Recurrent Neural Networks

Machine Learning Decal

Hosted by Machine Learning at Berkeley

ML@B

Agenda

Motivation

Basic Recurrent Layers

Long Short-Term Memory

Demo: Language Modeling

RNNs vs CNNs

Questions

Motivation

Time Series Tasks

ML@B

- **Speech recognition**: Transcribing audio of a person talking into the corresponding words

Time Series Tasks

ML@B

- **Speech recognition**: Transcribing audio of a person talking into the corresponding words
- **Video Classification**: Label a video containing a sequence of image frames into its appropriate category (e.g. action done in video)

Time Series Tasks

ML@B

- **Speech recognition**: Transcribing audio of a person talking into the corresponding words
- **Video Classification**: Label a video containing a sequence of image frames into its appropriate category (e.g. action done in video)
- **Music Composition**: Create novel pieces of music

Time Series Tasks

ML@B

- **Speech recognition**: Transcribing audio of a person talking into the corresponding words
- **Video Classification**: Label a video containing a sequence of image frames into its appropriate category (e.g. action done in video)
- **Music Composition**: Create novel pieces of music
- **Stock Prediction**: Predict the behavior of stocks given past trends

Time Series Tasks

ML@B

- **Speech recognition**: Transcribing audio of a person talking into the corresponding words
- **Video Classification**: Label a video containing a sequence of image frames into its appropriate category (e.g. action done in video)
- **Music Composition**: Create novel pieces of music
- **Stock Prediction**: Predict the behavior of stocks given past trends
- **Image Captioning**: Write a caption for a particular image

Image Captioning

"man in black shirt is playing guitar."

"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

Music Composition

ML@B

Play

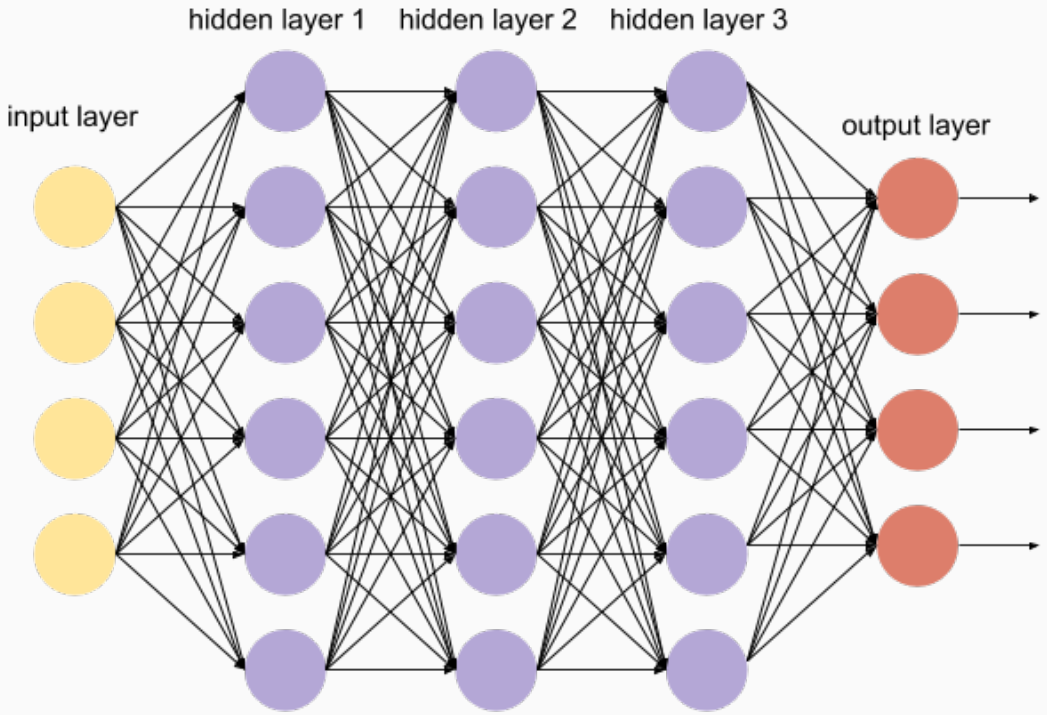
Using Regular Neural Networks

ML@B

Let's say we want to predict the following word given the phrase "the quick brown fox" (this is called a **language model**)

