ITCS 6156/8156 Fall 2023
Machine Learning

Recurrent Neural Networks

Instructor: Hongfei Xue
Email: hongfei.xue@charlotte.edu
Class Meeting: Mon & Wed, 4:00 PM – 5:15 PM, CHHS 376

UNIVERSITY OF NORTH CAROLINA CHARLOTTE

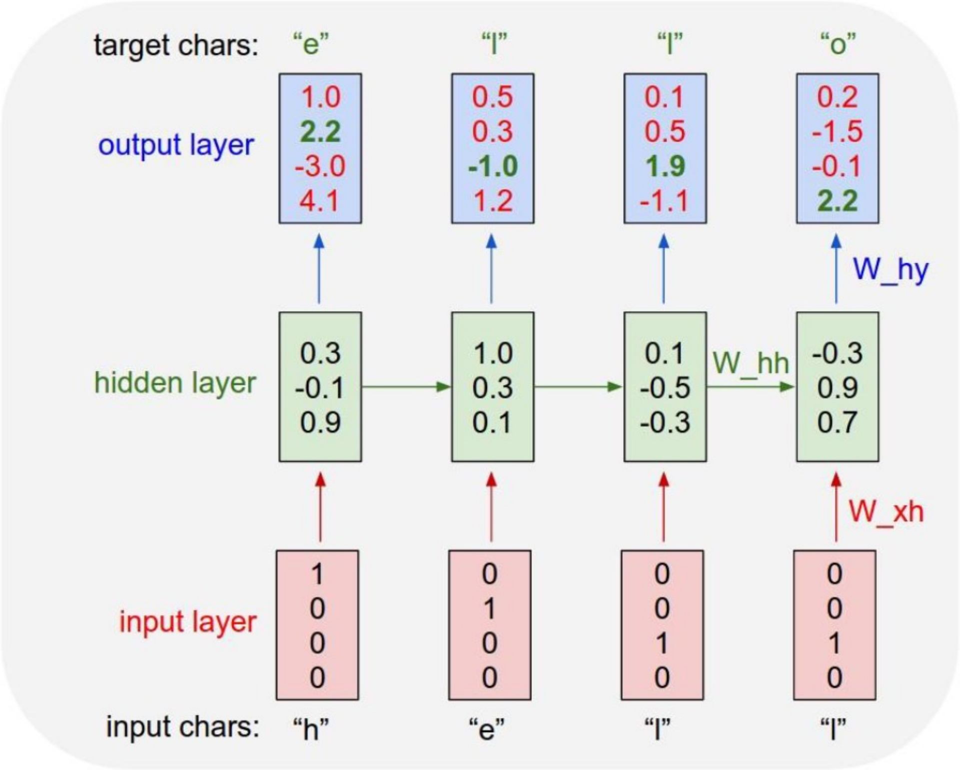
Some content in the slides is based on DeepMind's and Dr. Fei-Fei Li's lectures

Example: Character-level Language Model

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **"hello"**

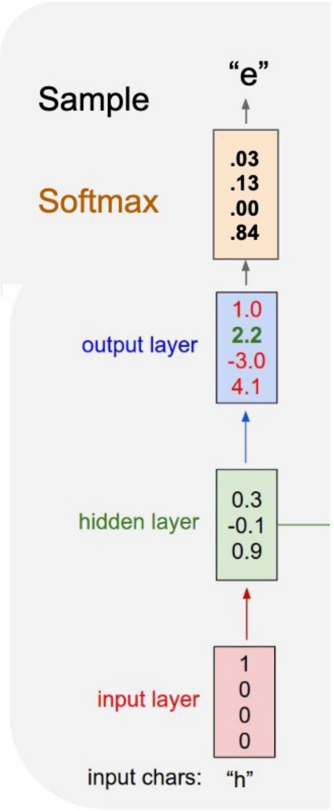


Example: Character-level Language Model

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

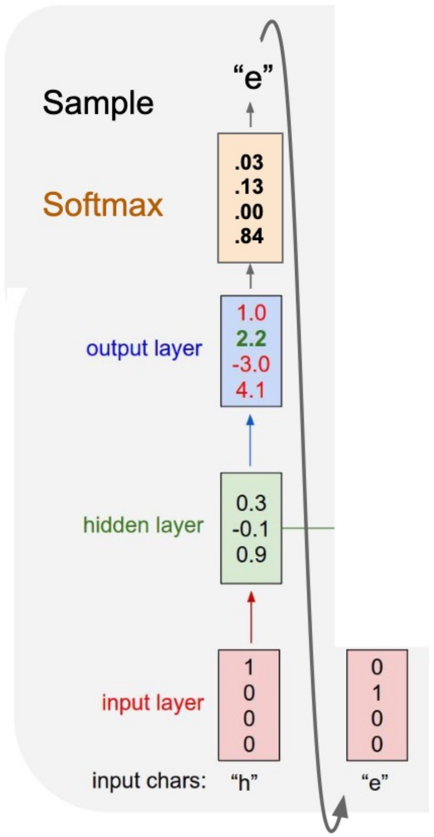


Example: Character-level Language Model

Example:
Character-level
Language Model
Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

