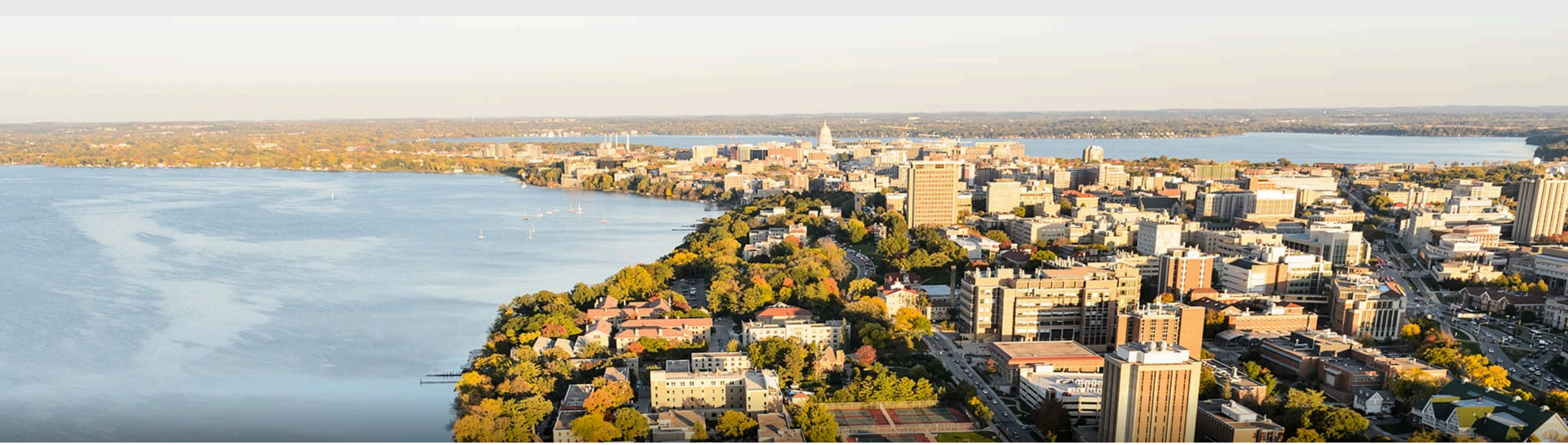


CS 760: Machine Learning
Recurrent Neural Networks

Josiah Hanna

University of Wisconsin-Madison

October 19, 2023

Announcements

- **Please submit midterm evaluations**
  - Bonus points for everyone on the midterm based on different percentages for participation.
  - >50% = 2 pts; > 75% = 2 more pts; > 90% = 6 pts; 100% = 10pts

- **Homework 4** due on Tuesday, October 31 now.

- Congrats on finishing midterm. Halfway!

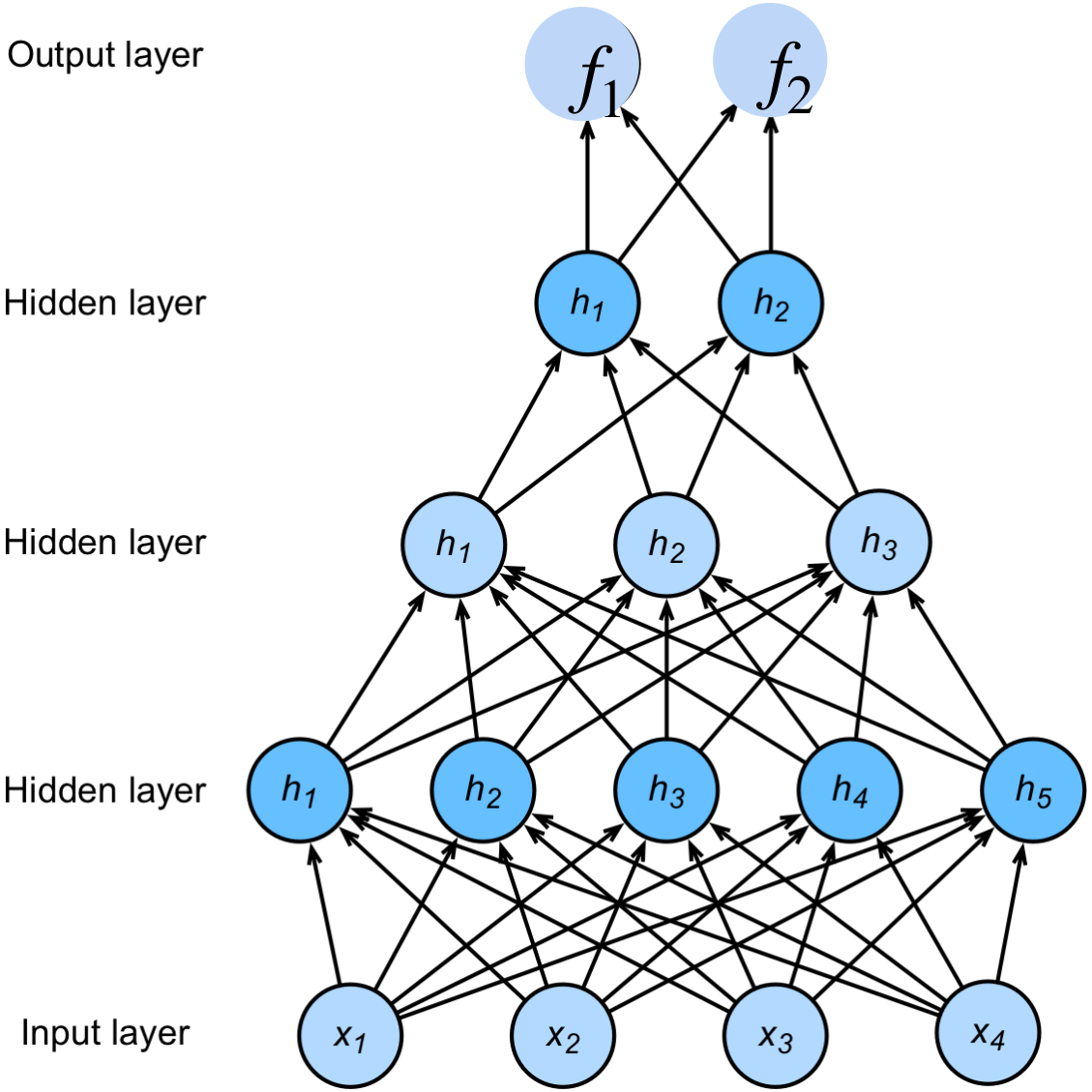
Outline

- **RNN Basics**
  - Sequential tasks, hidden state, vanilla RNN
- **RNN Variants + LSTMs**
  - RNN training, variants, LSTM cells

Outline

- **RNN Basics**
  - Sequential tasks, hidden state, vanilla RNN
- **RNN Variants + LSTMs**
  - RNN training, variants, LSTM cells

Neural Networks: Multi-layer perceptrons



$$\mathbf{h}_1 = \sigma(\mathbf{W}_1\mathbf{x} + \mathbf{b}_1)$$

$$\mathbf{h}_2 = \sigma(\mathbf{W}_2\mathbf{h}_1 + \mathbf{b}_2)$$

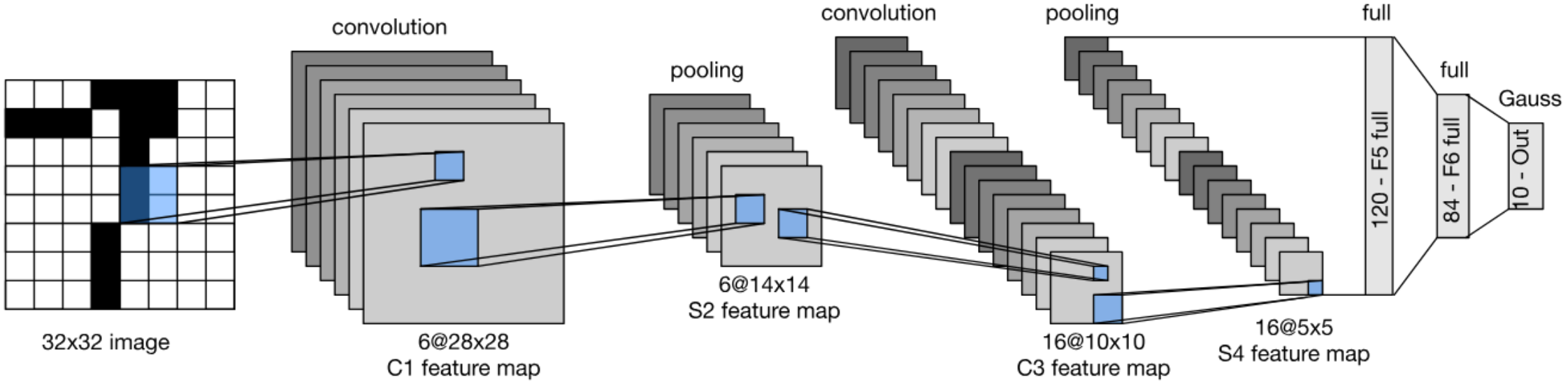
$$\mathbf{h}_3 = \sigma(\mathbf{W}_3\mathbf{h}_2 + \mathbf{b}_3)$$

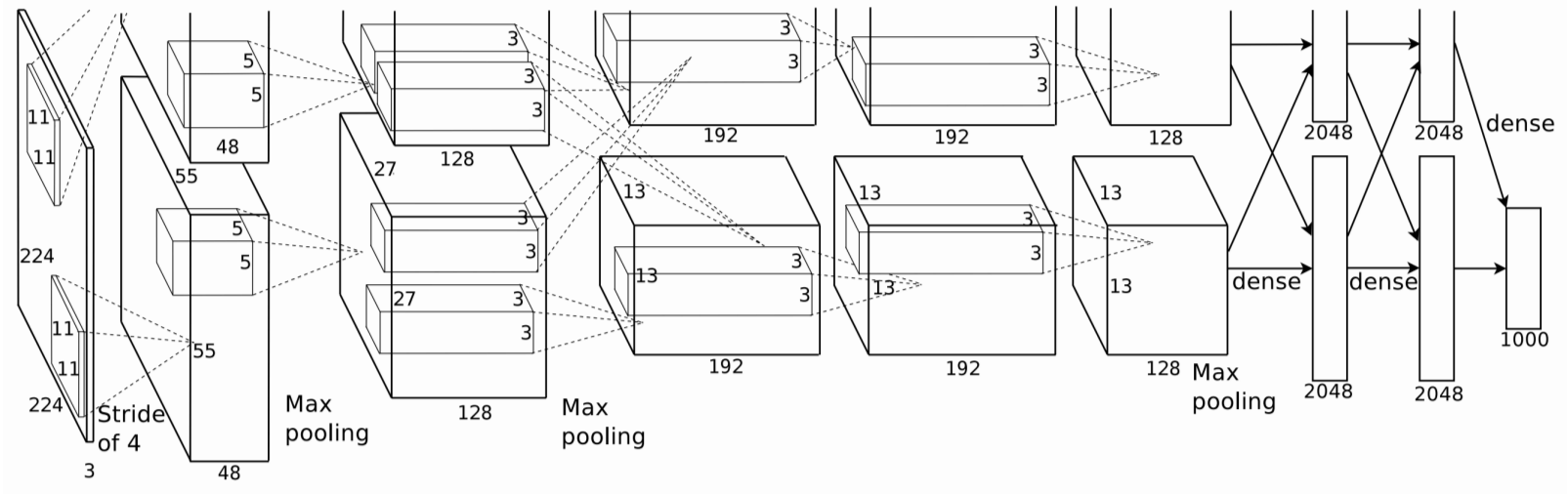
$$\mathbf{f} = \mathbf{W}_4\mathbf{h}_3 + \mathbf{b}_4$$