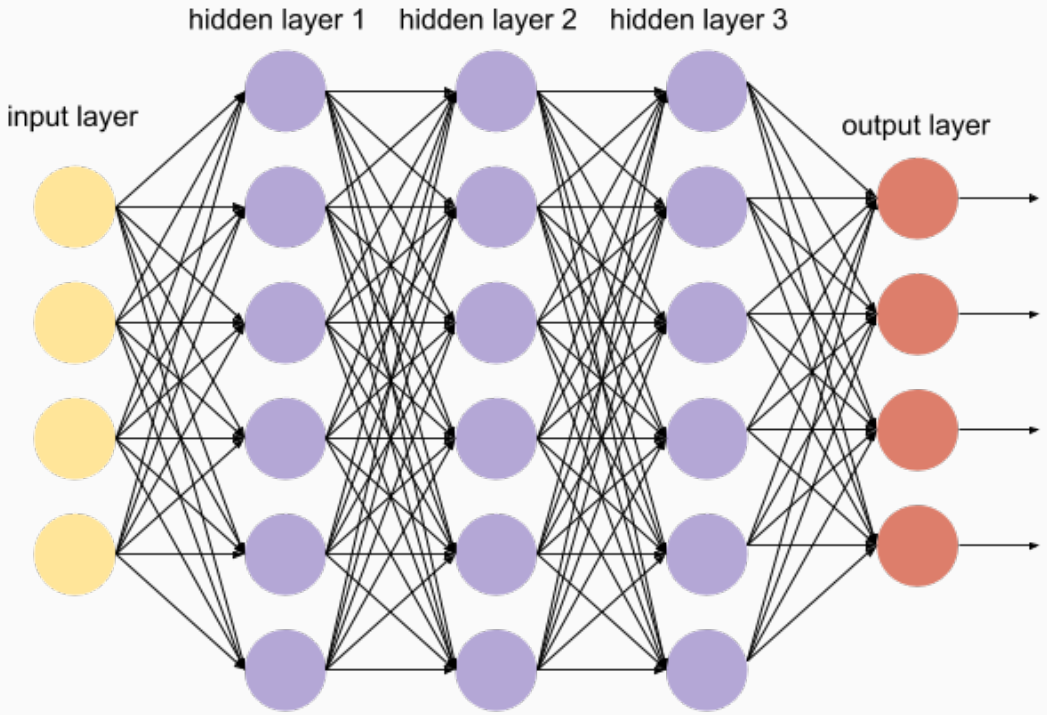
Using Regular Neural Networks

Let's say we want to predict the following word given the phrase "the quick brown fox" (this is called a **language model**)



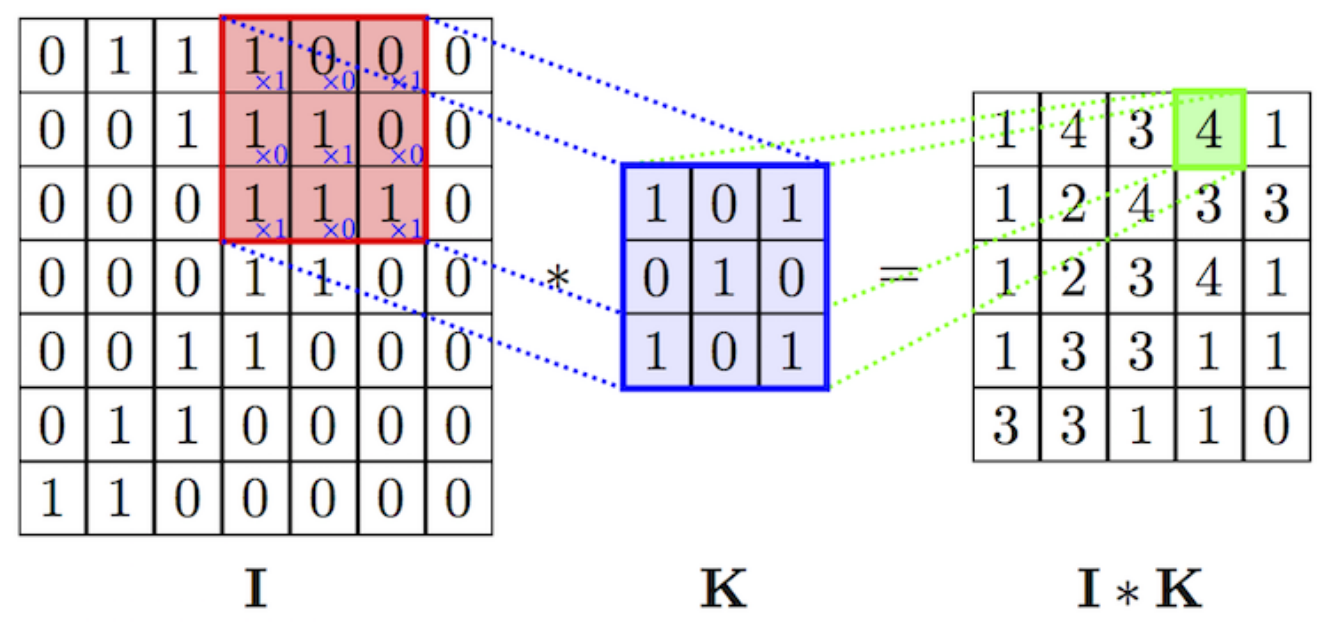
Using Regular Neural Networks

Let's say we want to predict the following word given the phrase "the quick brown fox" (this is called a **language model**)



