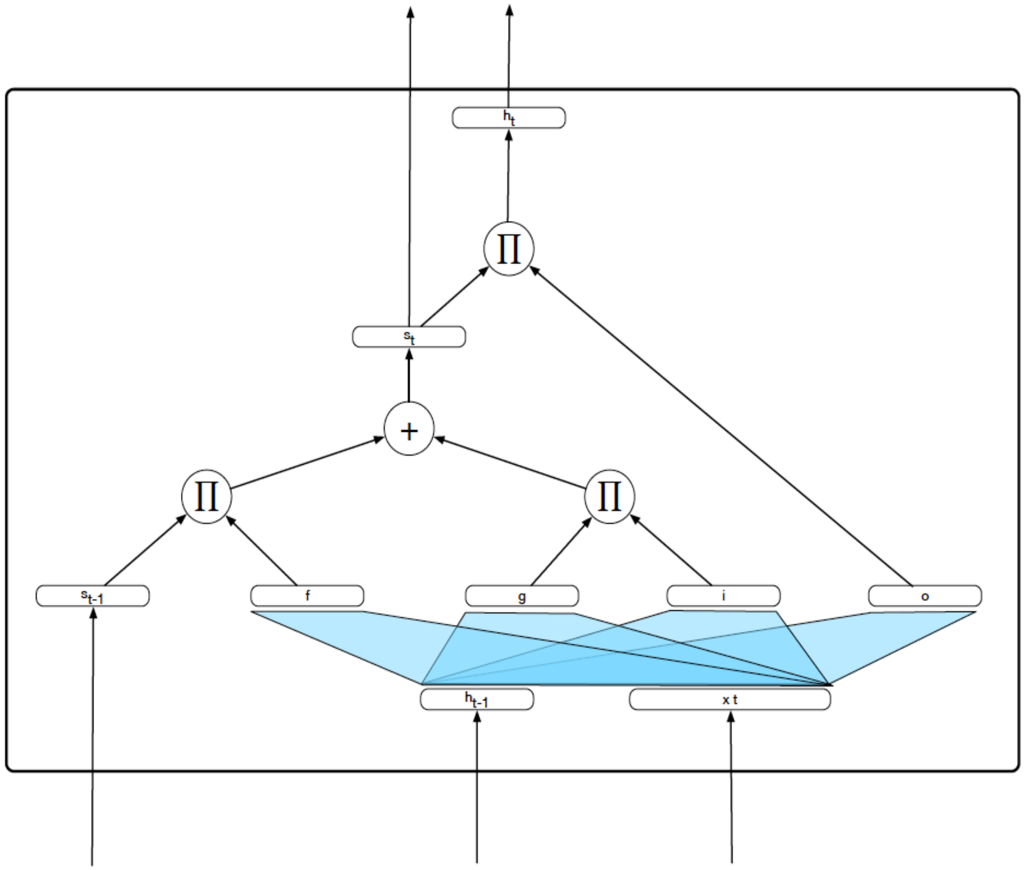
Remember: only consider the important ones

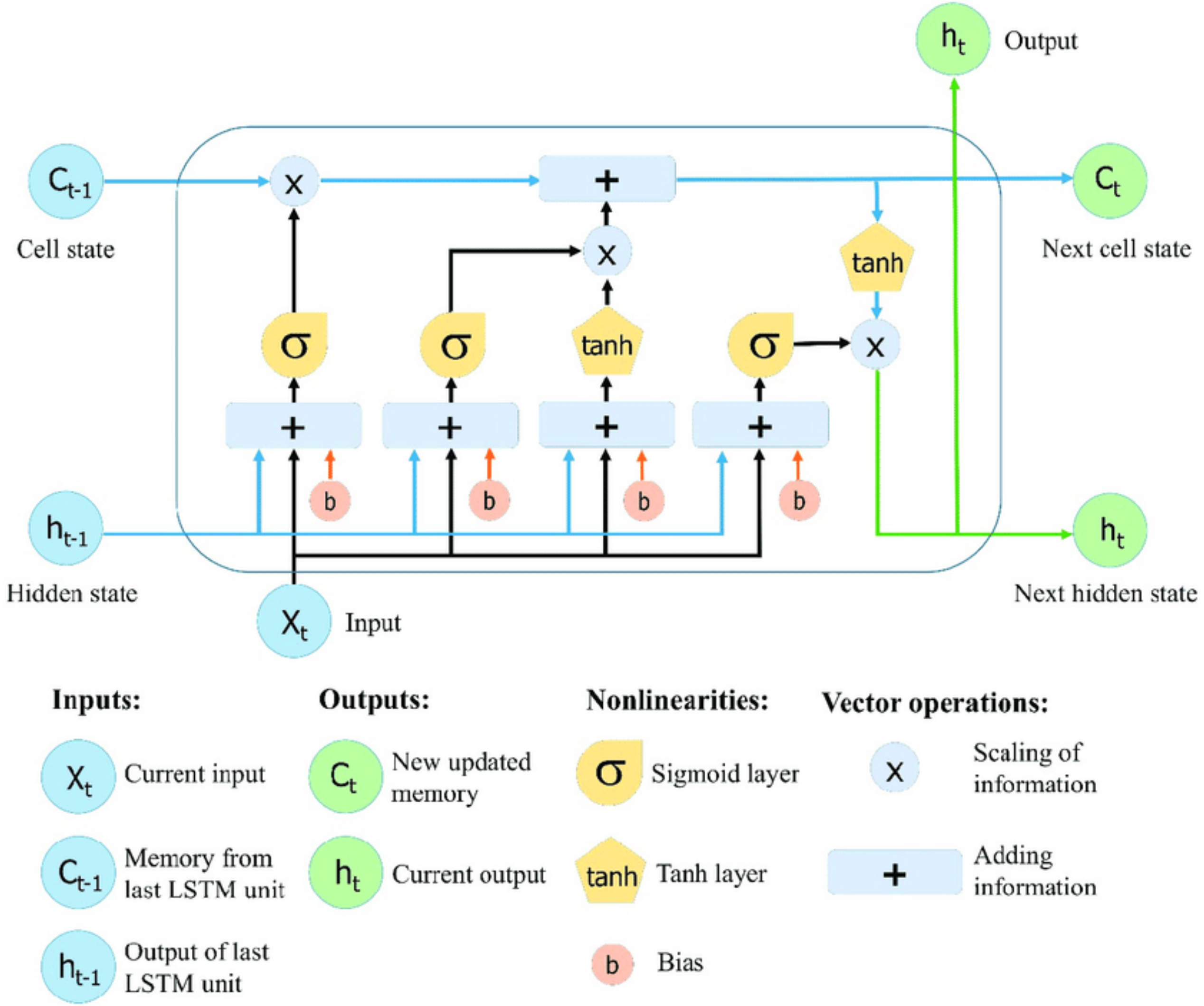
Recall: try to remember those that forgotten!