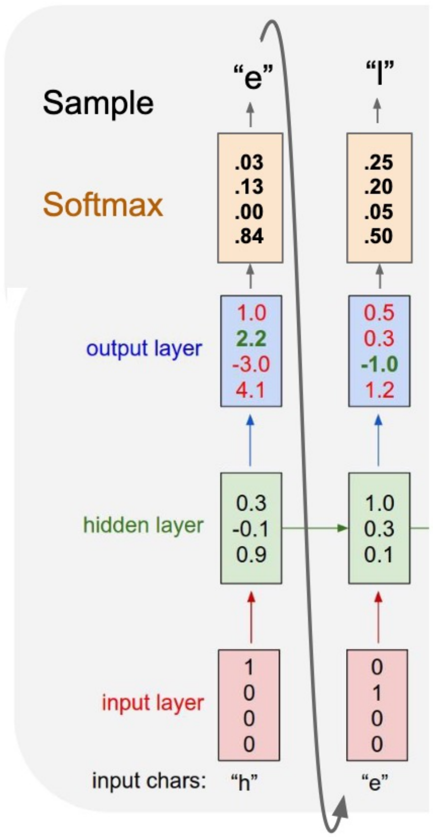


Example: Character-level Language Model

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

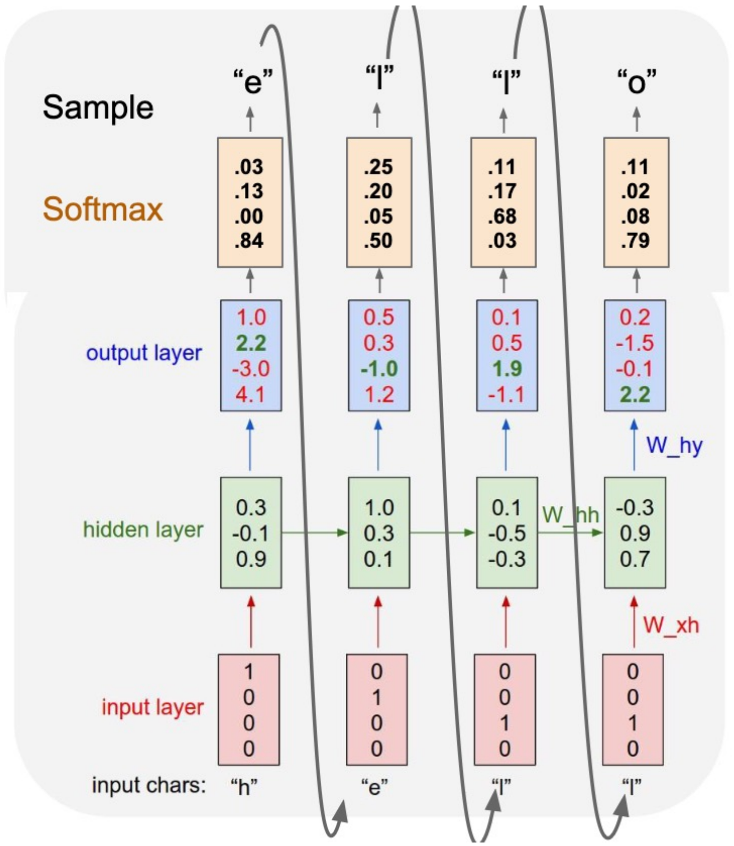


Example: Character-level Language Model

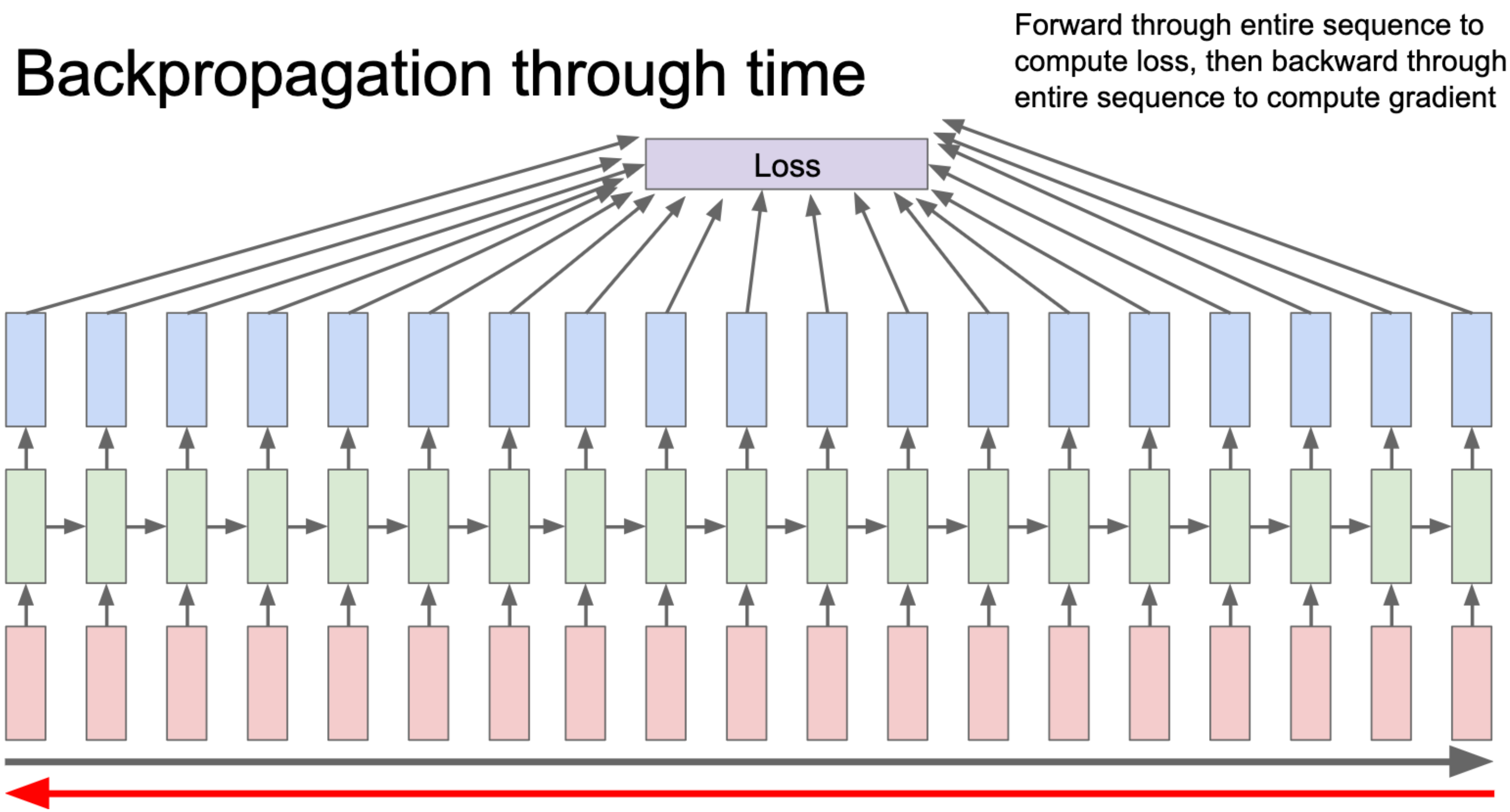
Example:
Character-level
Language Model
Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