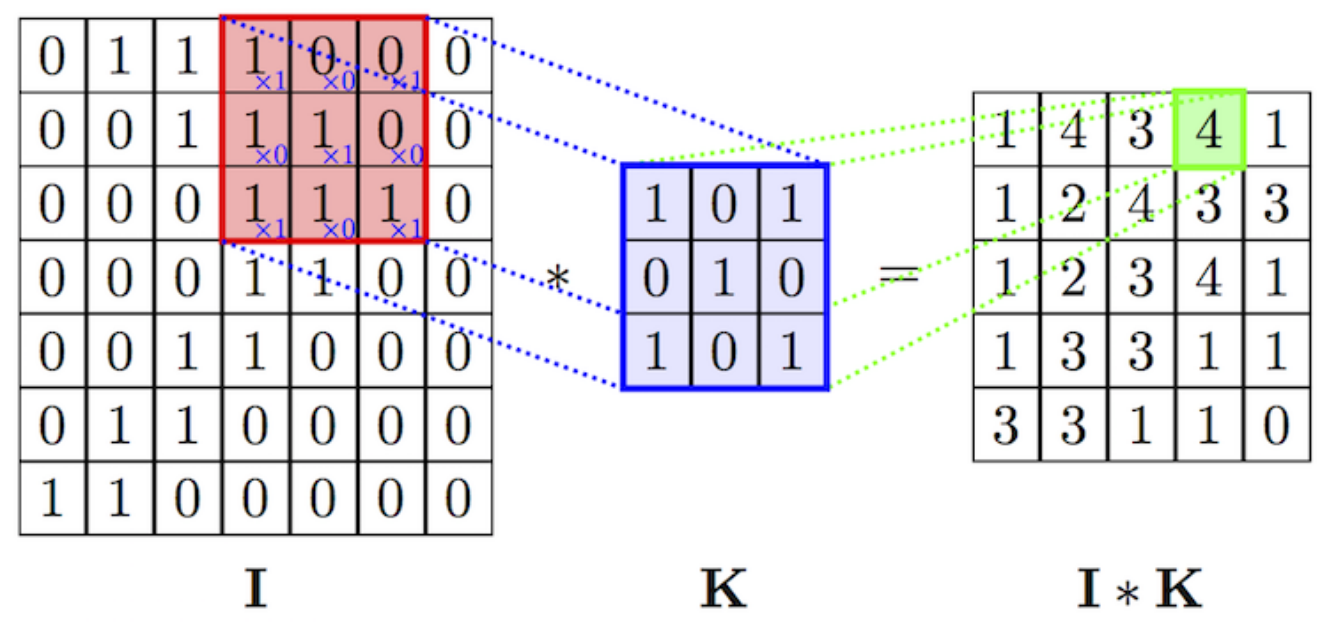
How could we train a standard neural network to perform this task? What are some issues that can arise?

Revisiting CNNs



Recall from last time how convolutional neural networks take advantage of spatial locality in images to learn more efficiently

Revisiting CNNs

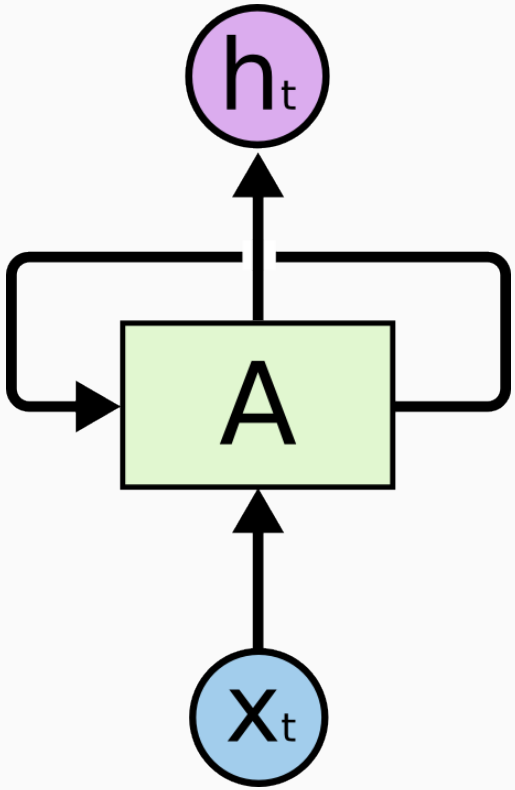


Recall from last time how convolutional neural networks take advantage of spatial locality in images to learn more efficiently

How can we exploit common structure in time series tasks?