Managing Context in RNNs



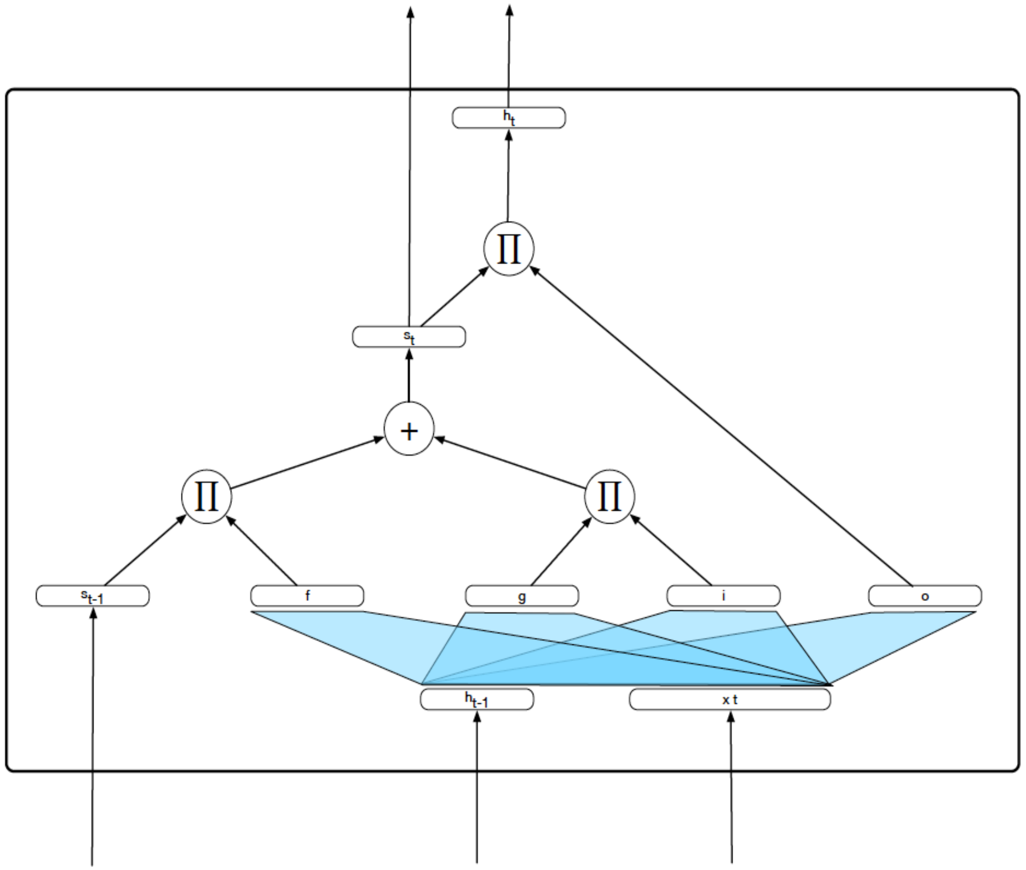
Long Short-Term Memory