$$\mathbf{y} = \text{softmax}(\mathbf{f})$$

NNs are composition of nonlinear functions

Neural Networks: Convolutional Neural Networks



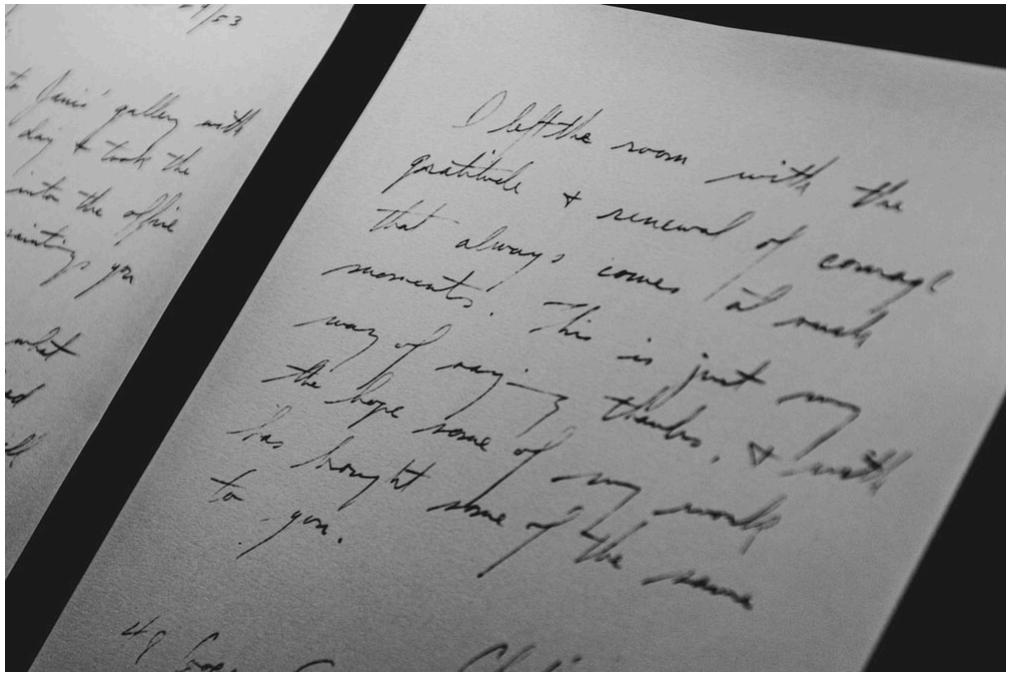


MLPs and CNNs

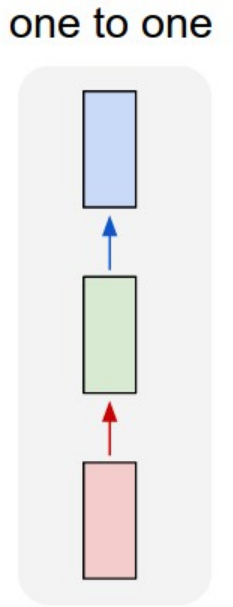
- Our models map **one input** object to **one output** object
  - Fixed-dimensional input vector

- What about sequential data?
  - I.e., language!
  - Also, video, many other data
  - Memory

- What should our models do?

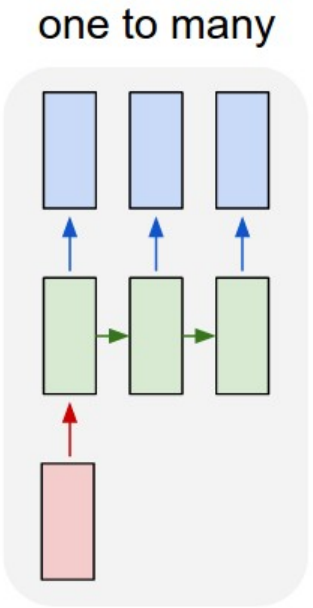


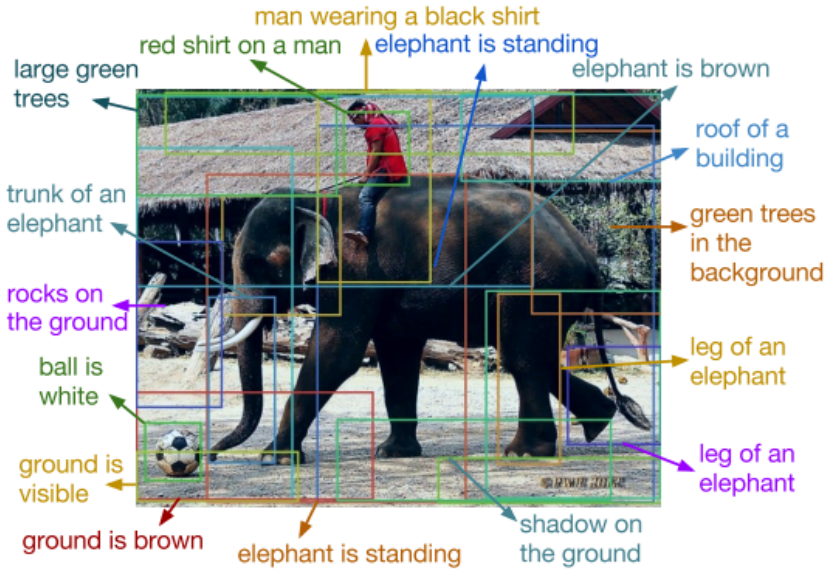
Inputs and Outputs in NNs



- Our standard model so far. One fixed input type, one output
  - Image classification

Inputs and Outputs in NNs

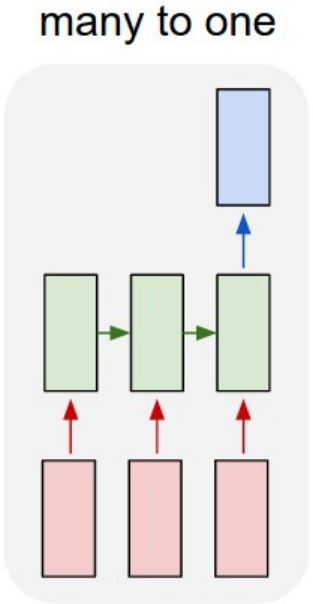


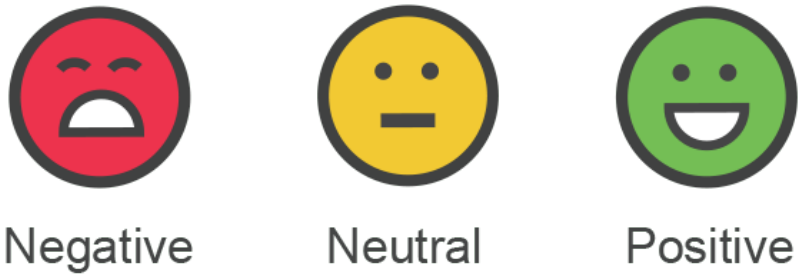


"DenseCap: Fully Convolutional Localization Networks for Dense Captioning", Johnson, Karpathy, Li

- One input, but sequence at the output
  - **Ex**: image captioning. Input: one image, Output: sequence of words

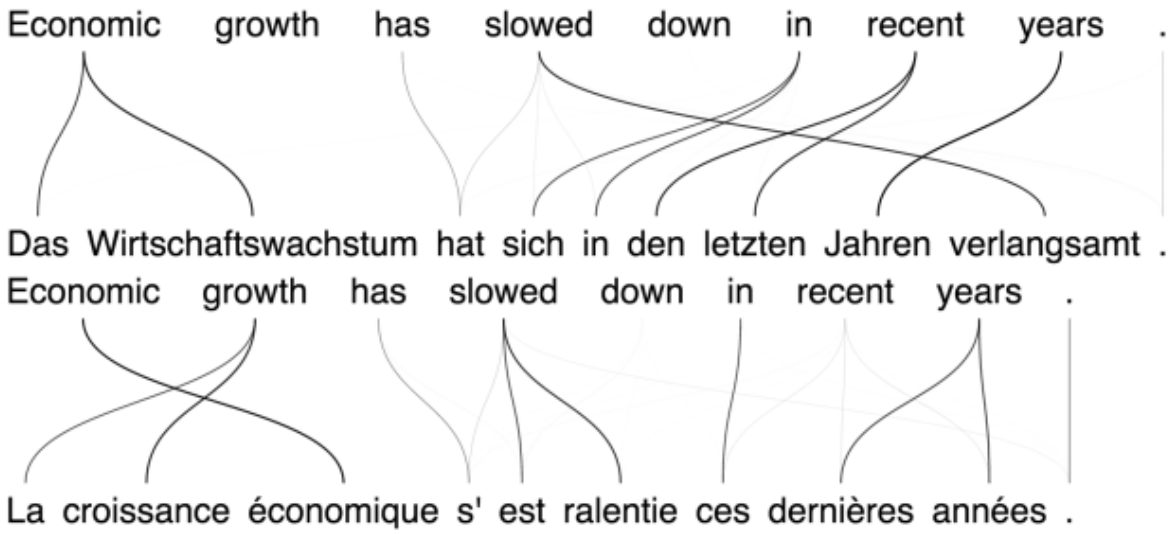
Inputs and Outputs in NNs



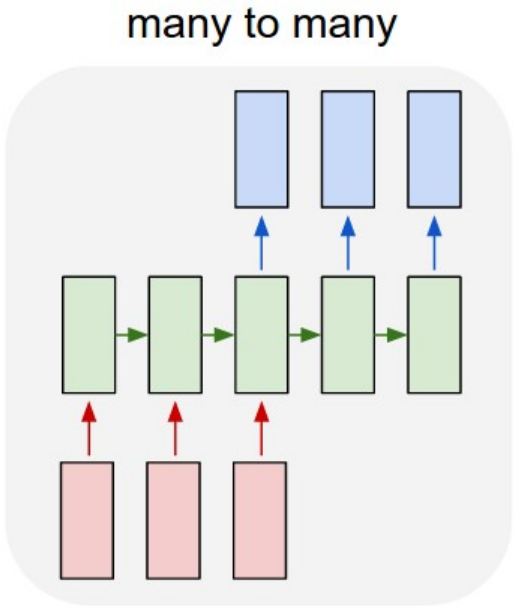


- Sequence input, one output
  - **E.g 1**: sentiment analysis. Input is a sentence, output is one of {positive, neutral, negative}
  - **E.g 2:** text-to-image generation