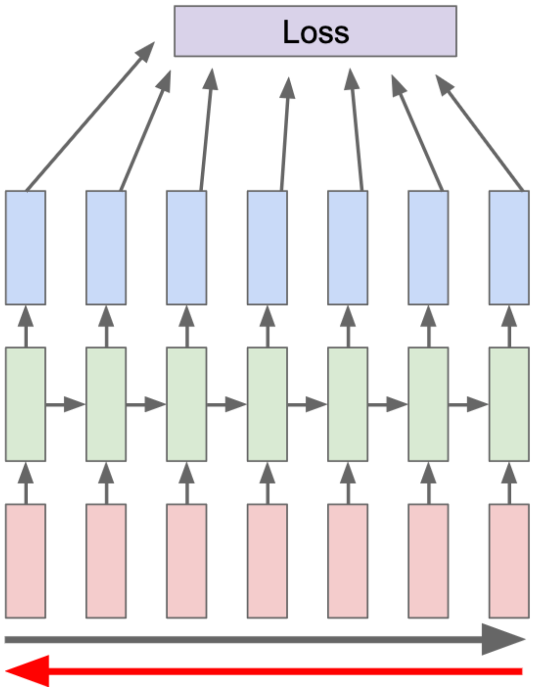


Backpropagation Through Time



Truncated Backpropagation Through Time

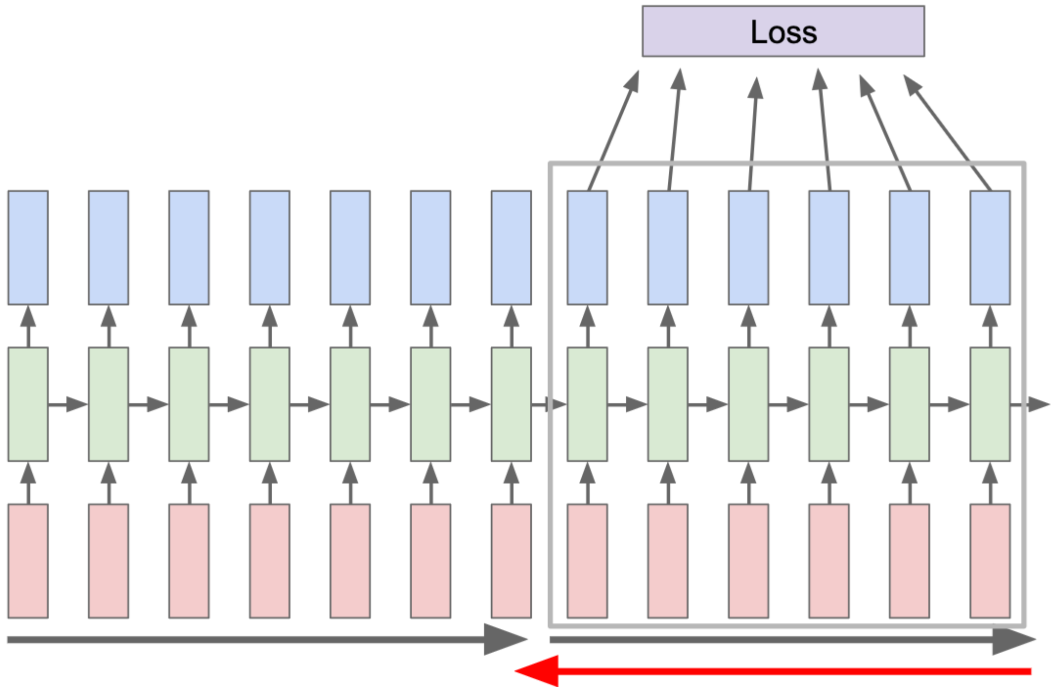
Truncated Backpropagation through time



Run forward and backward through chunks of the sequence instead of whole sequence