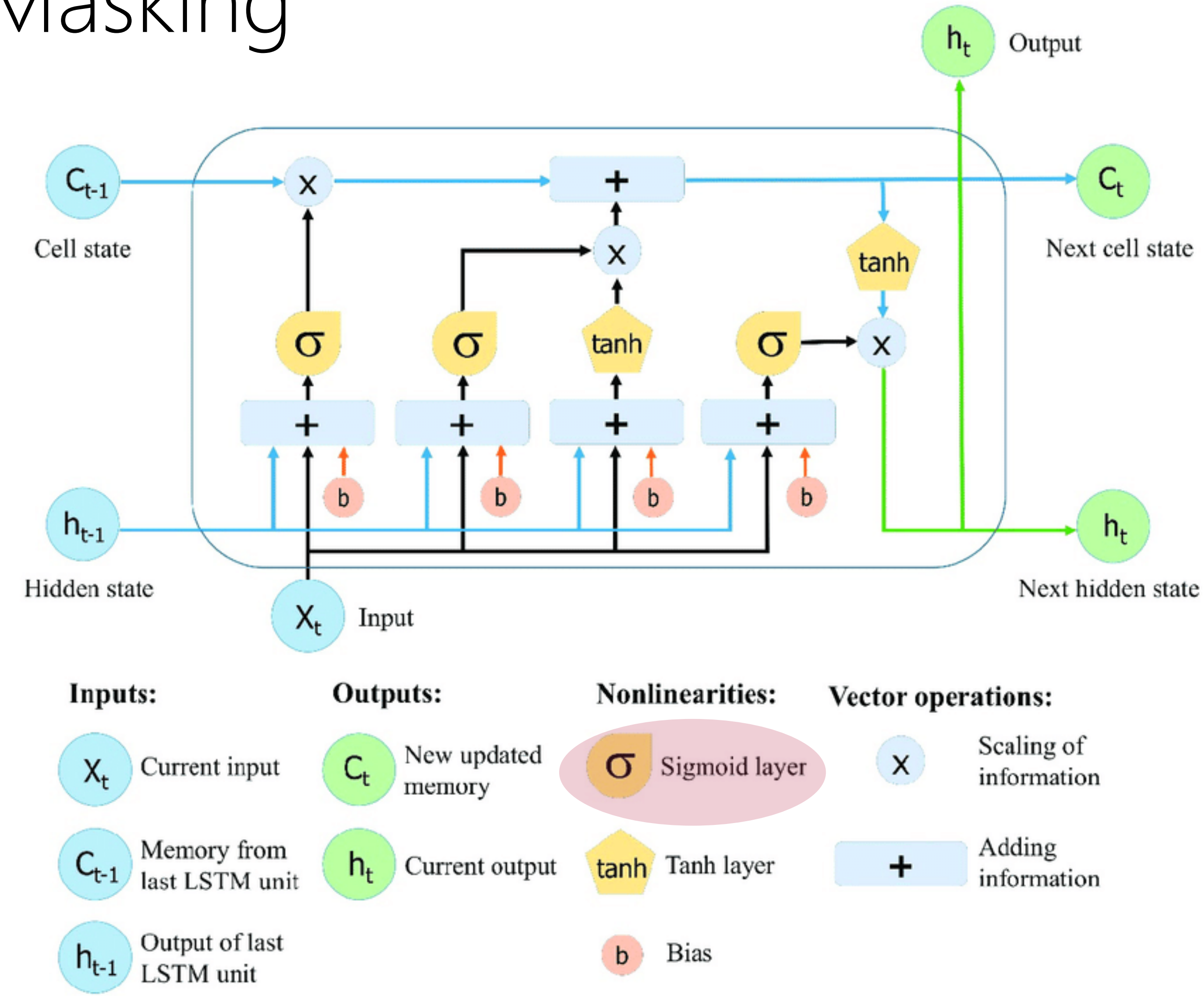




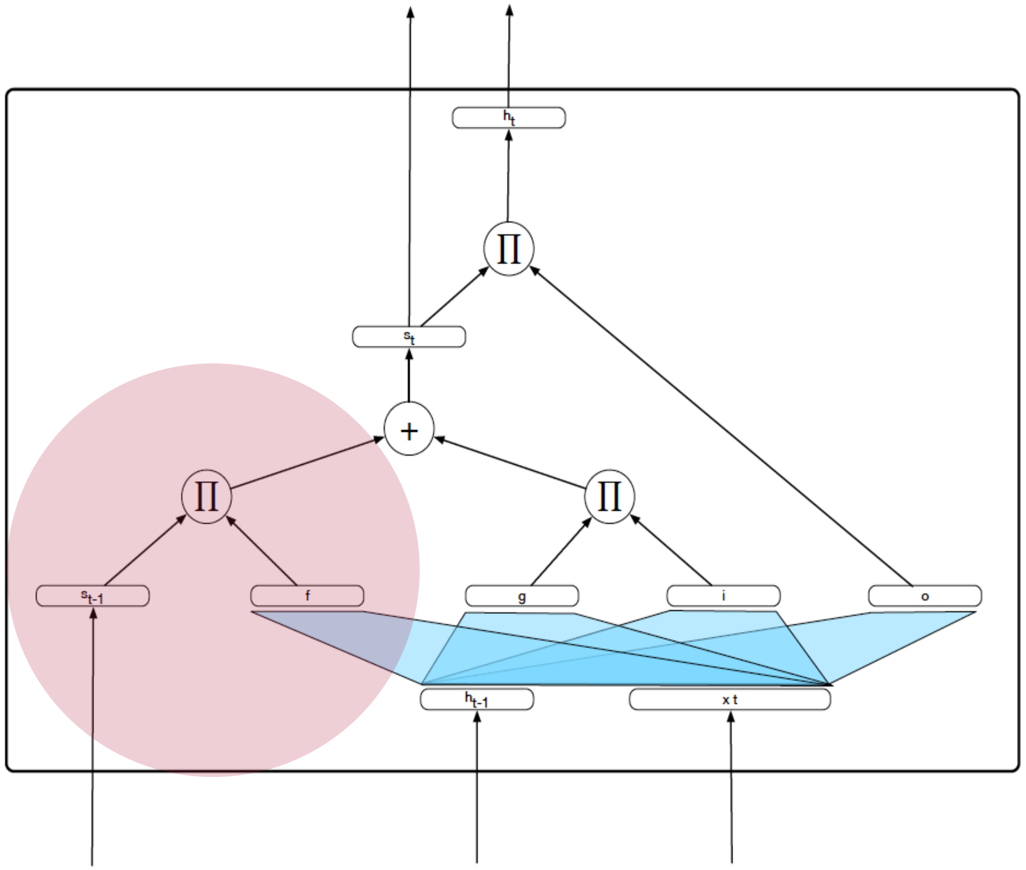
Long Short-Term Memory