Inputs and Outputs in NNs

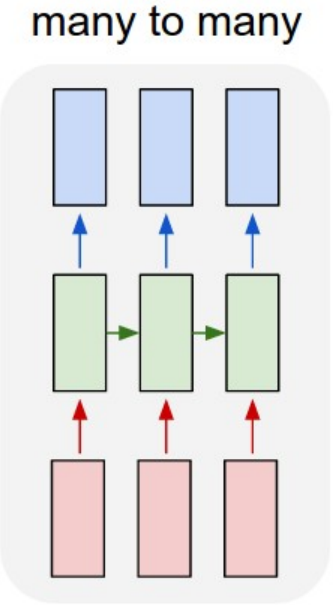


devblogs.nvidia.com



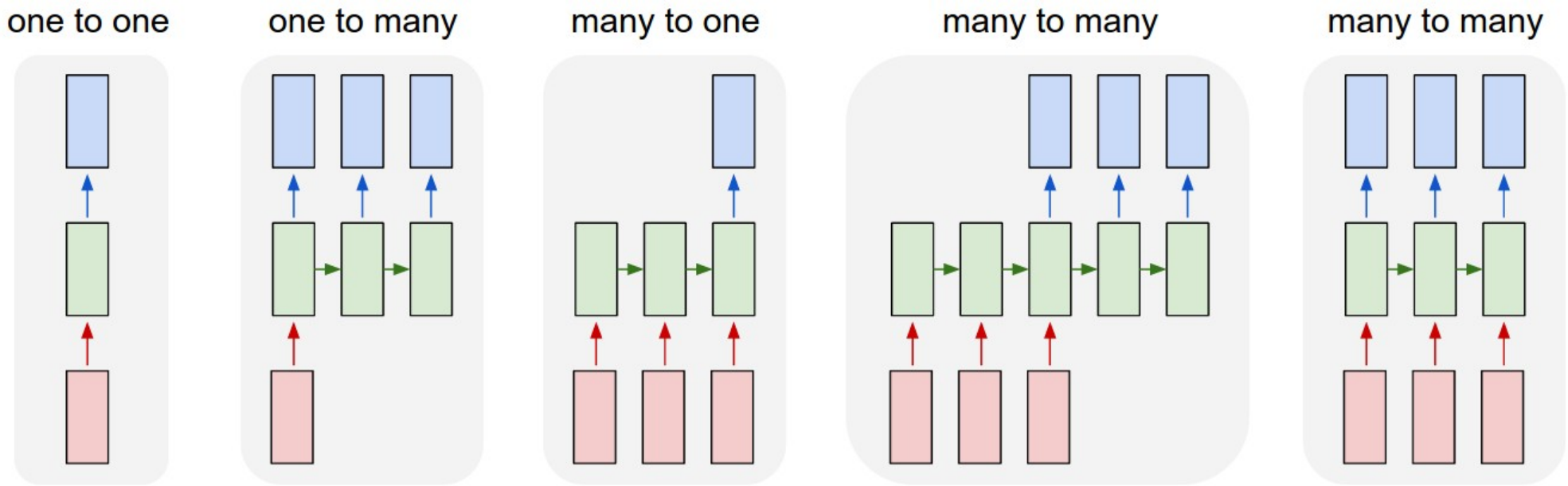
- Sequence input, sequence output
  - **Ex:** machine translation. Translate from language A to language B

Inputs and Outputs in NNs



- Synchronized input and output
  - **Ex:** Video classification: label each frame of a video

Tasks We Can Handle?

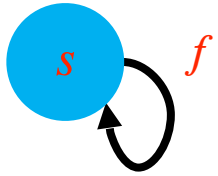


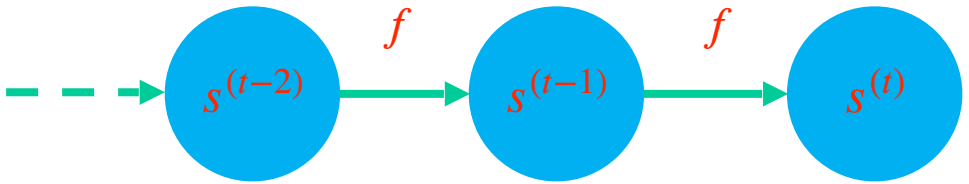
- We have only learned to do one-to-one so far…

Modeling Sequential Data

- Simplistic model of a dynamical system:
  - $s^{(t)}$ state at time t.
  - Transition function f

$$s^{(t+1)} = f(s^{(t)}; \theta)$$

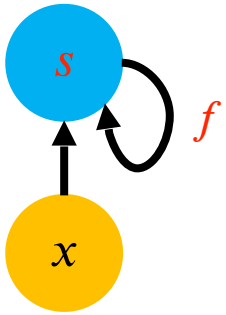


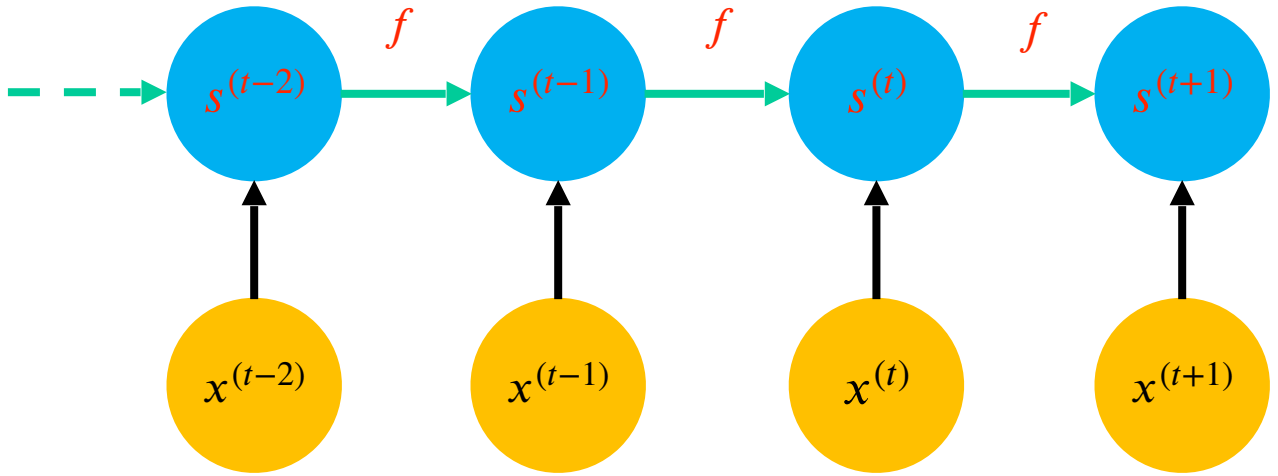


Modeling Sequential Data: External Input

- External inputs can also influence transitions
  - $s^{(t)}$ state at time t. Transition function f
  - $x^{(t)}$: input at time t

$$s^{(t+1)} = f(s^{(t)}, x^{(t+1)}; \theta)$$



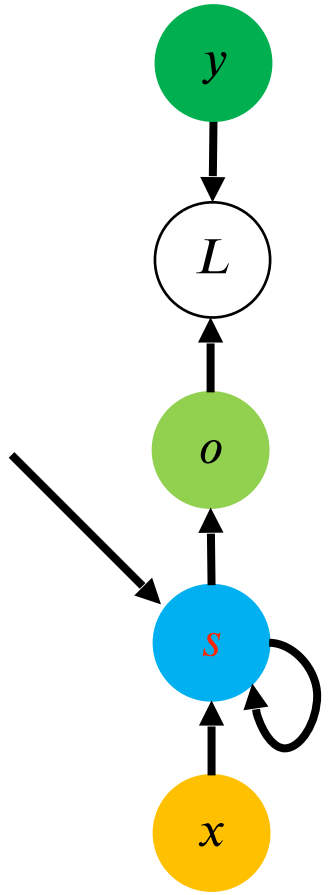


Important: the same f and θ for all time steps

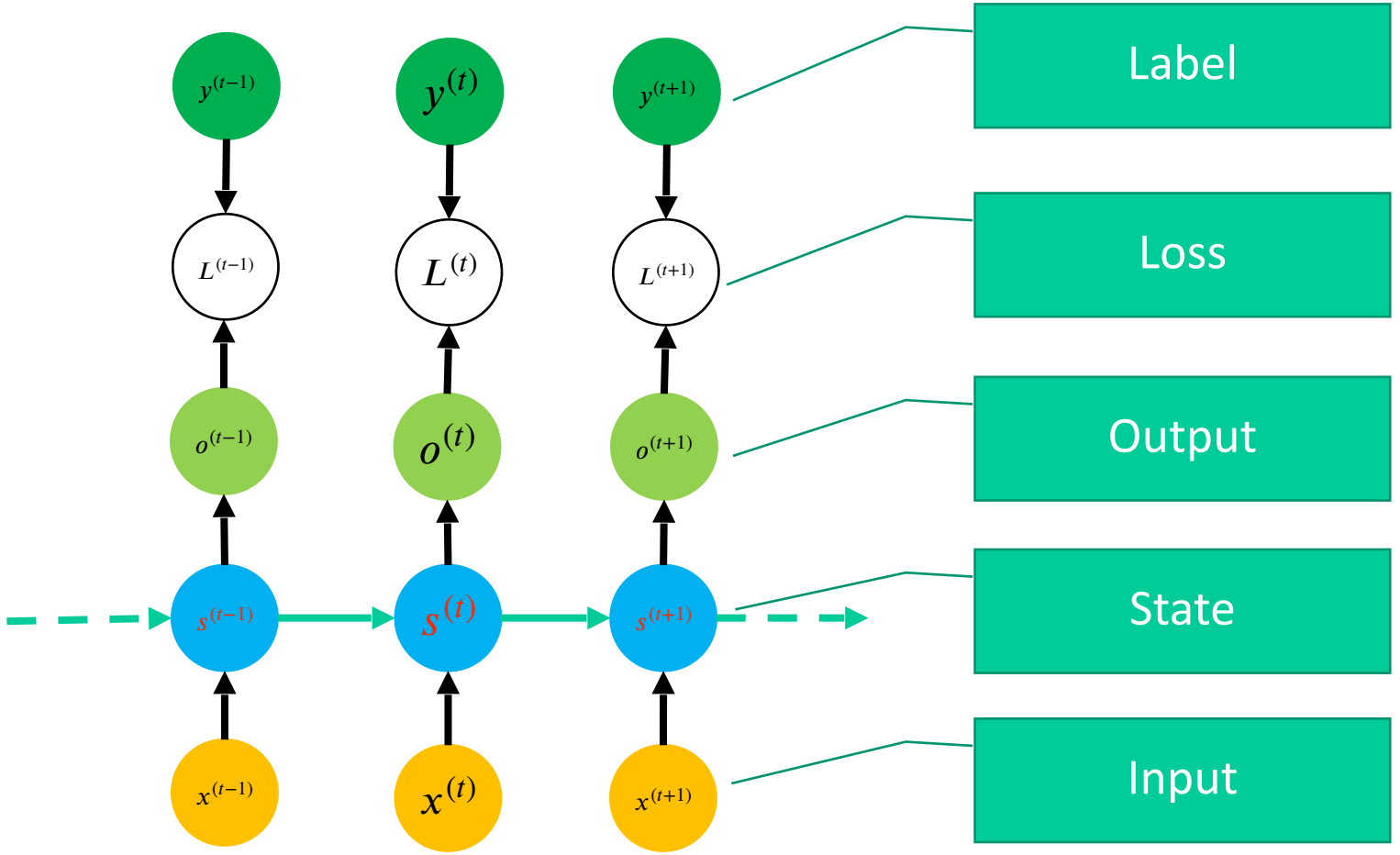
RNNs: Basic Components

- **Necessary components**
  - State s
- **Optional components at each time-step**
  - Input x
  - Output o
- **Optional components for training**
  - Label y
  - Loss function L