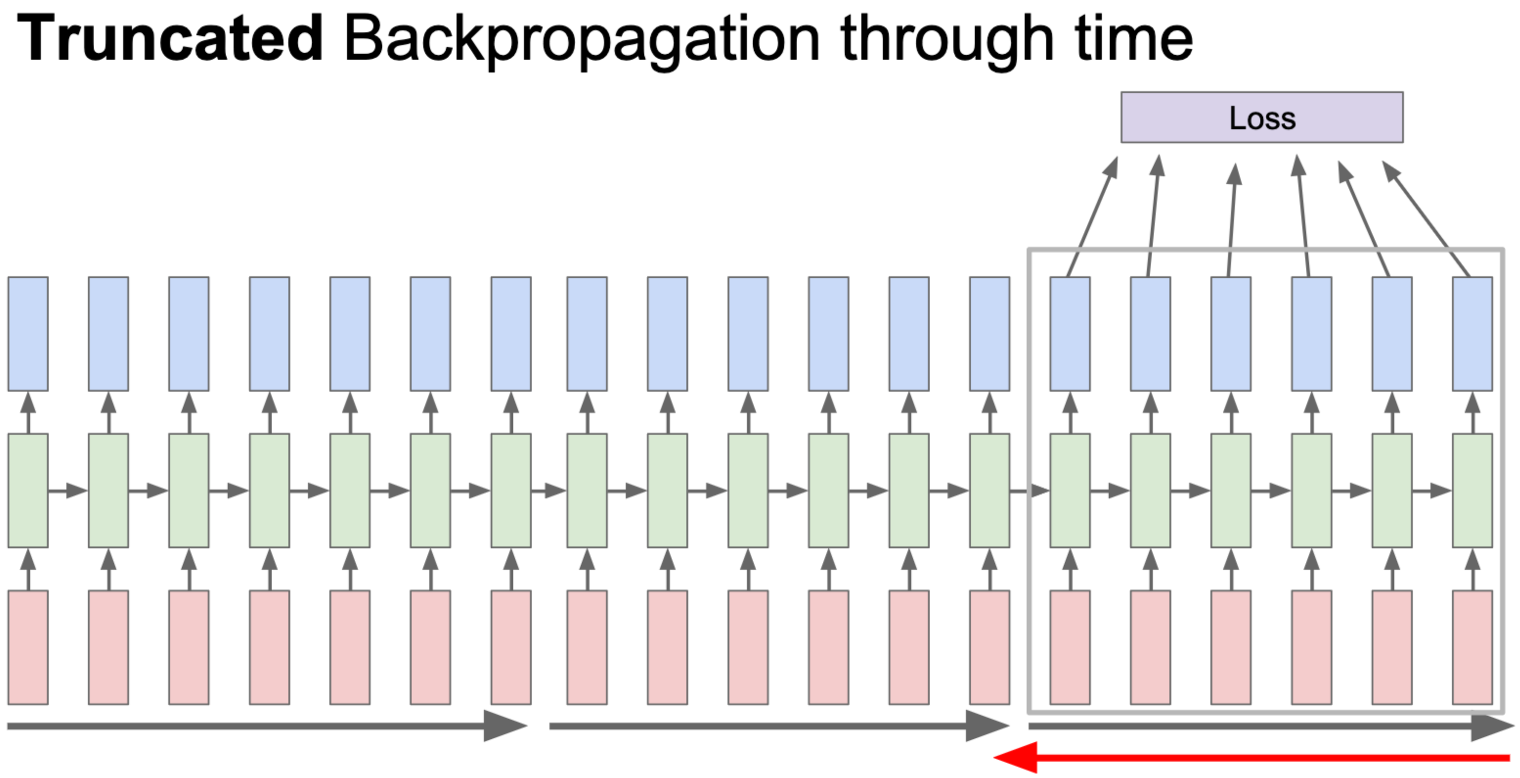
Truncated Backpropagation Through Time

Truncated Backpropagation through time



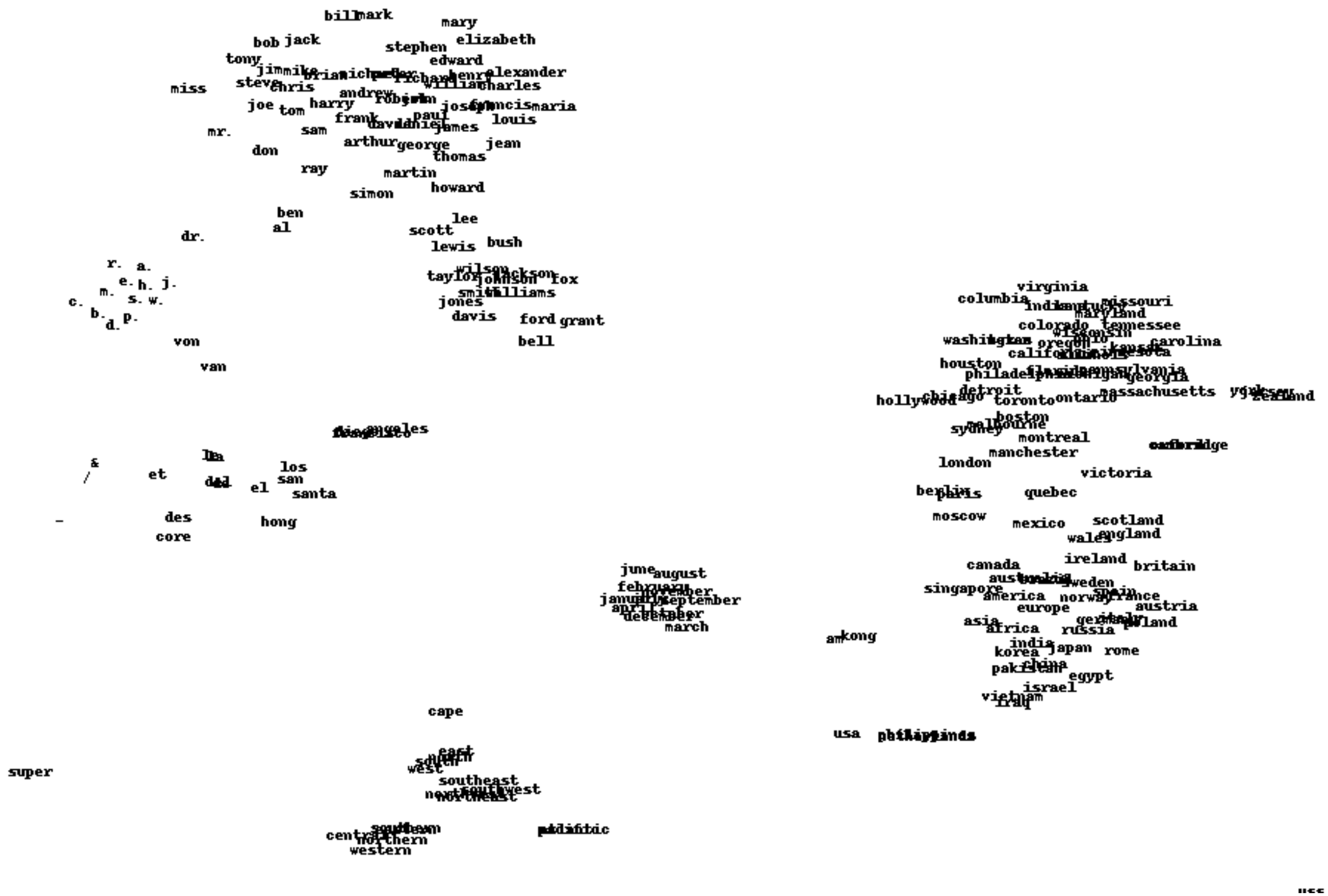
Carry hidden states forward in time forever, but only backpropagate for some smaller number of steps

Truncated Backpropagation Through Time

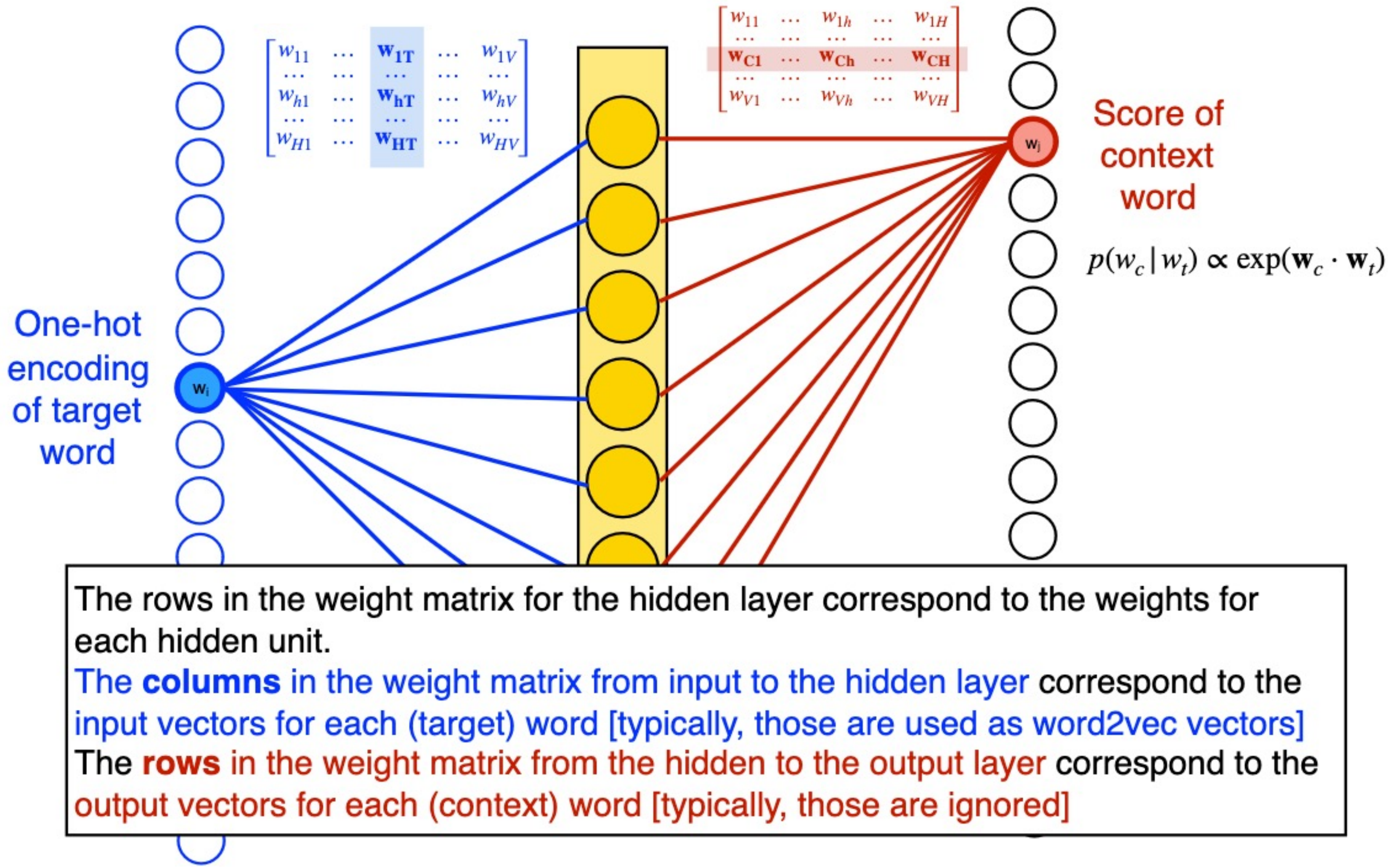


Word Embedding

- Can we represent words as vectors in space?