Basic Recurrent Layers

Adding Recurrence



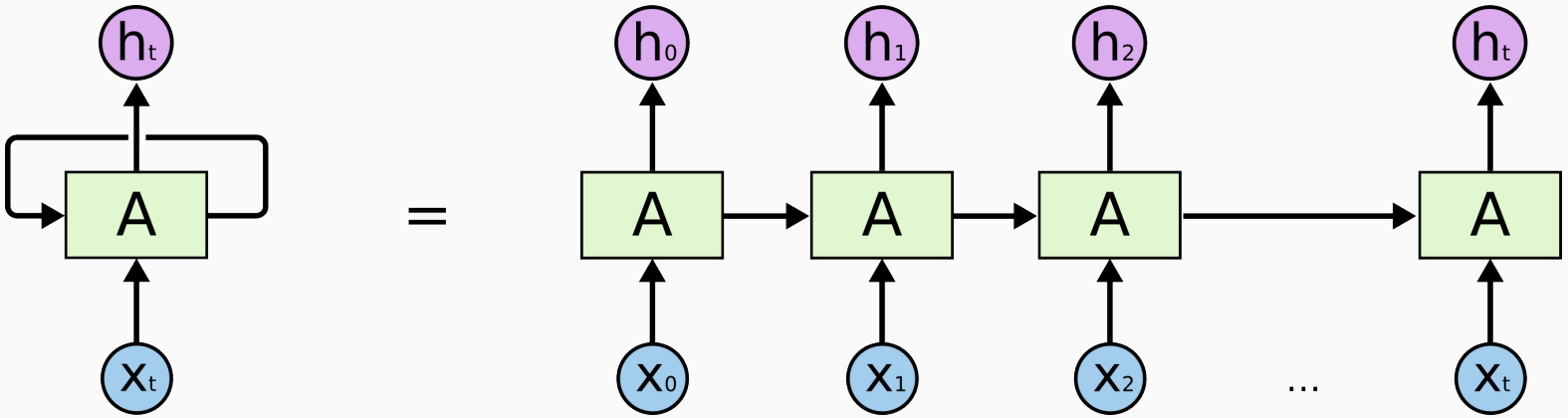
Key idea: not only do we give the layer the input, but also the previous output of the same layer!

Unrolling the RNN

We can **unroll** this recurrent layer to get a better sense of how it works over time.