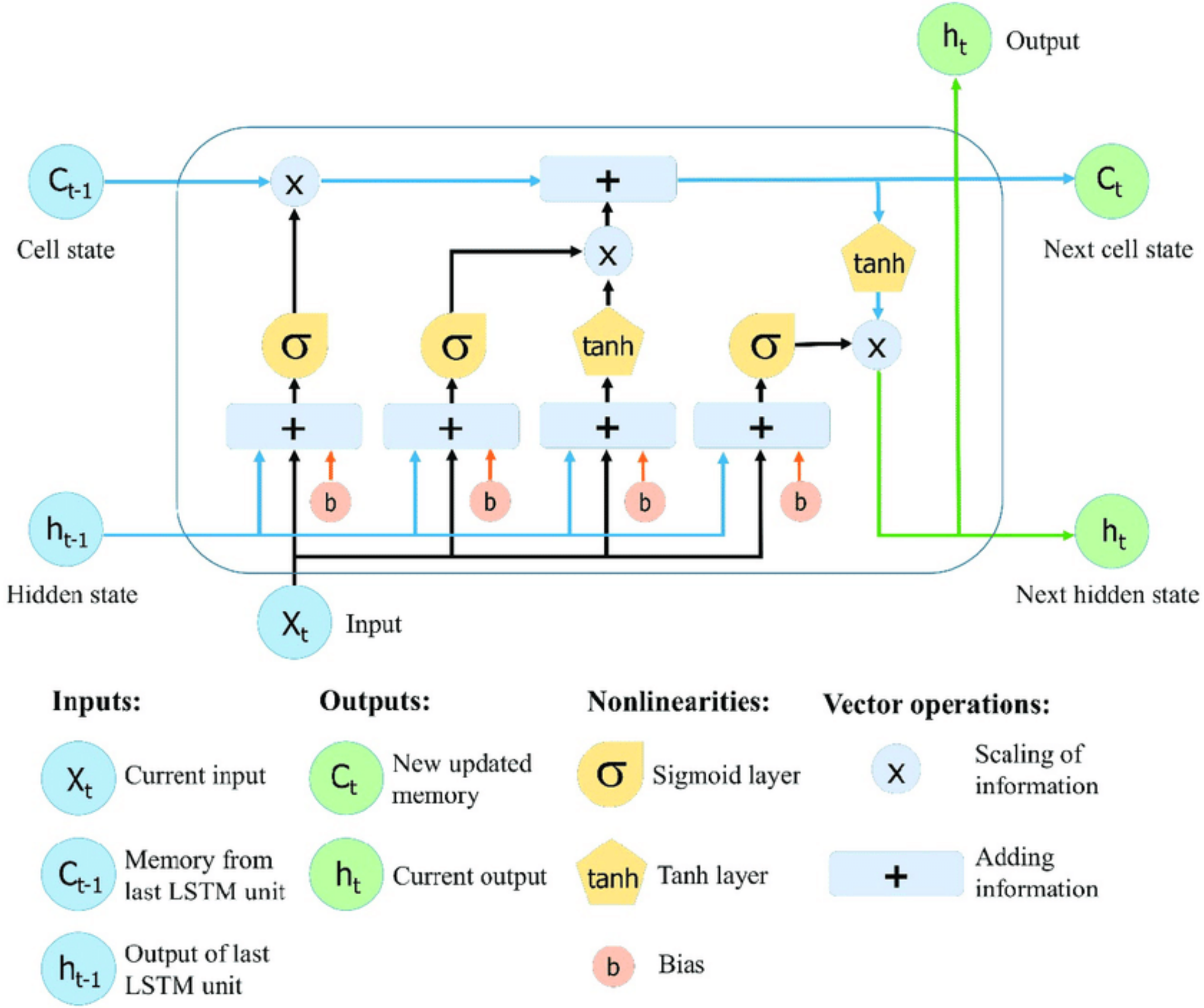
Sigmoid: Probabilistic Binary Masking