Recurrent: state is plugged back into itself



RNNs: Unrolled Graph



Recurrent Neural Networks, generally

- Use the principle from the example above:
  - Same computational function and parameters across different time steps of the sequence
- Each time step: takes the input entry and the previous state to compute the current state.
  - **State**: a (lossy) summary of the past
  - Shared functions / parameters across time steps
    - Reduces the capacity, helps with **generalization**
  - Uses the knowledge that sequential data can be processed in the same way at different time step

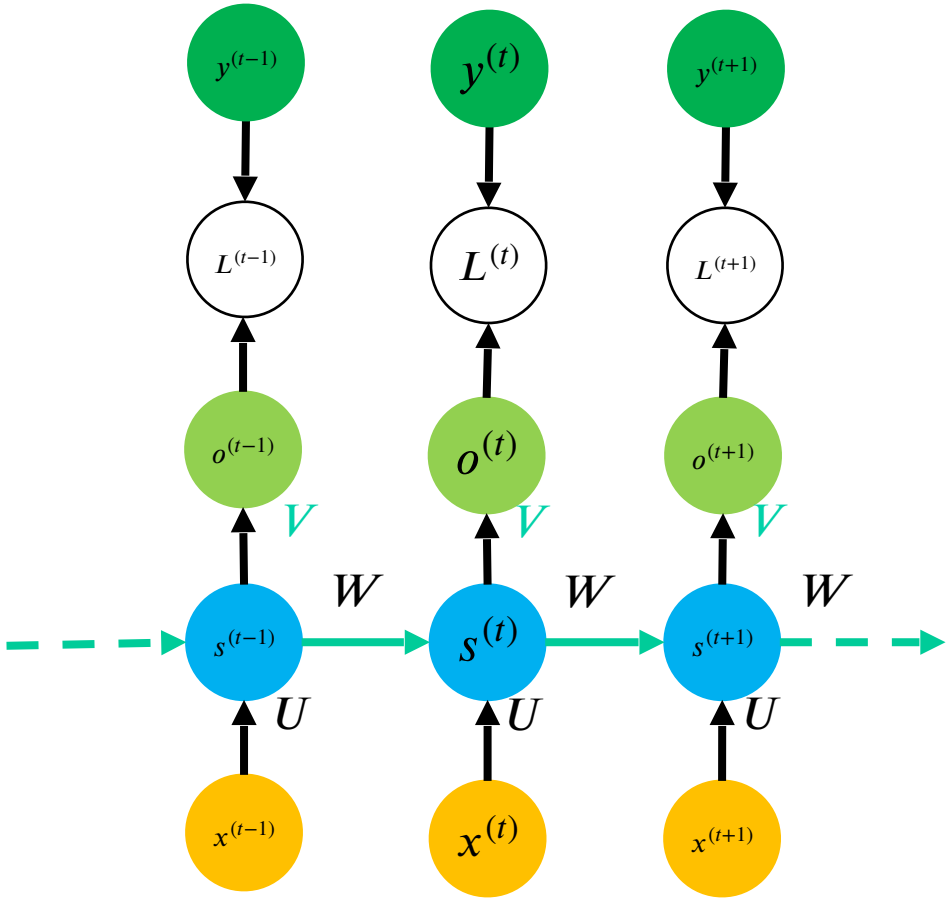
Similar logic as for parameter sharing with CNNs

Recurrent Neural Networks, generally

- **Training:**
  - Loss and outputs: depends on the application.
    - E.g. Computing at every time step (state)
  - And then back propagation.

Simple RNNs

- Classical RNN variant:



$$a^{(t)} = b + Ws^{(t-1)} + Ux^{(t)}$$
$$s^{(t)} = \tanh\left(a^{(t)}\right)$$
$$o^{(t)} = c + Vs^{(t)}$$
$$\hat{y}^{(t)} = \mathrm{softmax}\left(o^{(t)}\right)$$
$$L^{(t)} = \mathrm{CrossEntropy}(y^{(t)}, \hat{y}^{(t)})$$

$a^{(t)}$ is a linear combination of previous state and input at round t.

$s^{(t)}$ is a nonlinear activation of $a^{(t)}$.

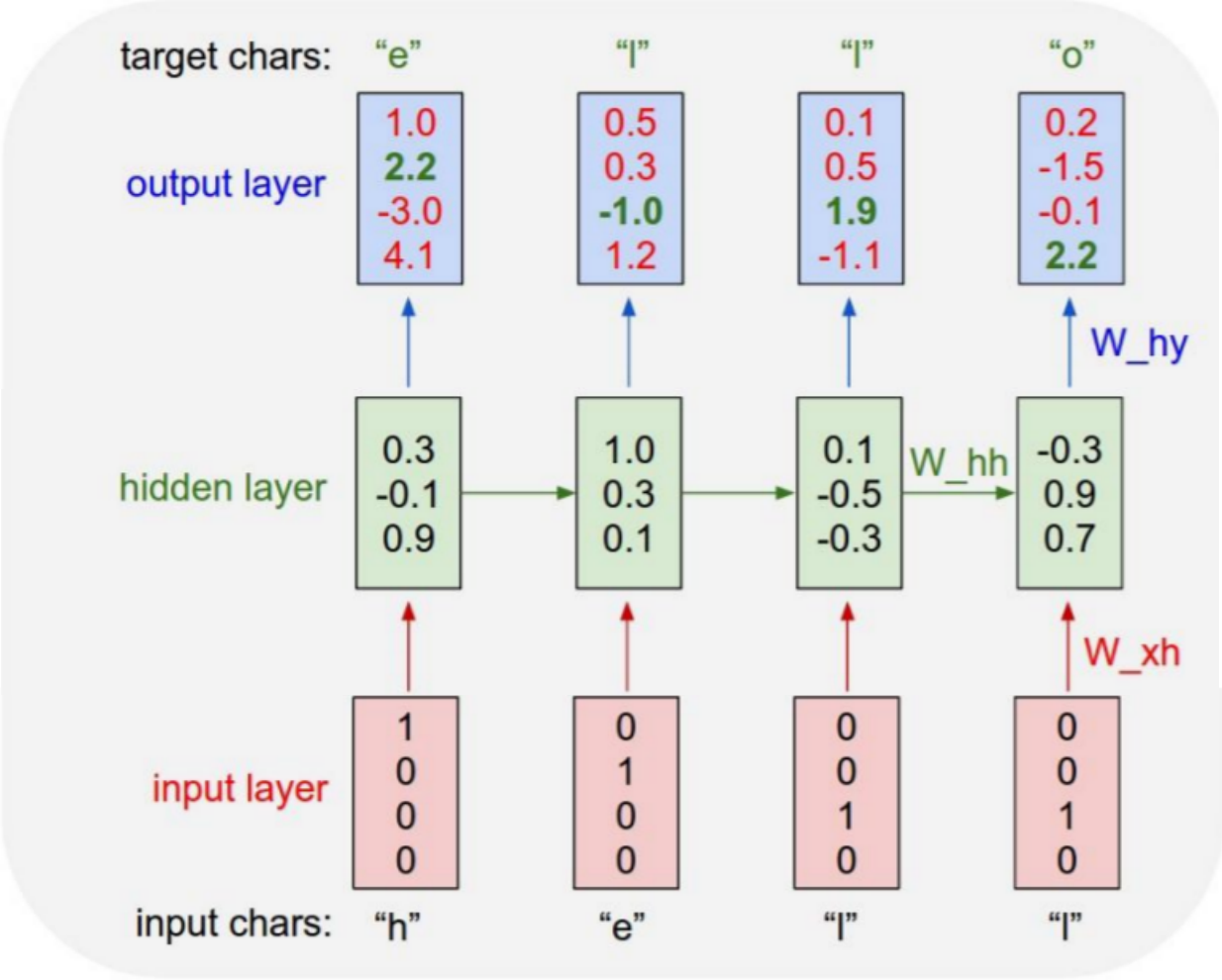
Example: Character Level Language Model

- Goal of language model: predict next character:

- Vocabulary
  {h,e,l,o}

- **Training** sequence:

"hello"

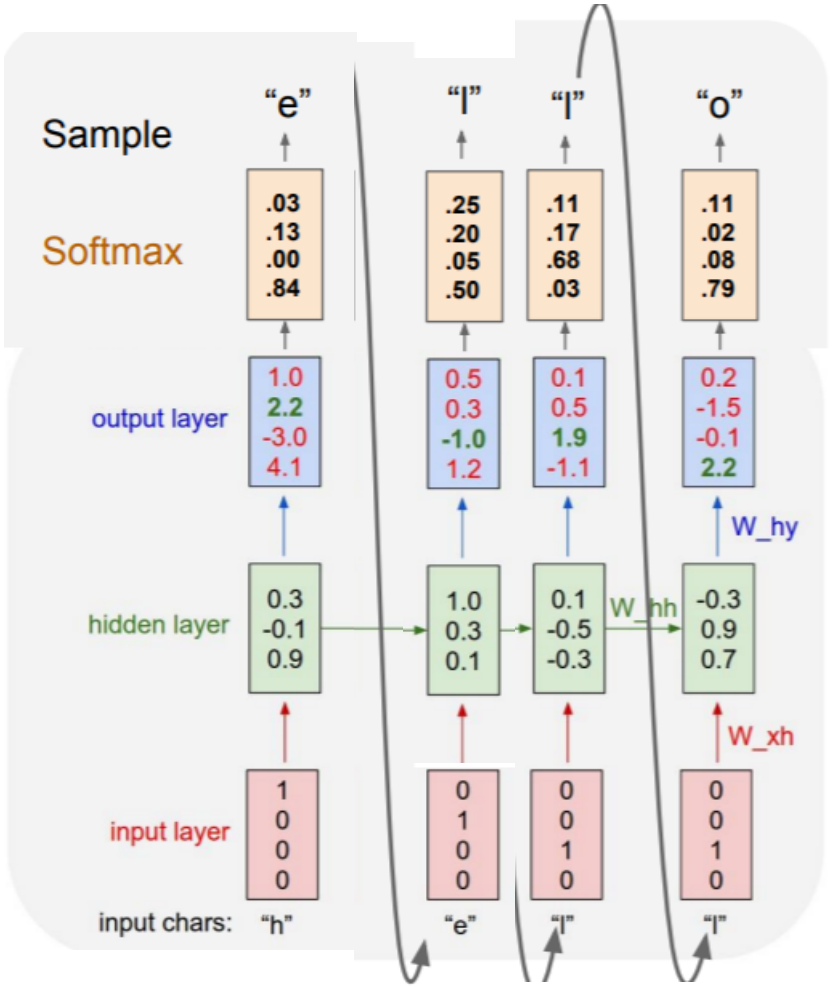


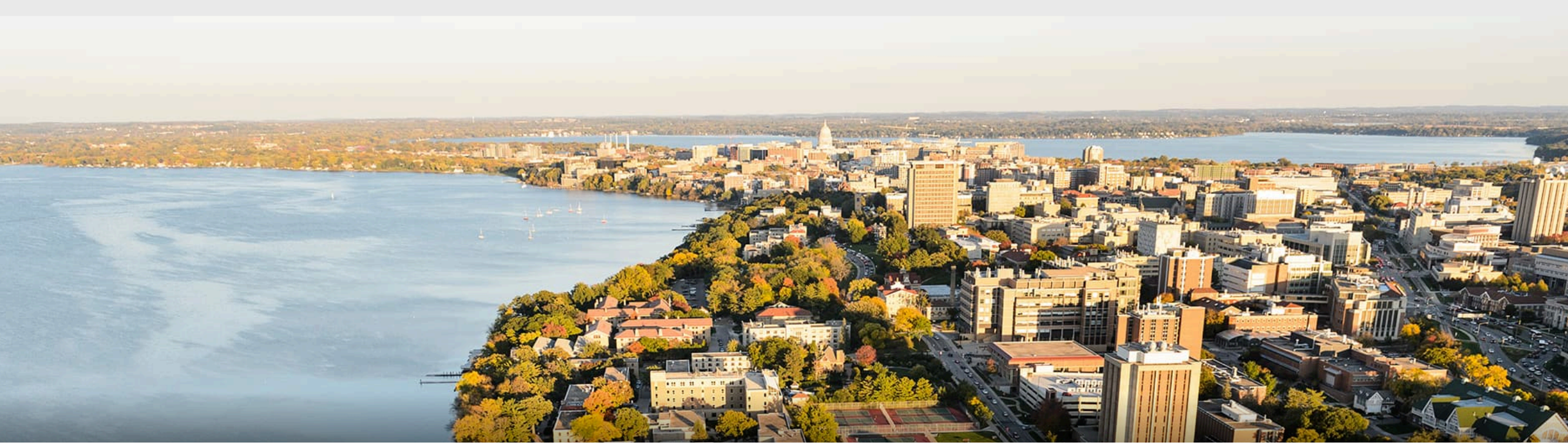
Example: Character Level Language Model

- Goal of language model: predict next character:

- Vocabulary
  {h,e,l,o}

- **Test** time:
  - Sample chars, feed into model





Break & Quiz

Quiz: Are these statements true or false?