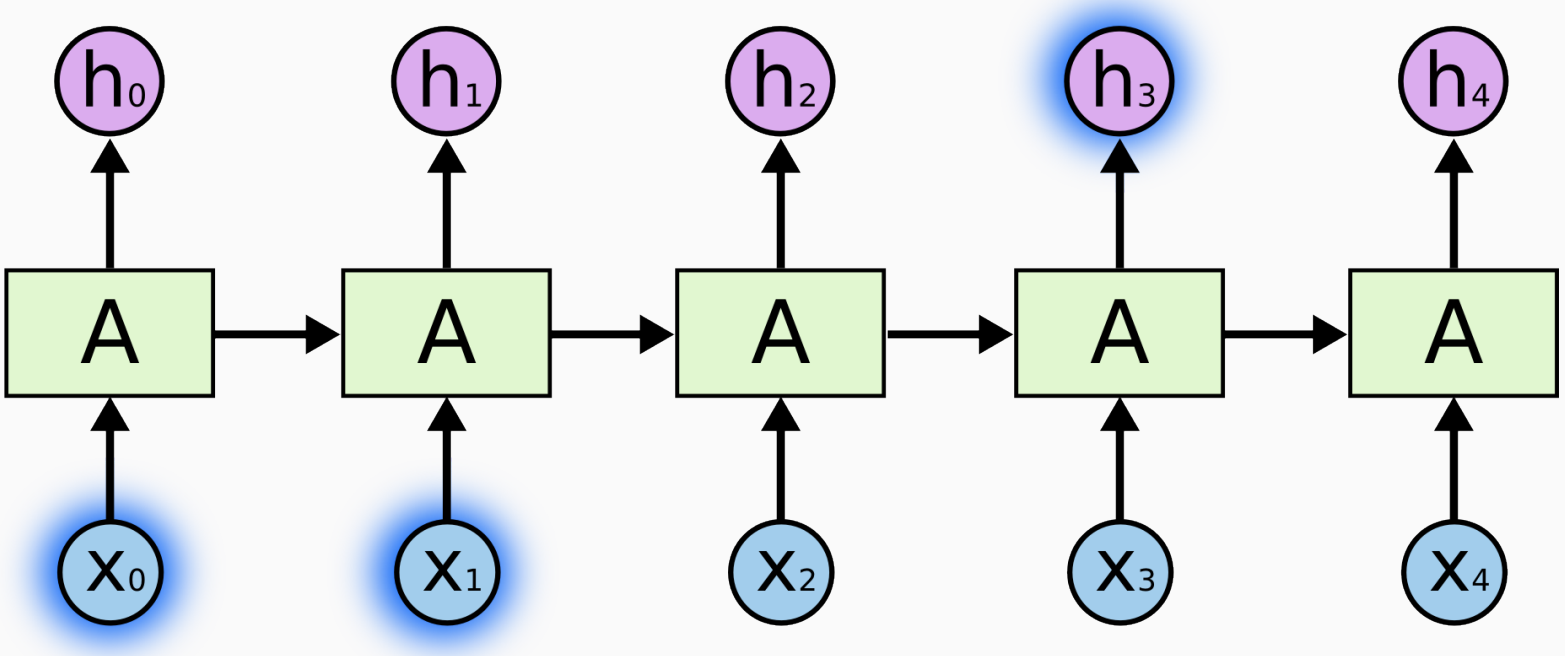
Unrolling the RNN

We can **unroll** this recurrent layer to get a better sense of how it works over time.



RNN Output

The output at any particular timestep now depends on all the inputs up to that timestep!

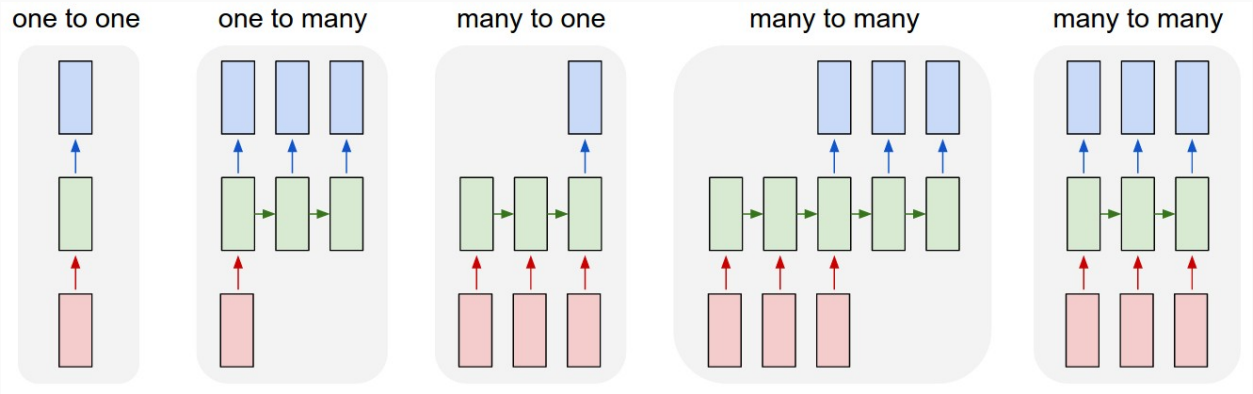


Types of RNN Layers

Recurrent Neural Networks can come in different varieties, depending on what the task is.