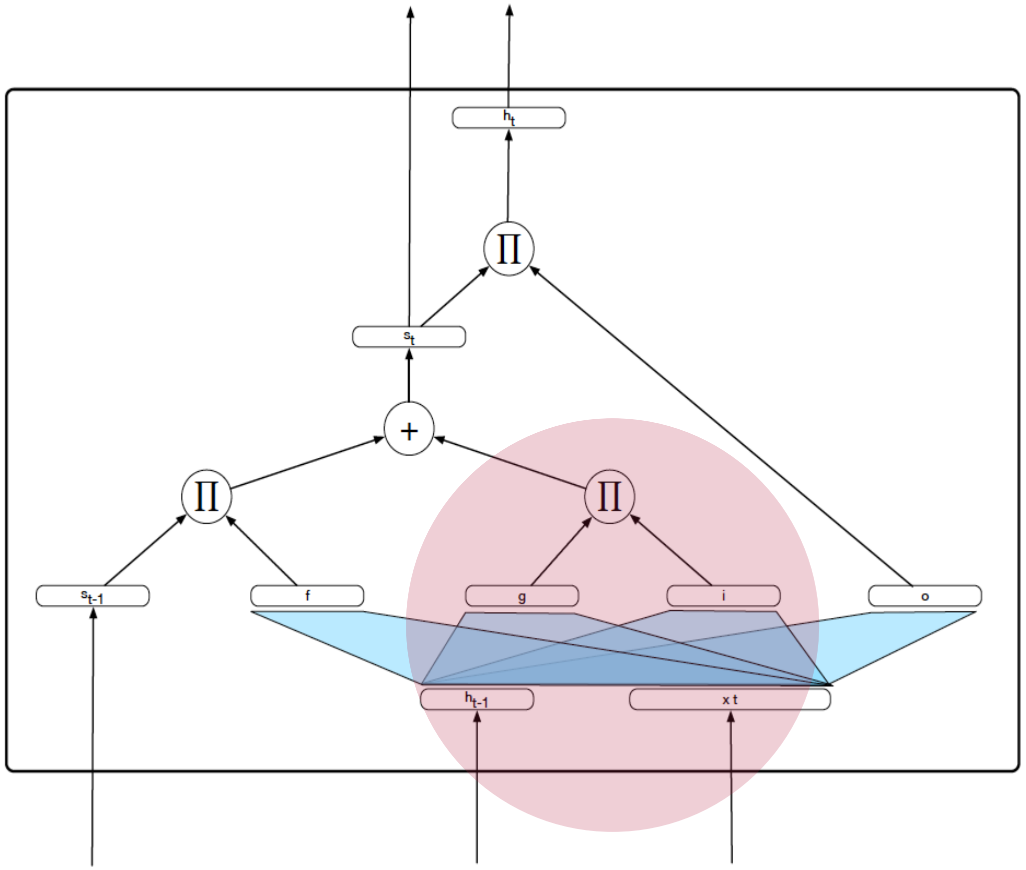


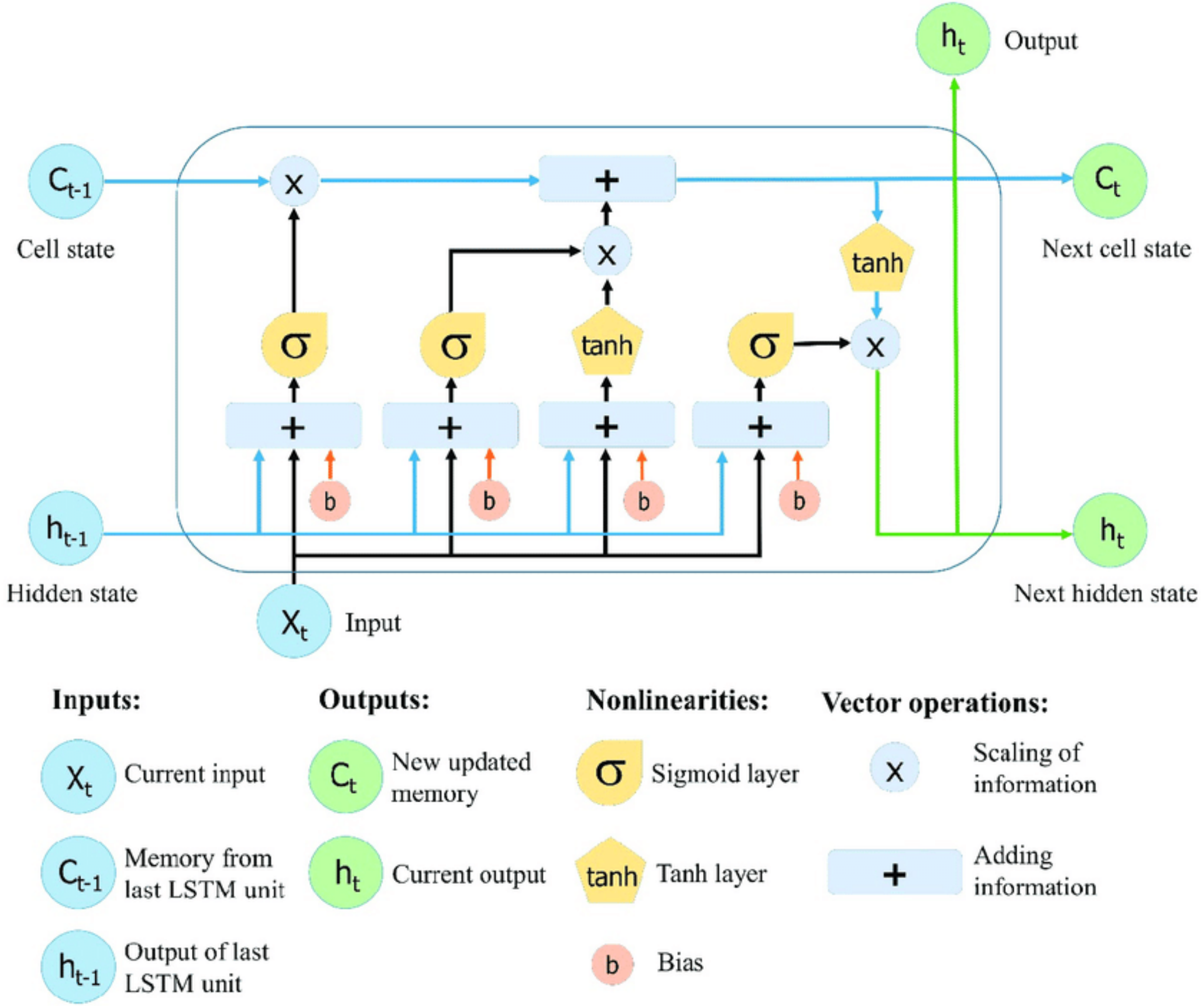
LSTM: Forget from the long memory