(A) Order matters in sequential data.

(B) A batch of sequential data always contains sequences of the same length.

(C) In an RNN, the hidden state $s^{(t)}$ is the linear combination of the previous hidden state $s^{(t-1)}$ and the external data $x^{(t)}$.

(A) **True.** As is shown by its name "sequential", order matters in sequential data.

(B) **False:** A batch of sequential data can have different length, such as different sentences.

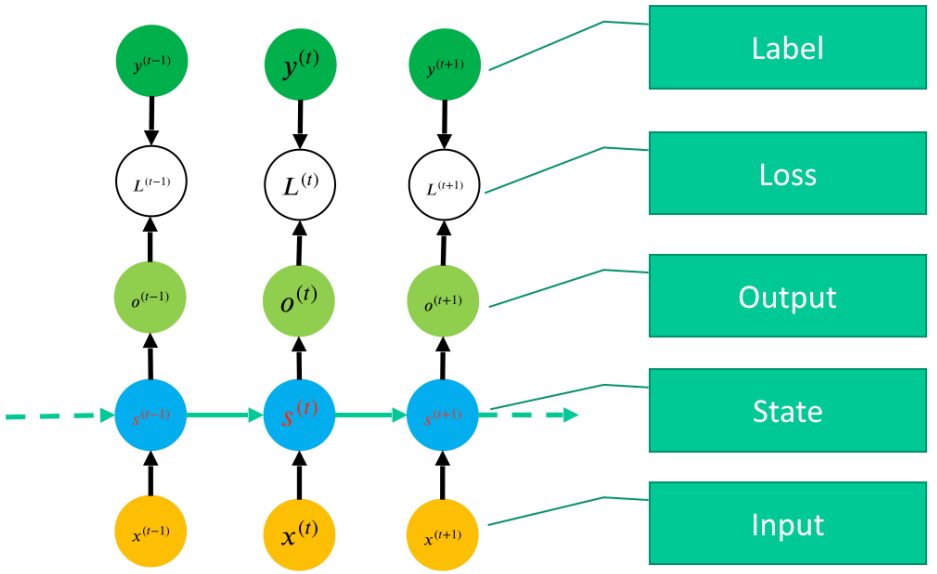
(C) **False:** We need to use an activation function to compute the hidden states, so it's not linear.

Quiz: Due to their sequential nature, RNNs have a Markovian property. Which statement below best describes this property.

1. The distribution of the current output is independent of past input characters, conditioned on the current input character.

2. The distribution of the current output is independent of past input characters, conditioned on the current input character and current state.

Ans: 2

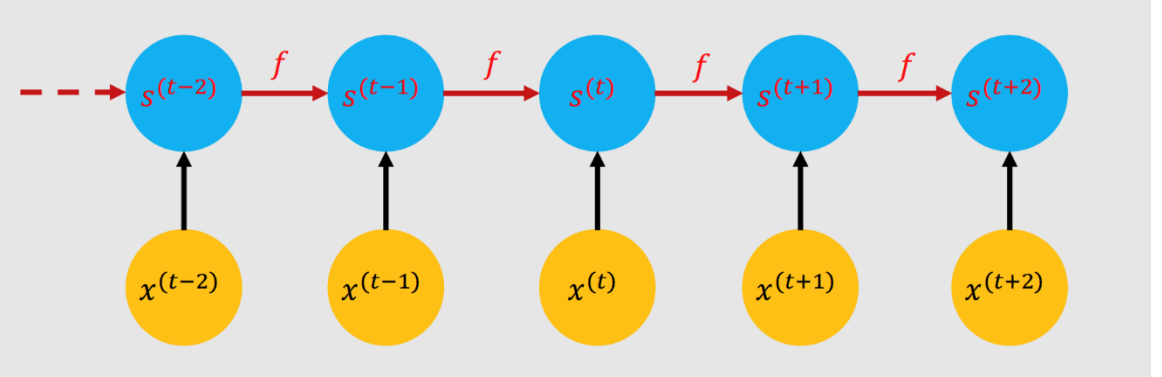


Quiz: Please choose the representation of $s^{(t+2)}$ in terms of

$s^{(t)}$, $x^{(t)}$, $x^{(t+1)}$, $x^{(t+2)}$ in the following dynamical system

$$s^{(t+1)} = f_\theta(s^{(t)}, x^{(t+1)}).$$

1. $f_\theta(s^{(t)}, x^{(t+1)})$
2. $f_\theta(s^{(t)}, x^{(t+2)})$
3. $f_\theta(f_\theta(s^{(t)}, x^{(t)}), x^{(t+1)})$
4. $f_\theta(f_\theta(s^{(t)}, x^{(t+1)}), x^{(t+2)})$

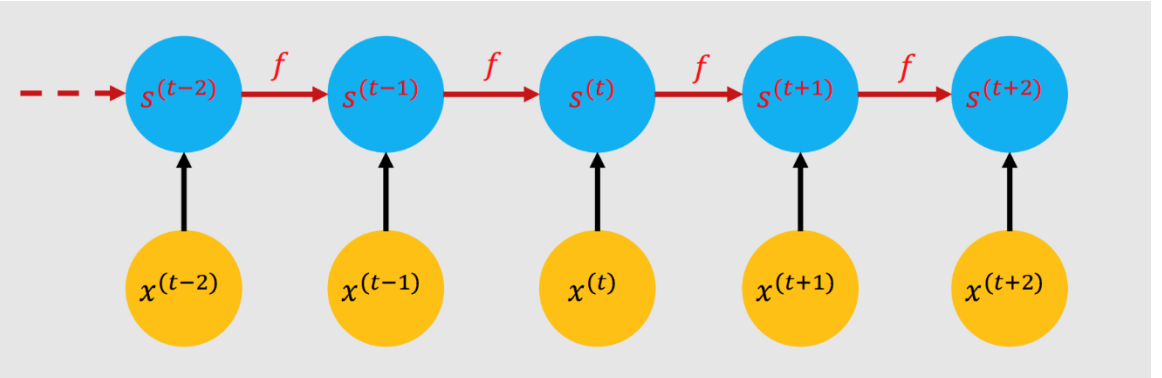


Quiz: Please choose the representation of $s^{(t+2)}$ in terms of

$s^{(t)}$, $x^{(t)}$, $x^{(t+1)}$, $x^{(t+2)}$ in the following dynamical system

$$s^{(t+1)} = f_\theta(s^{(t)}, x^{(t+1)}).$$

1. $f_\theta(s^{(t)}, x^{(t+1)})$
2. $f_\theta(s^{(t)}, x^{(t+2)})$
3. $f_\theta(f_\theta(s^{(t)}, x^{(t)}), x^{(t+1)})$
4. $f_\theta(f_\theta(s^{(t)}, x^{(t+1)}), x^{(t+2)})$



As is shown in this dynamic system, we have

$$s^{(t+2)} = f_\theta\left(s^{(t+1)}, x^{(t+2)}\right) = f_\theta\left(f_\theta\left(s^{(t)}, x^{(t+1)}\right), x^{(t+2)}\right),$$

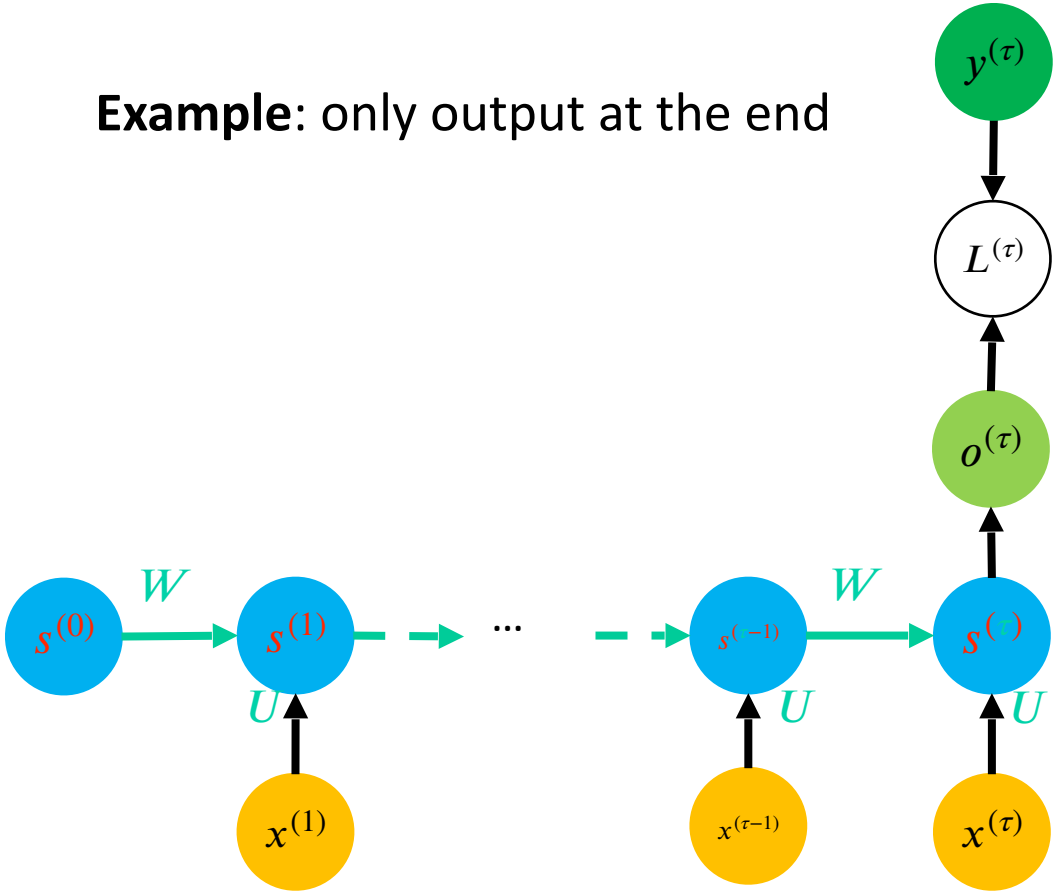
as $s^{(t+1)} = f_\theta\left(s^{(t)}, x^{(t+1)}\right)$.