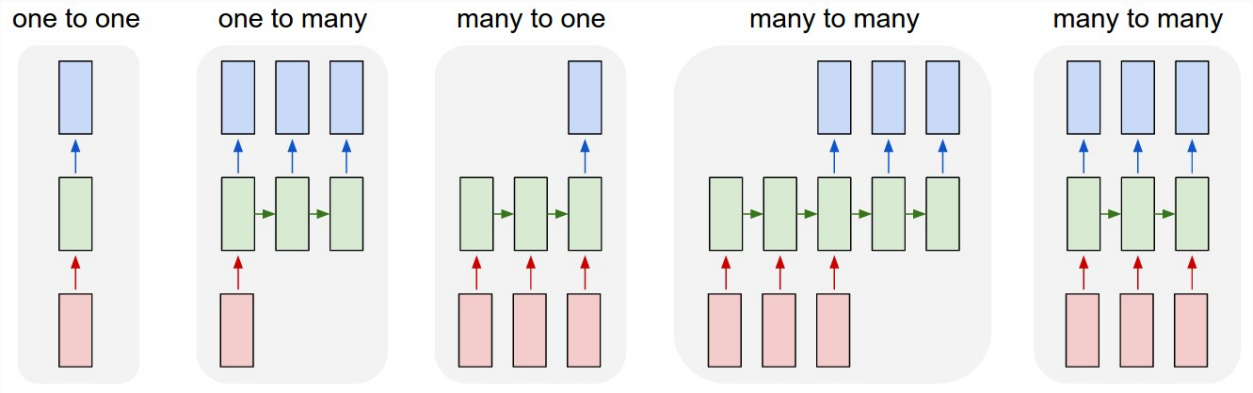
Types of RNN Layers

Recurrent Neural Networks can come in different varieties, depending on what the task is.



Types of RNN Layers

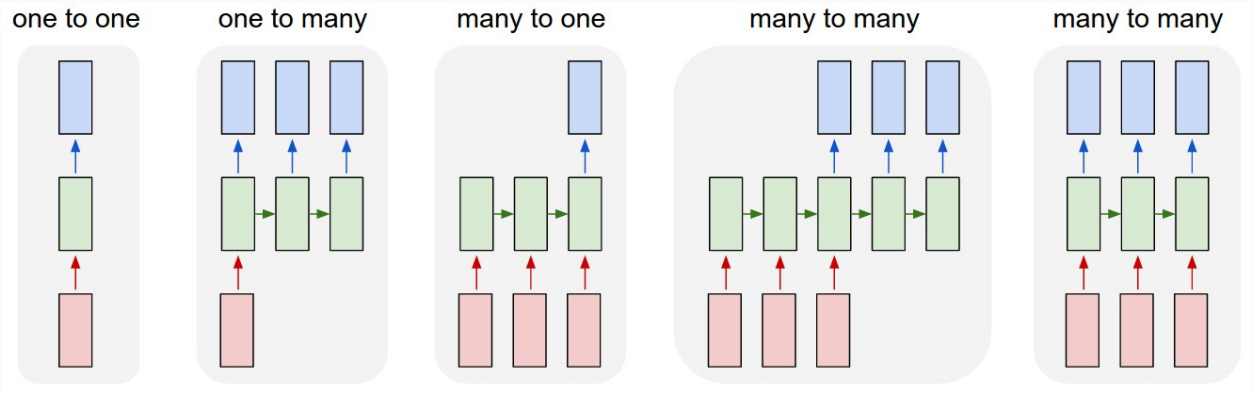
Recurrent Neural Networks can come in different varieties, depending on what the task is.



- **One-to-one**: Standard Neural Network

Types of RNN Layers

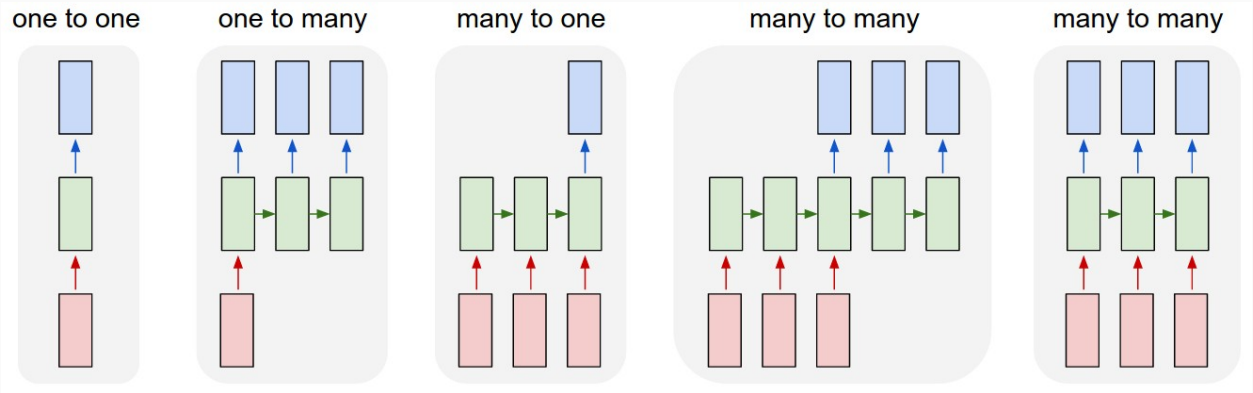
Recurrent Neural Networks can come in different varieties, depending on what the task is.