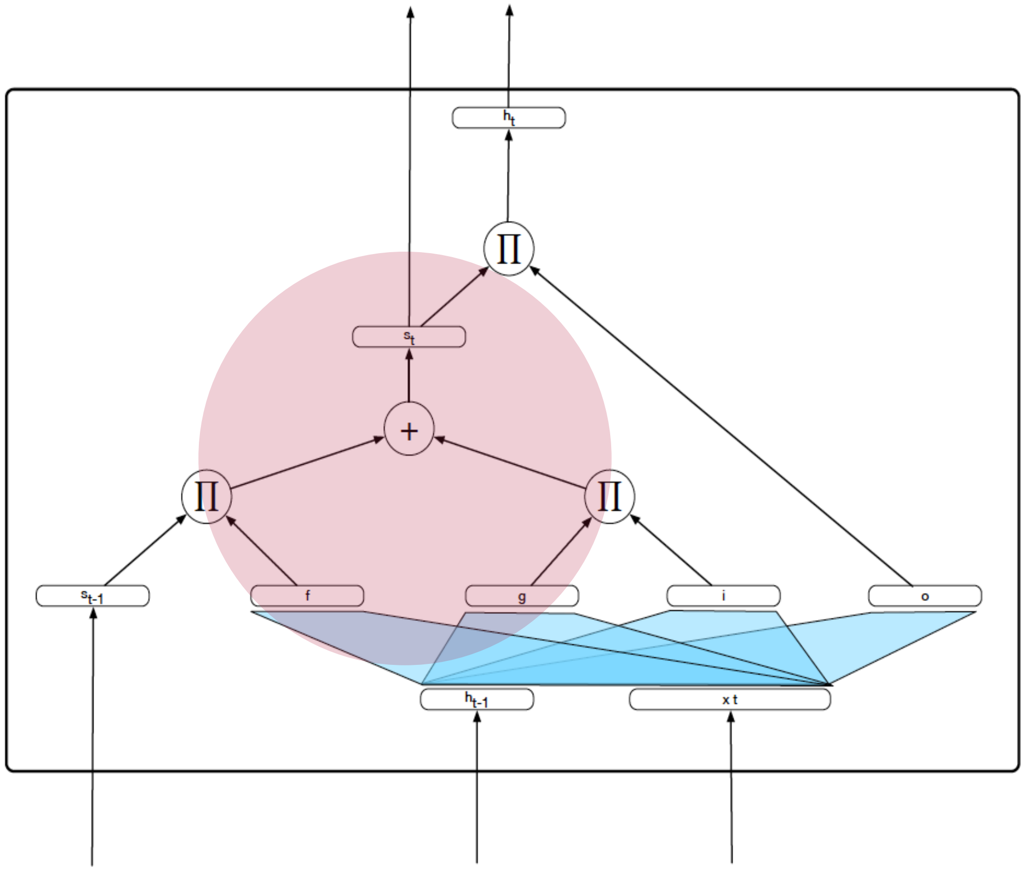


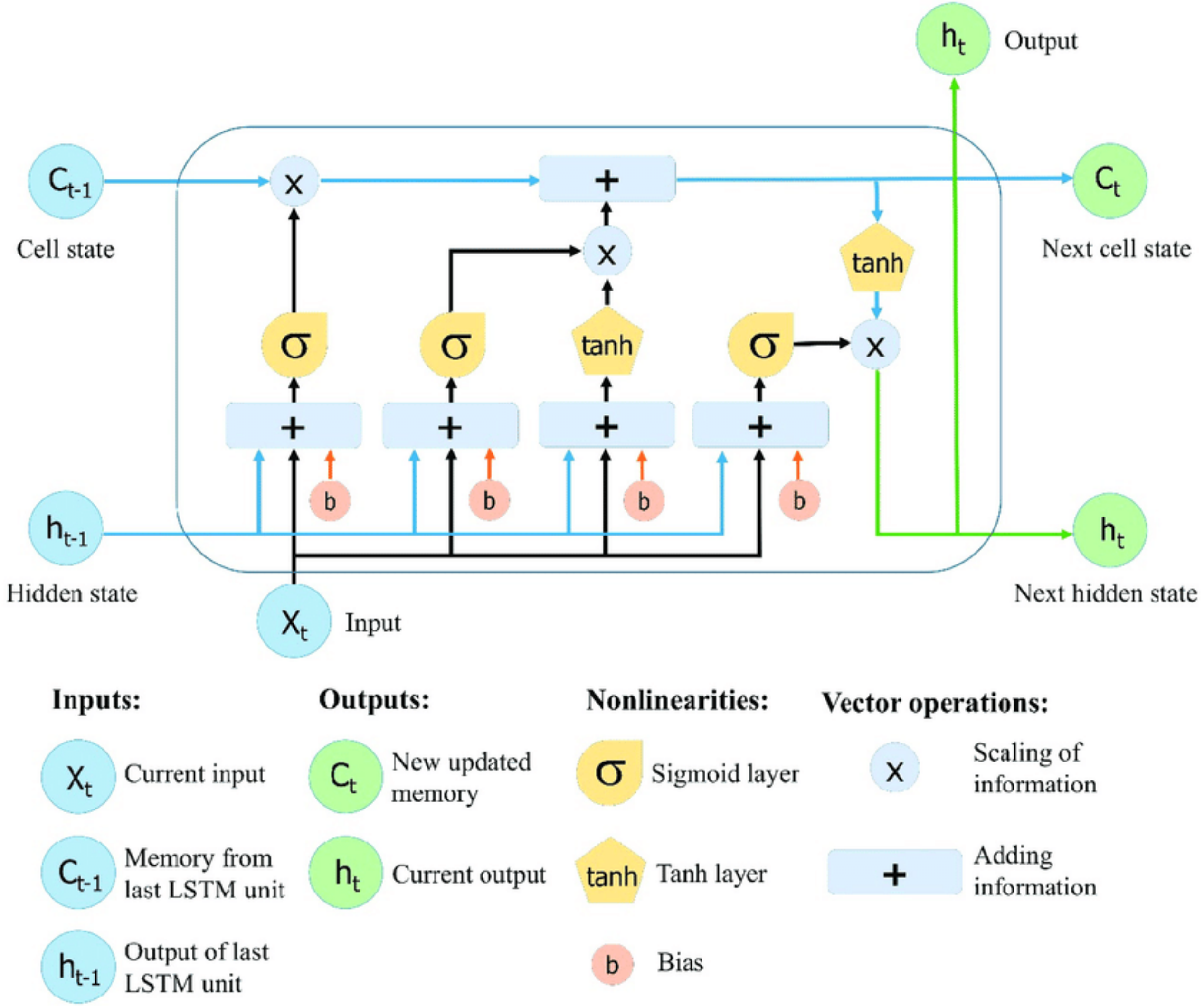
LSTM: Add to the long memory