Outline

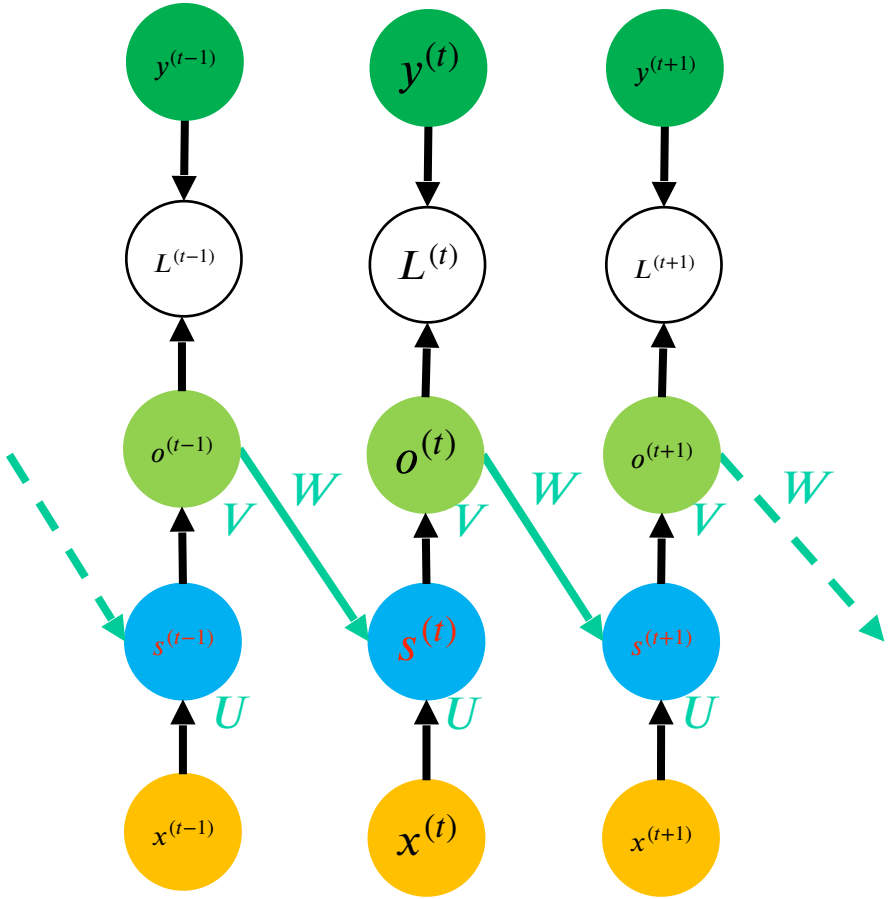
- **RNN Basics**
  - Sequential tasks, hidden state, vanilla RNN
- **RNN Variants + LSTMs**
  - RNN training, variants, LSTM cells

RNN Variants

Example: only output at the end



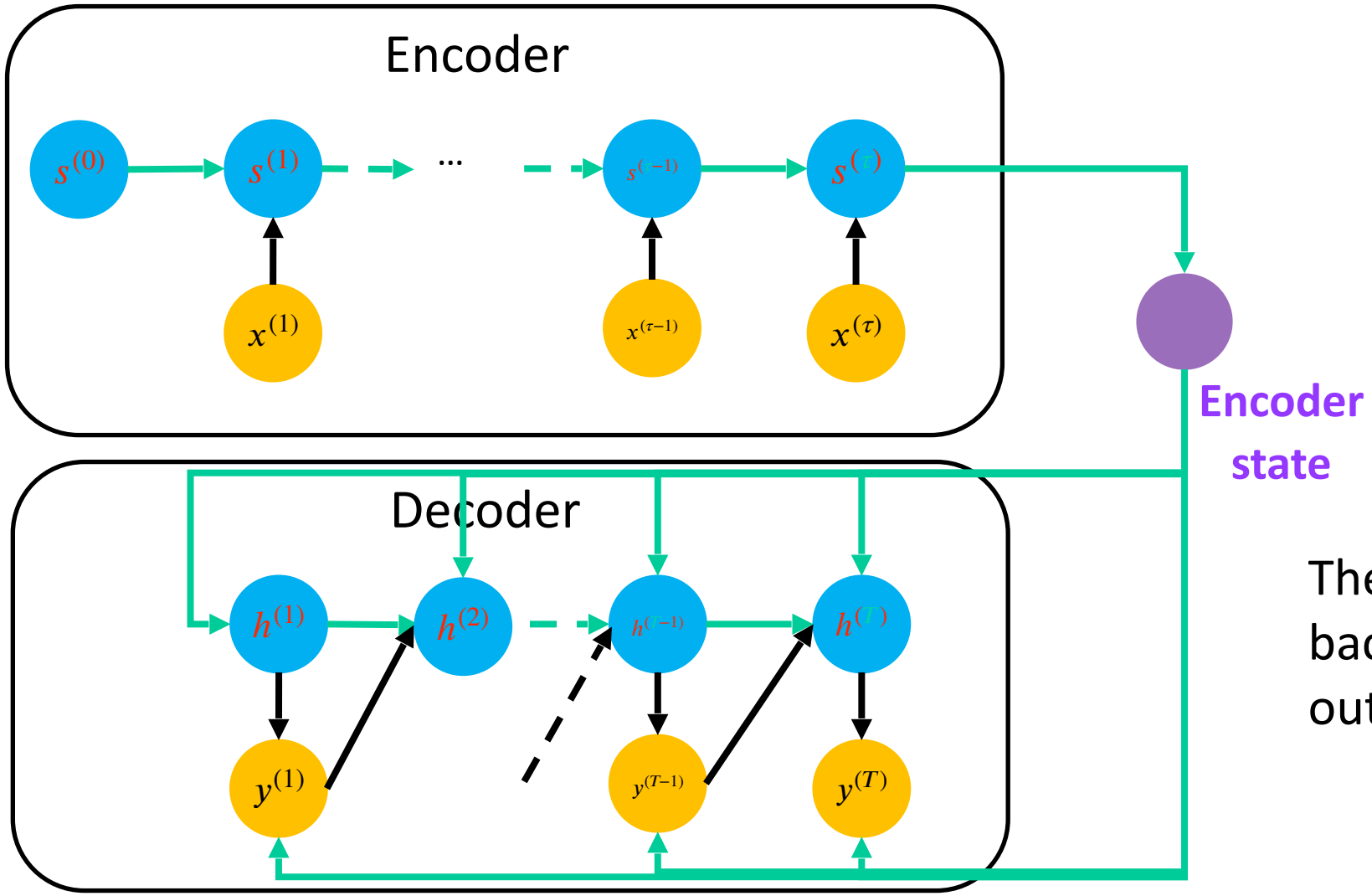
Example: use the output at the previous step



RNN Variants: Seq2Seq (a.k.a Encoder/Decoder) models

- What about mapping a sequence to a sequence of different length?
  - **Ex**: speech recognition, machine translation, question answering, etc.

RNN Variants: Seq2Seq models



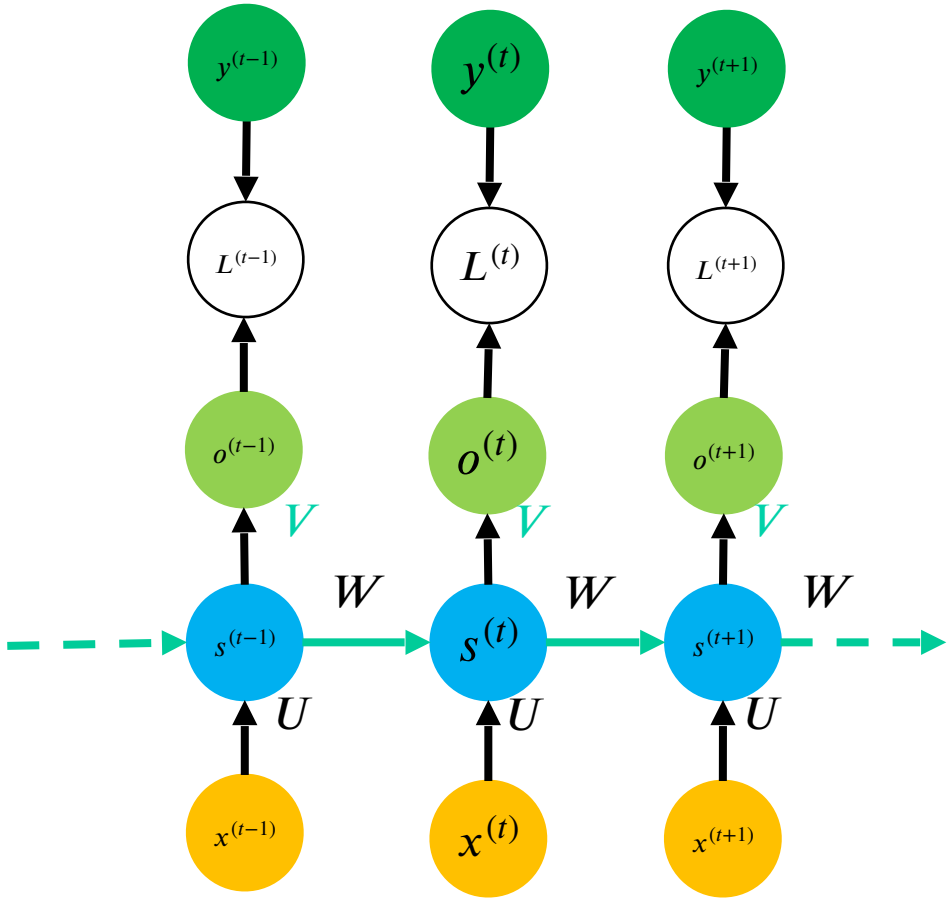
The encoder state is fed back to the decoder at each output.

Training RNNs

- Backpropagation Through Time
  - Idea: unfold the computational graph, and use backpropagation

- Conceptually: first compute the gradients of the internal nodes, then compute the gradients of the parameters

Training RNNs: computing gradients



$$a^{(t)} = b + Ws^{(t-1)} + Ux^{(t)}$$
$$s^{(t)} = \tanh\left(a^{(t)}\right)$$
$$o^{(t)} = c + Vs^{(t)}$$
$$\hat{y}^{(t)} = \text{softmax}\left(o^{(t)}\right)$$
$$L^{(t)} = \text{CrossEntropy}(y^{(t)}, \hat{y}^{(t)})$$

$$\frac{\partial L^{(t)}}{\partial U} = \frac{\partial L^{(t)}}{\partial s^{(t)}} \frac{\partial s^{(t)}}{\partial a^{(t)}} \frac{\partial a^{(t)}}{\partial U}$$
$$= \frac{\partial L^{(t)}}{\partial s^{(t)}} \frac{\partial s^{(t)}}{\partial a^{(t)}} \left(\frac{\partial (Ux^{(t)})}{\partial U} + \frac{\partial (Ws^{(t)})}{\partial s^{(t-1)}} \frac{\partial s^{(t-1)}}{\partial U} \right)$$

RNN Problems with long sequences

- **Training:** What happens to gradients in backprop w. many layers?
  - In an RNN trained on long sequences (*e.g.* 100 time steps) the gradients can easily explode or vanish.