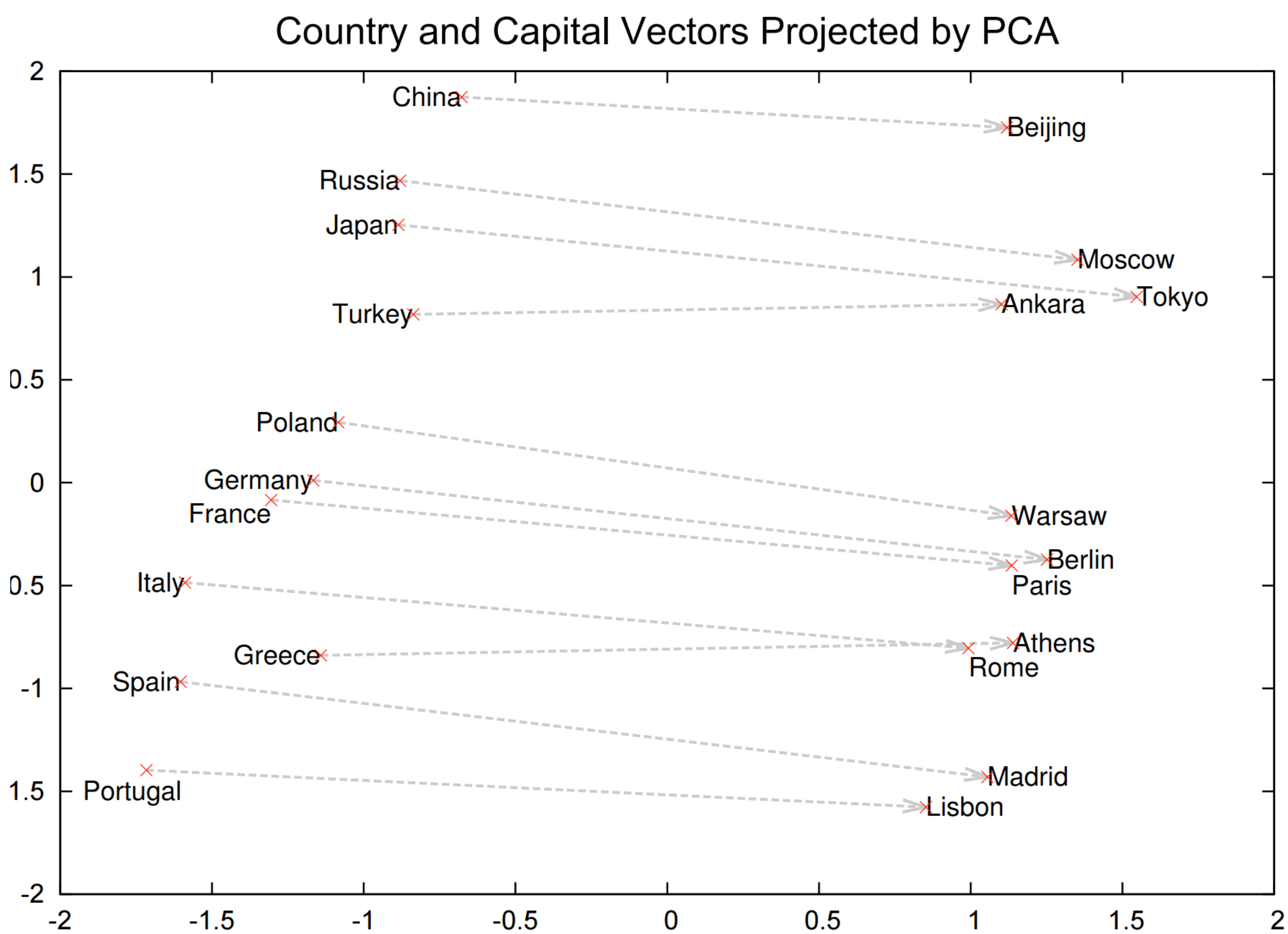
Word Embedding



The rows in the weight matrix for the hidden layer correspond to the weights for each hidden unit.
The **columns** in the weight matrix from input to the hidden layer correspond to the input vectors for each (target) word [typically, those are used as word2vec vectors]
The **rows** in the weight matrix from the hidden to the output layer correspond to the output vectors for each (context) word [typically, those are ignored]

Word Embedding

- Word Analogies:



Long-term Dependencies are Important

… Finally, Tim was planning to visit France on the final week of his journey. He was quite excited to try the local delicacies and had lots of recommendations for good restaurants and exhibitions. His first stop was, of course, the capital where he would meet his long-time Friend Jean-Pierre. In order to arrive for breakfast he took the early 5 AM train from London to …

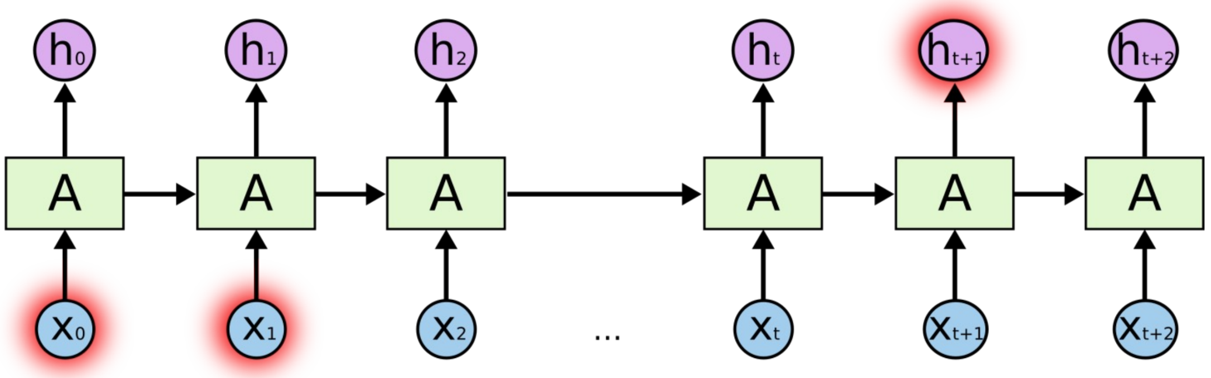
Long-term Dependencies are Important

… Finally, Tim was planning to visit **France** on the final week of his journey. He was quite excited to try the local delicacies and had lots of recommendations for good restaurants and exhibitions. His first stop was, of course, the **capital** where he would meet his long-time Friend Jean-Pierre. In order to arrive for breakfast he took the early 5 AM train from London to …

PARIS!