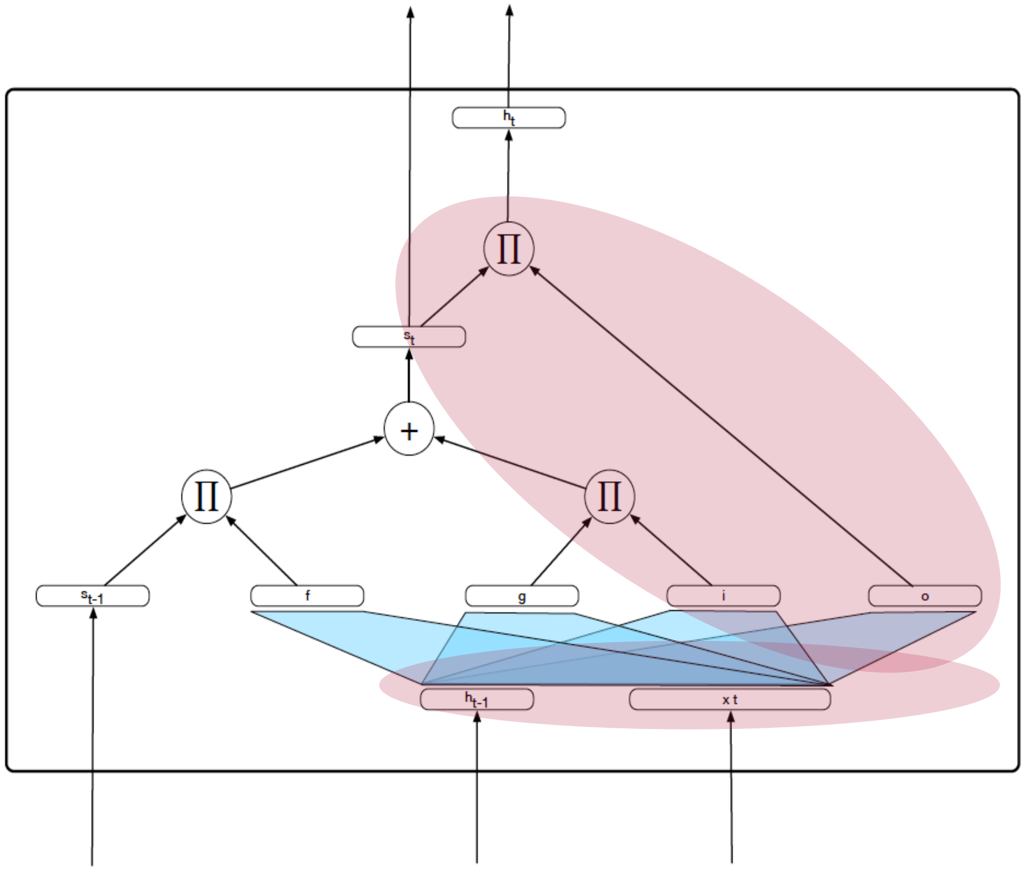


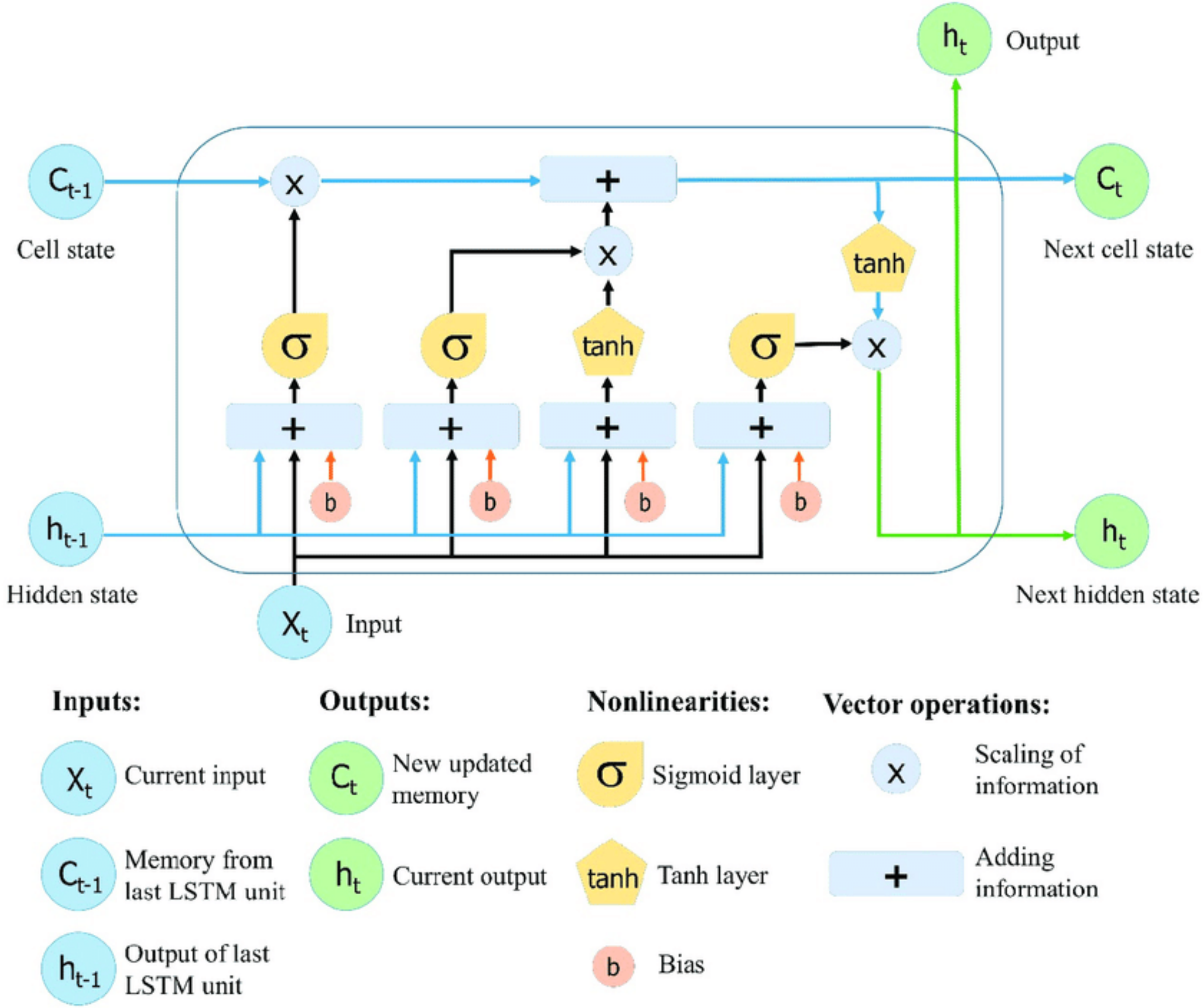
LSTM: Updated long memory





LSTM: Short memory





Gated Recurrent Unit (GRU)

LSTM: too many weights!