- **Memory/retention:** very hard to detect that current target output **depends** on an input from long ago.
  - Simple RNNs have difficulty dealing with long-range dependencies.

Gradients for Neural Networks

- Compute the gradient of the loss ℓ w.r.t. $\mathbf{W}_t$

$$\frac{\partial \ell}{\partial \mathbf{W}^t} = \frac{\partial \ell}{\partial \mathbf{h}^d} \underbrace{\frac{\partial \mathbf{h}^d}{\partial \mathbf{h}^{d-1}} \cdots \frac{\partial \mathbf{h}^{t+1}}{\partial \mathbf{h}^t}} \frac{\partial \mathbf{h}^t}{\partial \mathbf{W}^t}$$

Multiplication of *many* matrices

Wikipedia

Two Issues for very Deep Neural Networks and RNNs

Gradient Exploding

$$1.5^{100} \approx 4 \times 10^{17}$$

Gradient Vanishing

$$\prod_{i=t}^{d-1} \frac{\partial \mathbf{h}^{i+1}}{\partial \mathbf{h}^{i}}$$

$$0.8^{100} \approx 2 \times 10^{-10}$$

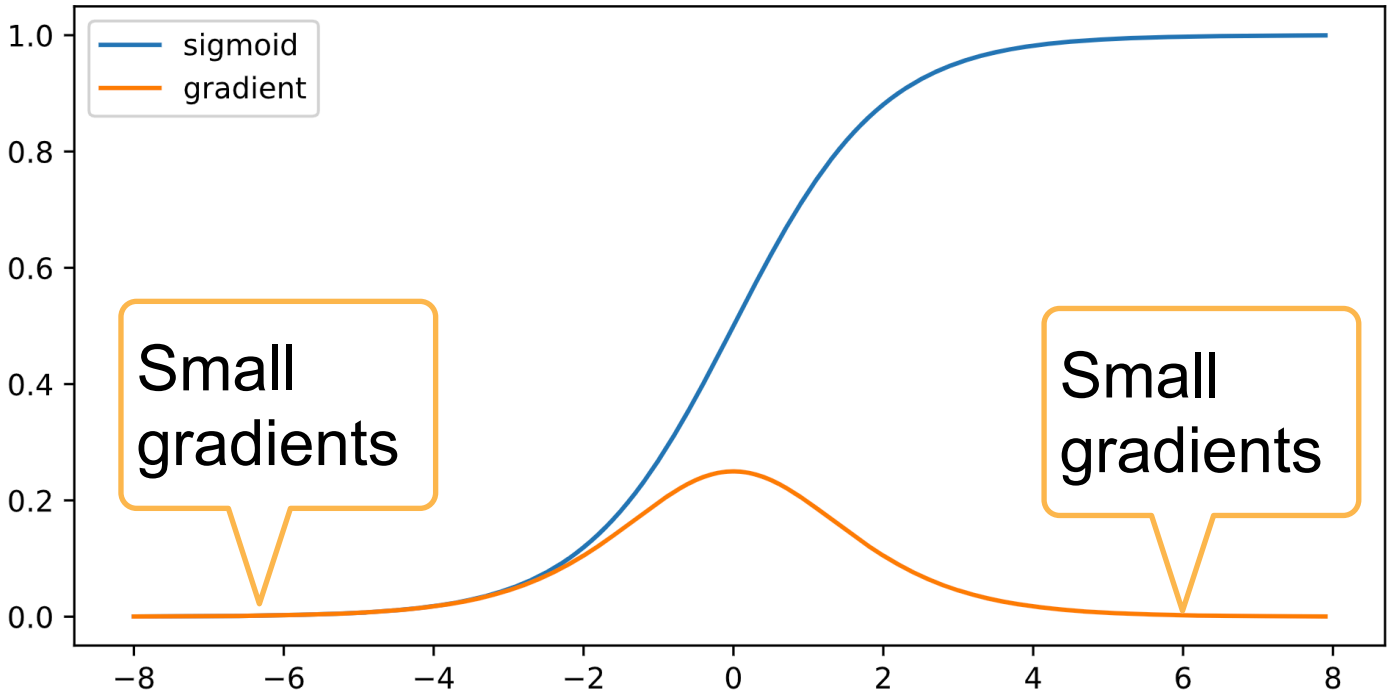
Issues with Gradient Exploding

- Value out of range: infinity value (NaN)
- Sensitive to learning rate (LR)
  - Not small enough LR -> larger gradients
  - Too small LR -> No progress
  - May need to change LR dramatically during training

Gradient Vanishing

- Use sigmoid as the activation function

$$\sigma(x) = \frac{1}{1 + e^{-x}} \qquad \sigma'(x) = \sigma(x)(1 - \sigma(x))$$



Stabilize Training: Practical Considerations

- Goal: make sure gradient values are in a proper range
  - E.g. in [1e-6, 1e3]
- Multiplication -> plus
  - Architecture change (e.g., ResNet)
- Normalize
  - Batch Normalization, Gradient clipping
- Proper activation functions
- Proper initialization

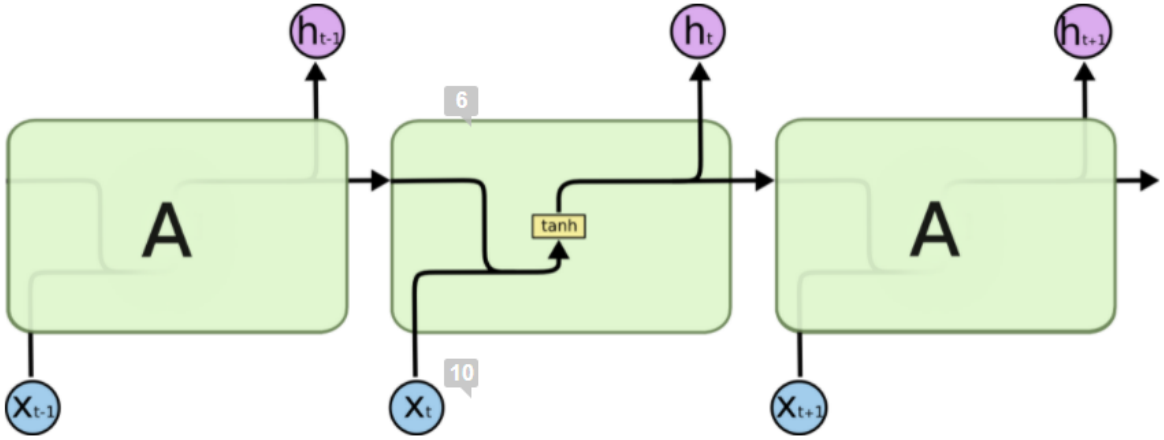
RNN Problems with long sequences

- **Training:** What happens to gradients in backprop w. many layers?
  - In an RNN trained on long sequences (*e.g.* 100 time steps) the gradients can easily explode or vanish.
  - We can avoid this by initializing the weights very carefully.

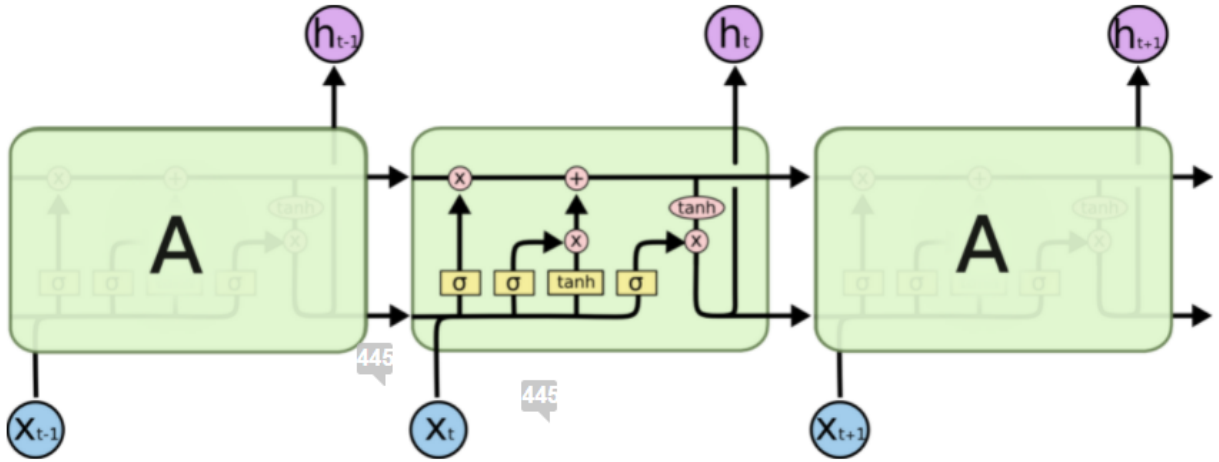
- **Memory/retention:** very hard to detect that current target output **depends** on an input from long ago.
  - Simple RNNs have difficulty dealing with long-range dependencies.

LSTM (Long Short-Term Memory)

- RNN: can write structure as:

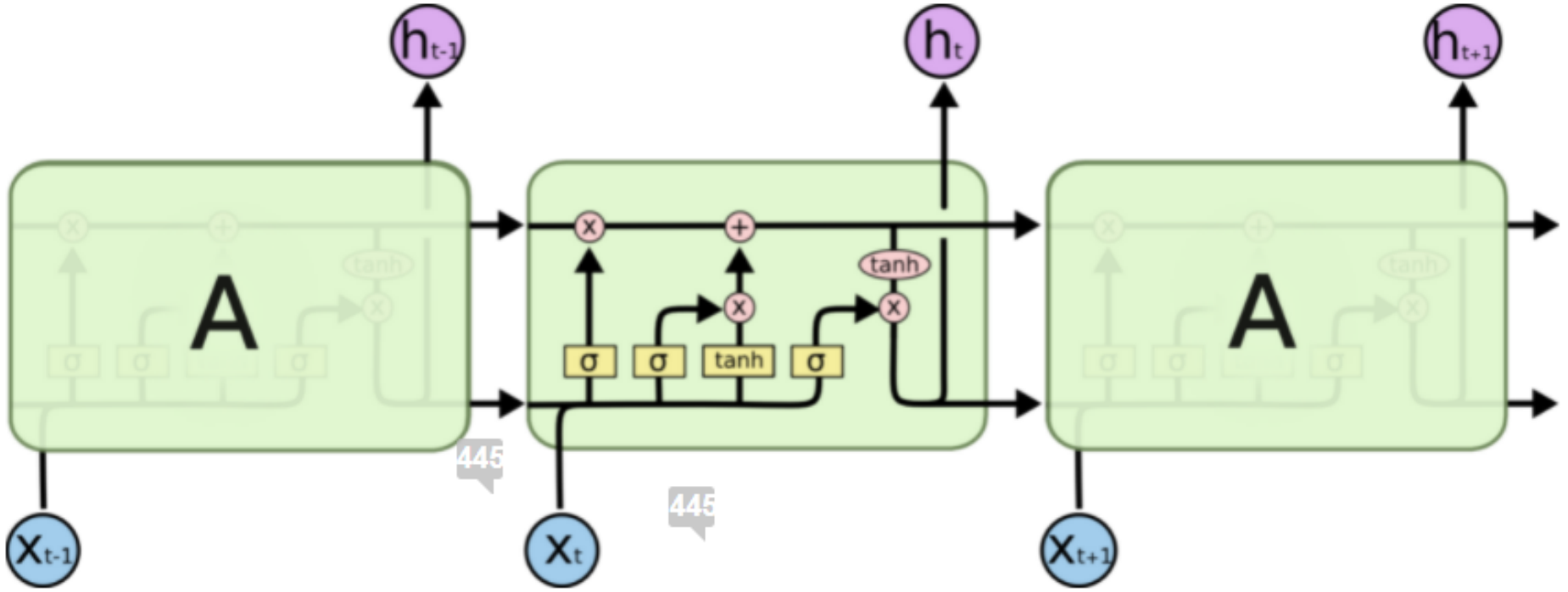


- Long Short-Term Memory Cell: deals with problem of long-term dependencies.