- **One-to-one**: Standard Neural Network
- **One-to-many**: Image Captioning

Types of RNN Layers

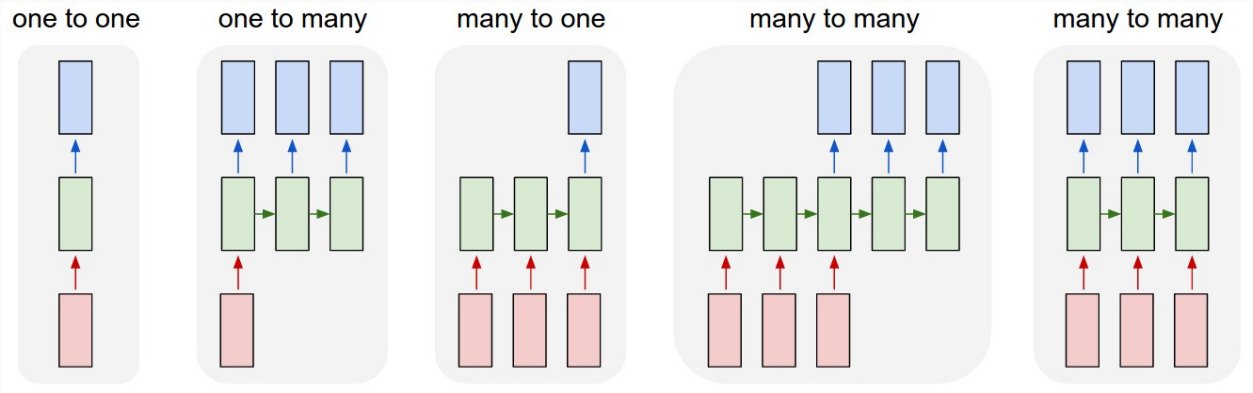
Recurrent Neural Networks can come in different varieties, depending on what the task is.



- **One-to-one**: Standard Neural Network
- **One-to-many**: Image Captioning
- **Many-to-one**: Sentiment Analysis

Types of RNN Layers

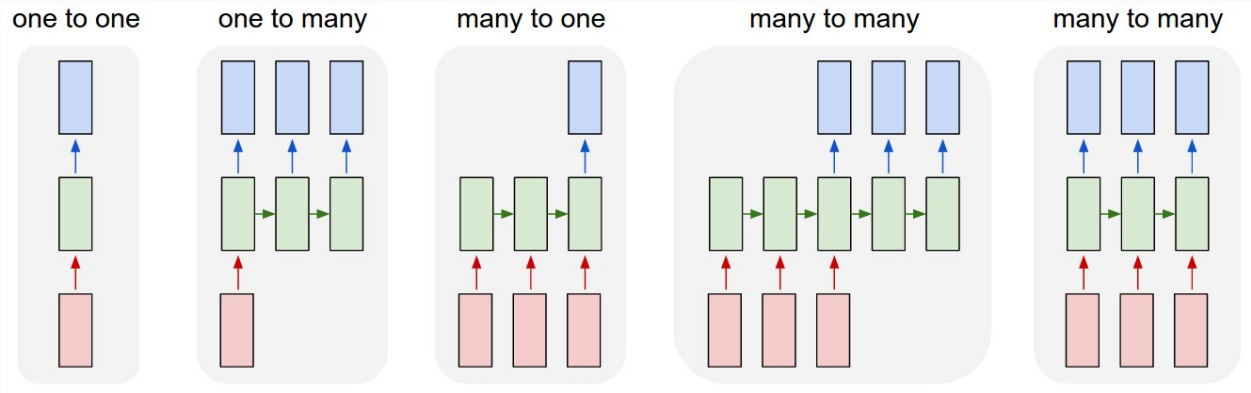
Recurrent Neural Networks can come in different varieties, depending on what the task is.



- **One-to-one**: Standard Neural Network
- **One-to-many**: Image Captioning
- **Many-to-one**: Sentiment Analysis
- **Many-to-many (type 1)**: Machine Translation

Types of RNN Layers

Recurrent Neural Networks can come in different varieties, depending on what the task is.