Long Distance Dependencies

- It is very difficult to train RNNs to retain information over many time steps
- This make is very difficult to learn RNNs that handle long-distance dependencies, such as subject-verb agreement.

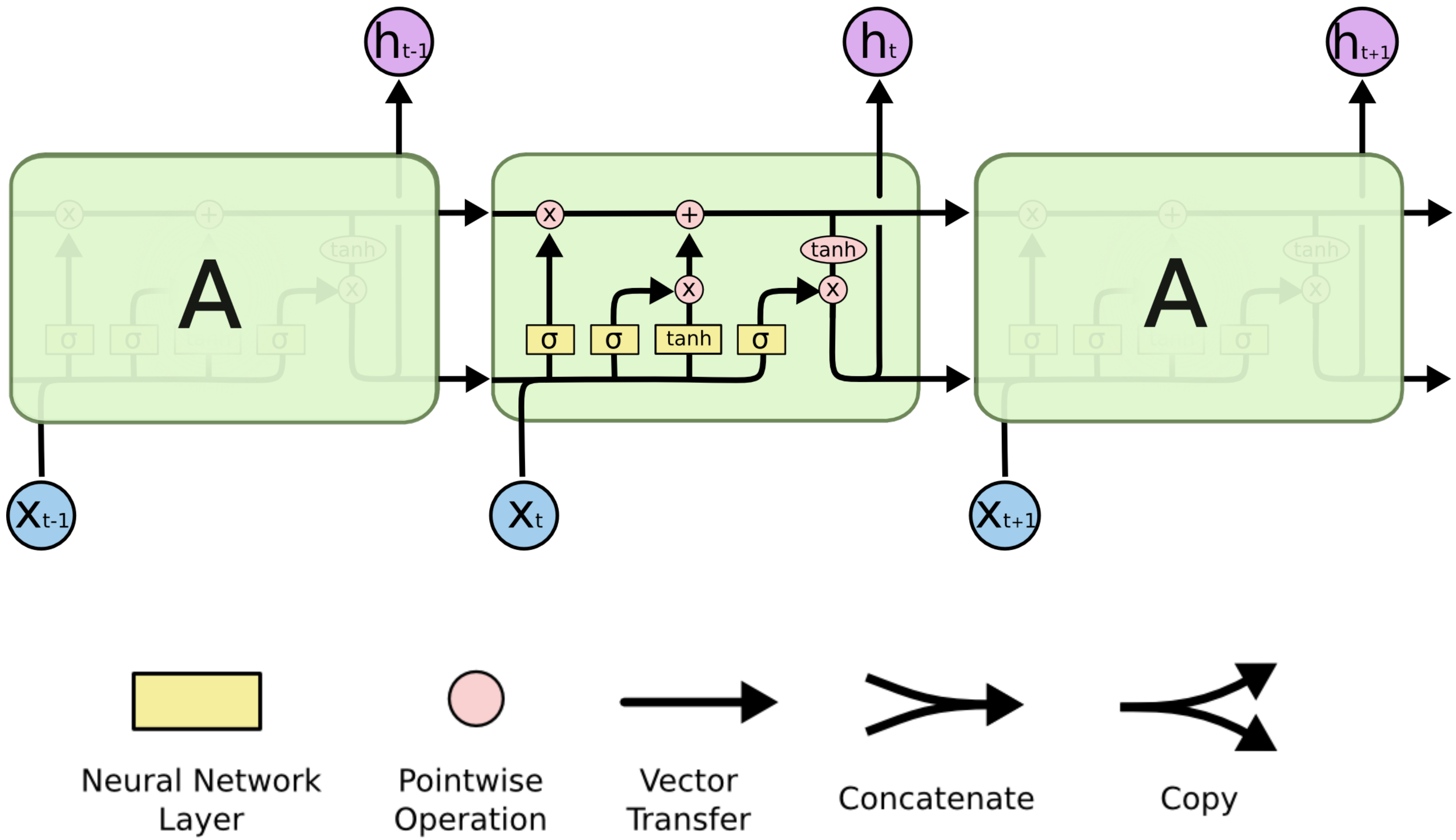


Long Short-Term Memory (LSTM) networks

- LSTM networks, add additional gating units in each memory cell.
  - Forget gate
  - Input gate
  - Output gate

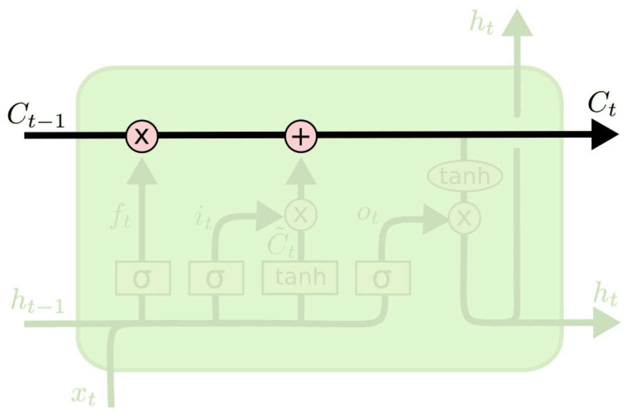
- Prevents vanishing/exploding gradient problem and allows network to retain state information over longer periods of time.

LSTM Network Architecture



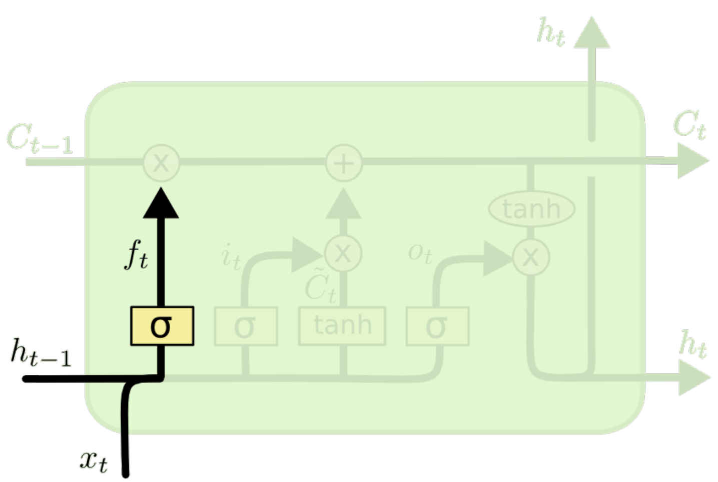
Cell State

- Maintains a vector C_t that is the same dimensionality as the hidden state, h_t
- Information can be added or deleted from this state vector via the forget and input gates.