LSTM (Long Short-Term Memory)

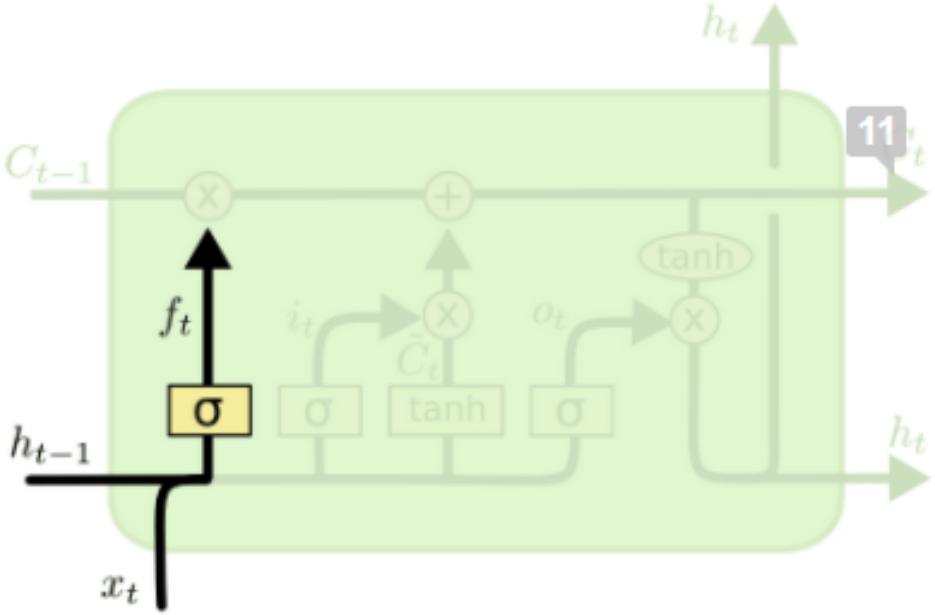
- Long Short-Term Memory Cell: deals with problem.



- h_t: short term memory
- C_t: long term memory

Understanding the LSTM Cell

- Step-by-step
  - Good reference: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

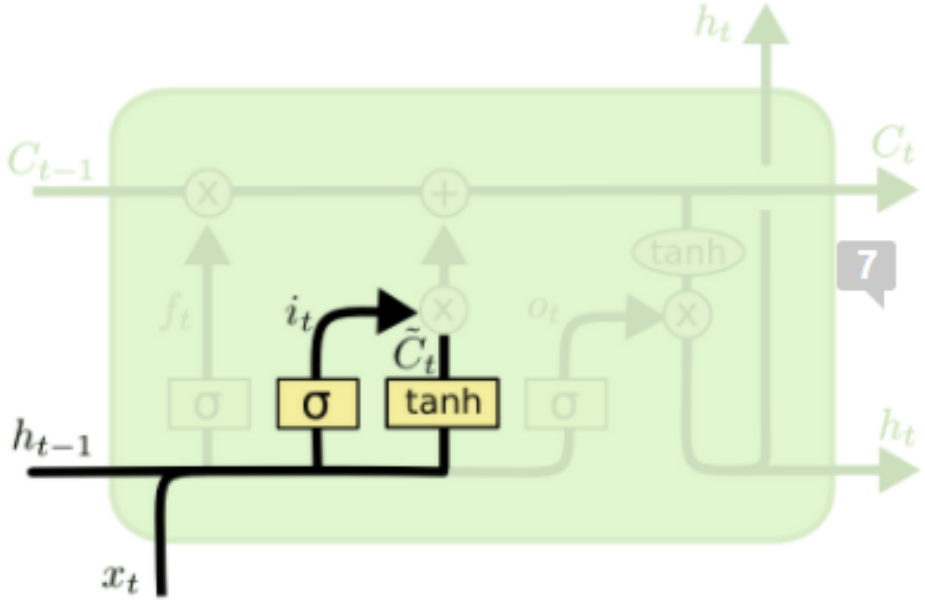


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

- "**Forget**" gate.
  - Can remove all or part of any entry in cell state C
  - σ denotes the sigmoid (logistic) activation; think of this as a soft on/off function

Understanding the LSTM Cell

- Step-by-step



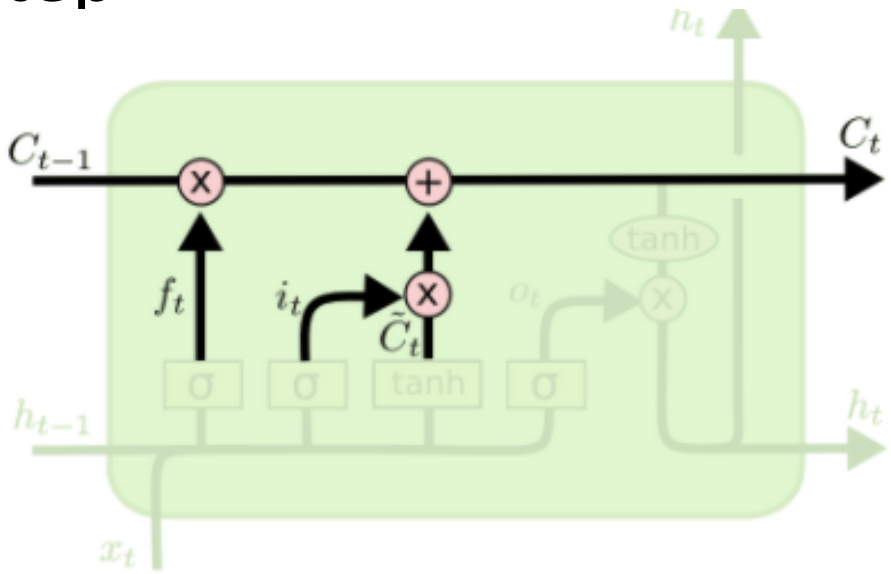
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$

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

- **Input** gate. Combine:
  - What entries in C_{t-1} we'll update
  - Candidates for updating: $\acute{C}_t$
  - Add information to cell state C_{t-1} (post-forgetting)

Understanding the LSTM Cell

- Step-by-step

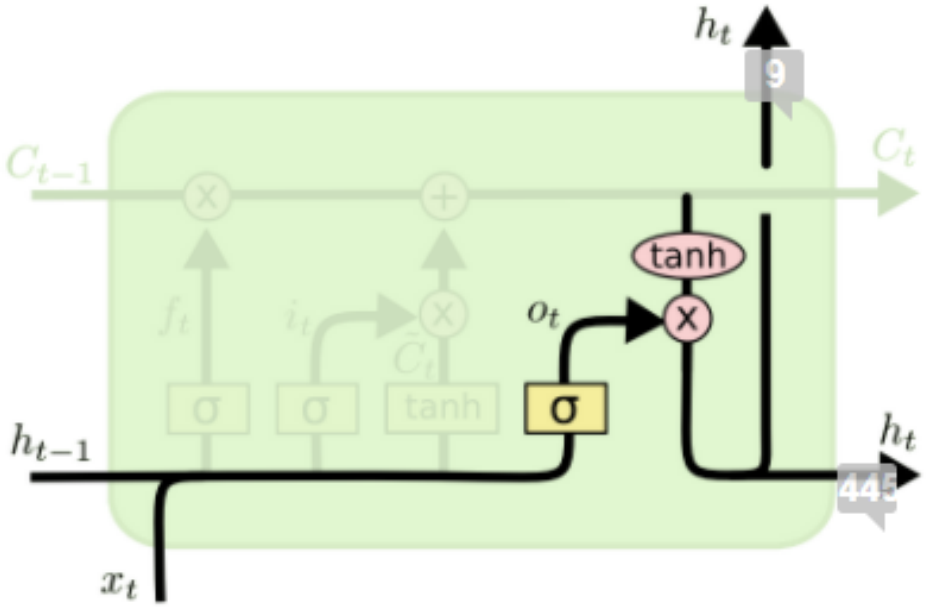


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

- Updating C_{t-1} to C_t
  - Forget, then
  - Add new information

Understanding the LSTM Cell

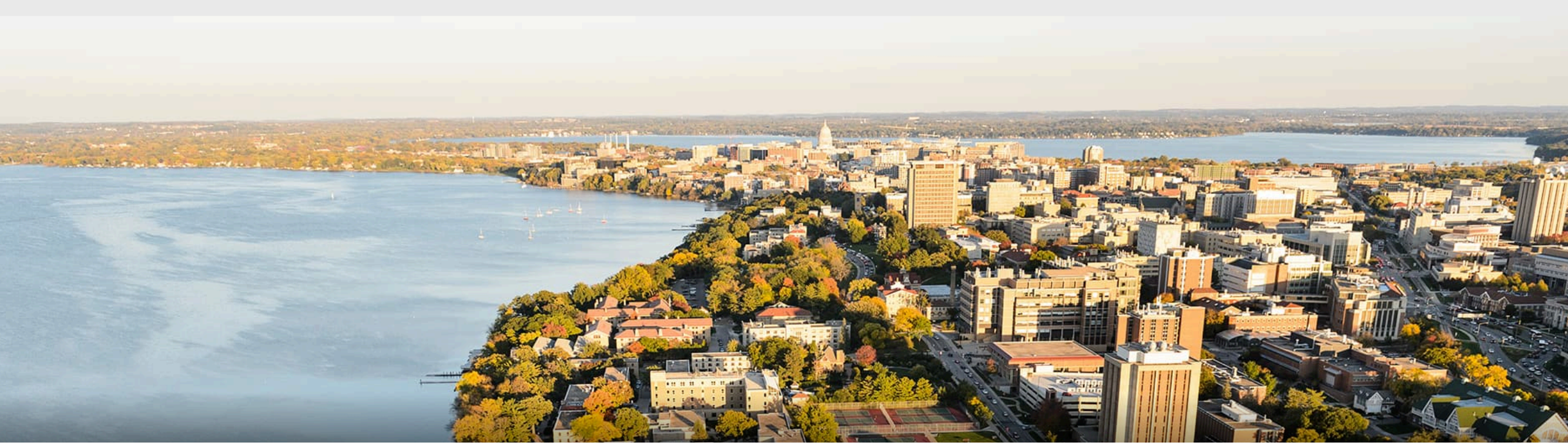
- Step-by-step



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

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

- **Output** gate
  - Combine hidden state, input as before, but also
  - Modify according to cell state C_t



Thanks Everyone!

Some of the slides in these lectures have been adapted/borrowed from materials developed by Mark Craven, David Page, Jude Shavlik, Tom Mitchell, Nina Balcan, Elad Hazan, Tom Dietterich, Pedro Domingos, Jerry Zhu, Yingyu Liang, Volodymyr Kuleshov , Sharon Li, Chris Olah