Forget Gate

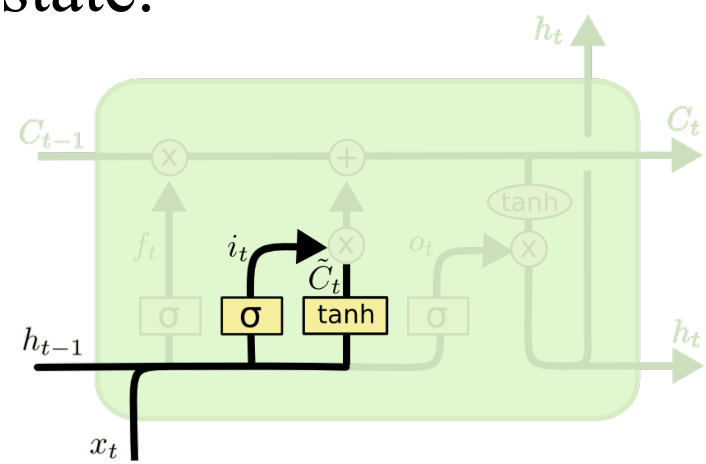
- Forget gate computes a 0-1 value using a logistic sigmoid output function from the input, x_t, and the current hidden state, h_t:
- Multiplicatively combined with cell state, "forgetting" information where the gate outputs something close to 0.



$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \ + \ b_f\right)$$

Input Gate

- First, determine which entries in the cell state to update by computing 0-1 sigmoid output.
- Then determine what amount to add/subtract from these entries by computing a tanh output (valued –1 to 1) function of the input and hidden state.

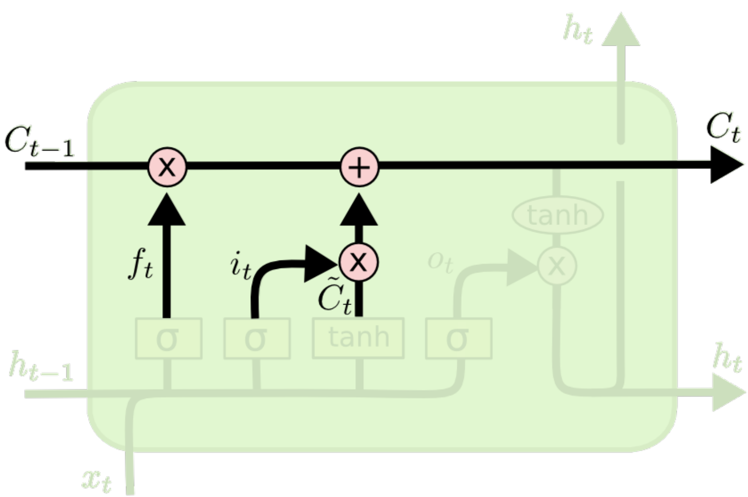


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Updating the Cell State

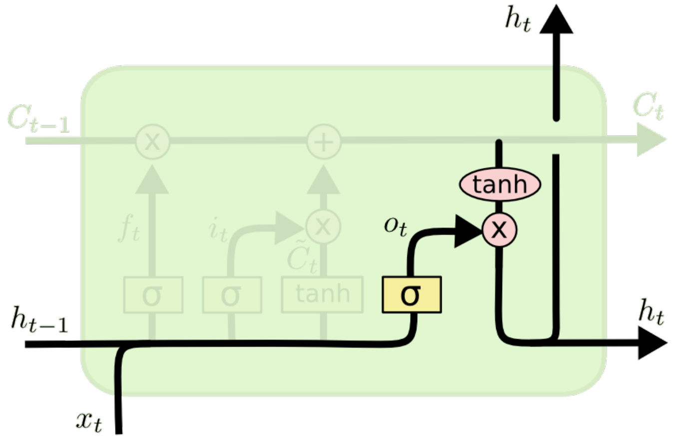
- Cell state is updated by using component-wise vector multiply to "forget" and vector addition to "input" new information.



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Output Gate

- Hidden state is updated based on a "filtered" version of the cell state, scaled to –1 to 1 using tanh.
- Output gate computes a sigmoid function of the input and current hidden state to determine which elements of the cell state to "output".

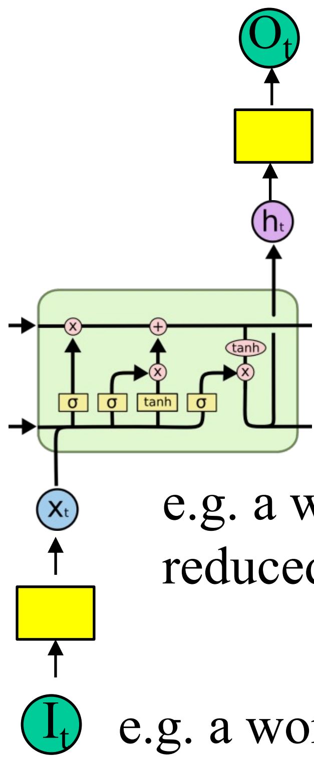


$$o_t = \sigma\left(W_o \left[h_{t-1}, x_t\right] + b_o\right)$$

$$h_t = o_t * \tanh\left(C_t\right)$$

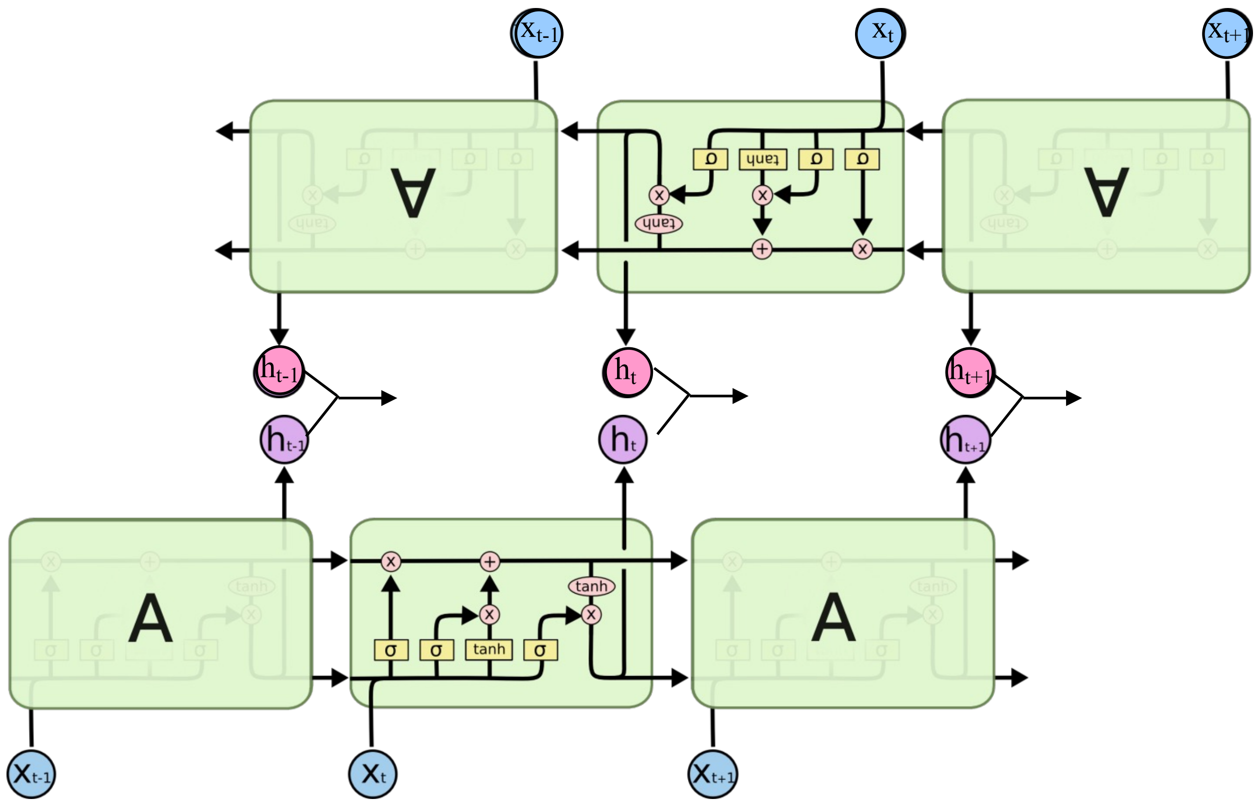
Overall Network Archtecture

- Single or multilayer networks can compute LSTM inputs from problem inputs and problem outputs from LSTM outputs.



Bi-directional LSTM (Bi-LSTM)

- Separate LSTMs process sequence forward and backward and hidden layers at each time step are concatenated to form the cell output.



Multilayer RNNs/LSTMs

Multilayer RNNs

$$\boxed{h_t^l = \tanh W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}}$$

$h \in \mathbb{R}^n$ $\quad W^l \; [n \times 2n]$

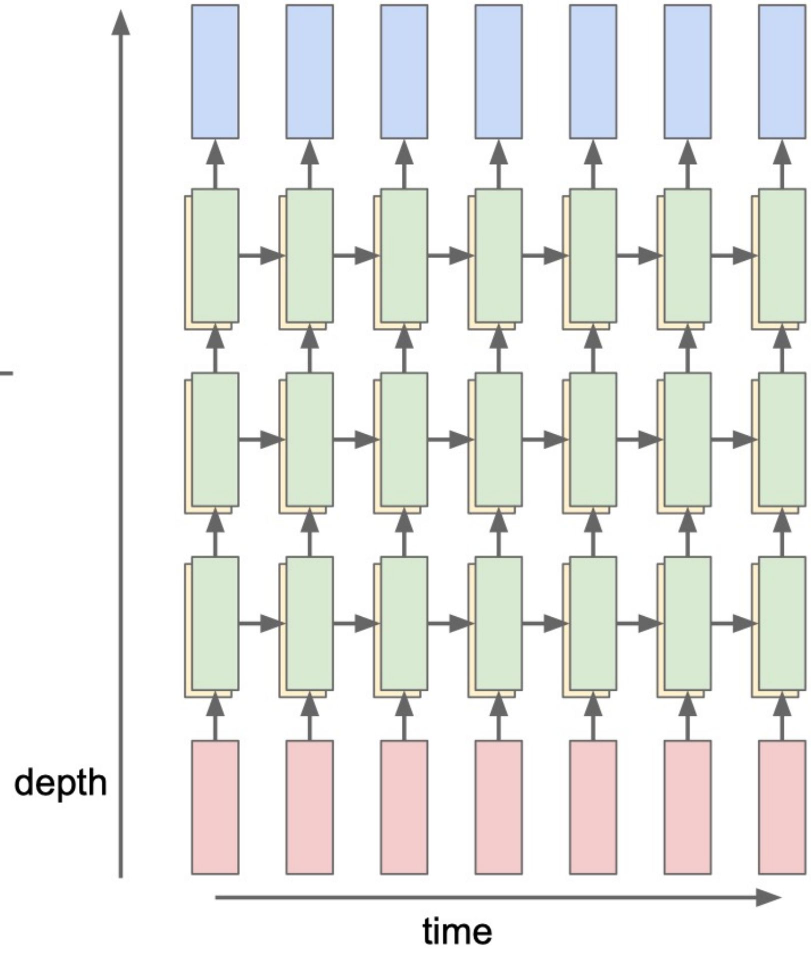
LSTM:

$W^l \; [4n \times 2n]$

$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \text{sigm} \\ \text{sigm} \\ \text{sigm} \\ \tanh \end{pmatrix} W^l \begin{pmatrix} h_t^{l-1} \\ h_{t-1}^l \end{pmatrix}$$

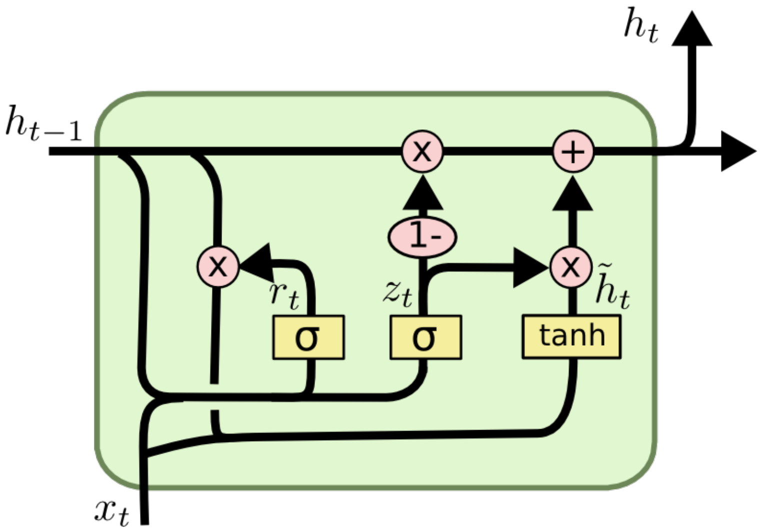
$$c_t^l = f \odot c_{t-1}^l + i \odot g$$

$$h_t^l = o \odot \tanh(c_t^l)$$



Gated Recurrent Unit (GRU)

- Alternative RNN to LSTM that uses fewer gates (Cho, et al., 2014)
  - Combines forget and input gates into "update" gate.
  - Eliminates cell state vector



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

GRU vs. LSTM

- GRU has significantly fewer parameters and trains faster.
- Experimental results comparing the two are still inconclusive, many problems they perform the same, but each has problems on which they work better.

Conclusions of LSTM

- By adding "gates" to an RNN, we can prevent the vanishing/exploding gradient problem.
- Trained LSTMs/GRUs can retain state information longer and handle long-distance dependencies.

Questions?

UNIVERSITY OF NORTH CAROLINA
CHARLOTTE