- **One-to-one**: Standard Neural Network
- **One-to-many**: Image Captioning
- **Many-to-one**: Sentiment Analysis
- **Many-to-many (type 1)**: Machine Translation
- **Many-to-many (type 2)**: Video Classification by Frame

Structure of an RNN Cell

ML@B

- Recall that for the fully connected neural net, each layer is as follows:

$$h = f(Wx + b)$$

where f is a nonlinear activation function

Structure of an RNN Cell

ML@B

- Recall that for the fully connected neural net, each layer is as follows:

$$h = f(Wx + b)$$

  where f is a nonlinear activation function

- Since we now want to include the previous timestep's output as input, the new layer will look like

$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$

Structure of an RNN Cell

ML@B

- Recall that for the fully connected neural net, each layer is as follows:

$$h = f(Wx + b)$$

  where f is a nonlinear activation function

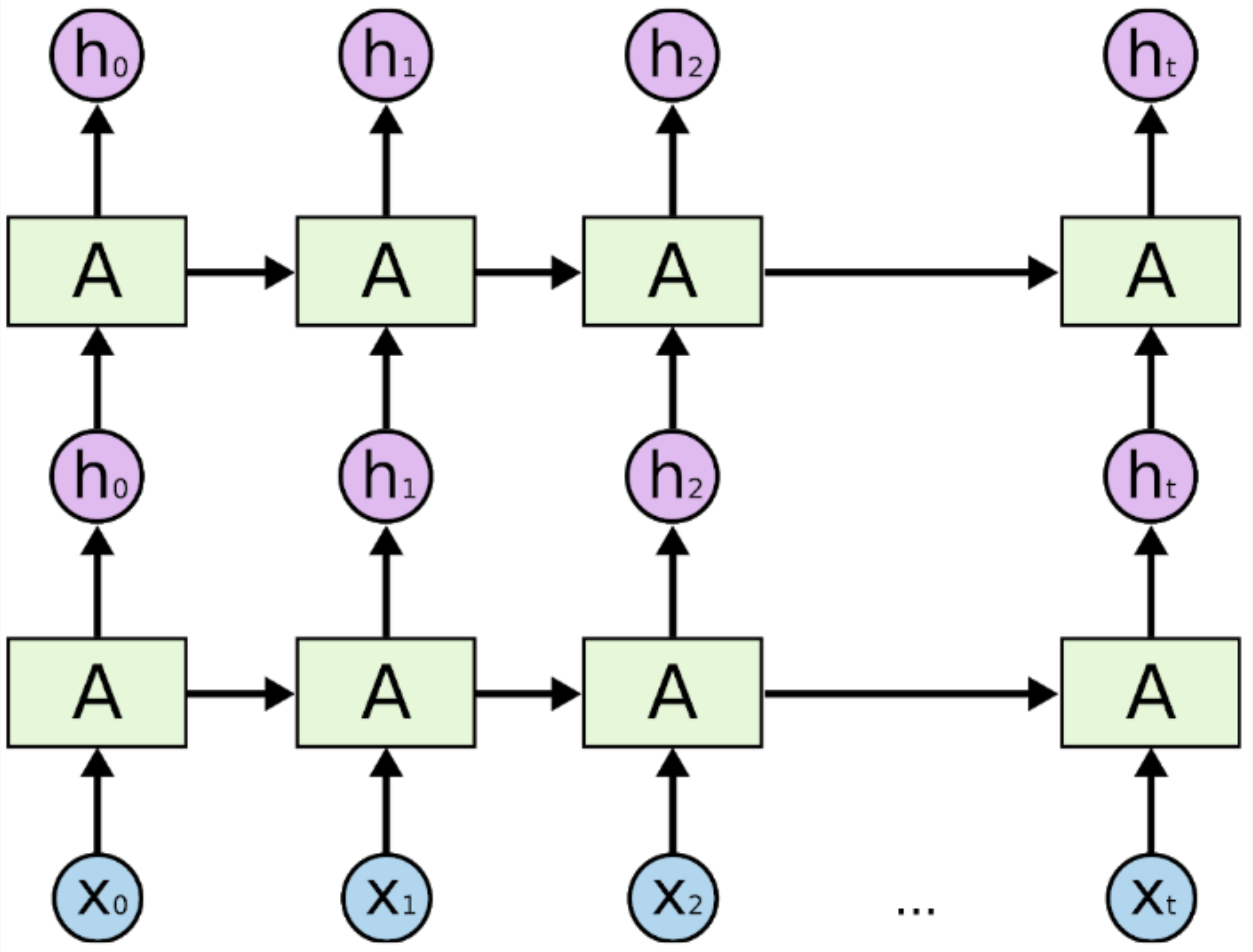
- Since we now want to include the previous timestep's output as input, the new layer will look like

$$h_t = \tanh(W_h h_{t-1} + W_x x_t)$$

- We use a tanh activation so that values within the hidden layer don't explode over long time periods

Multi-layer RNNs

Note: We can stack multiple recurrent layers, just like fully connected NNs and CNNs!



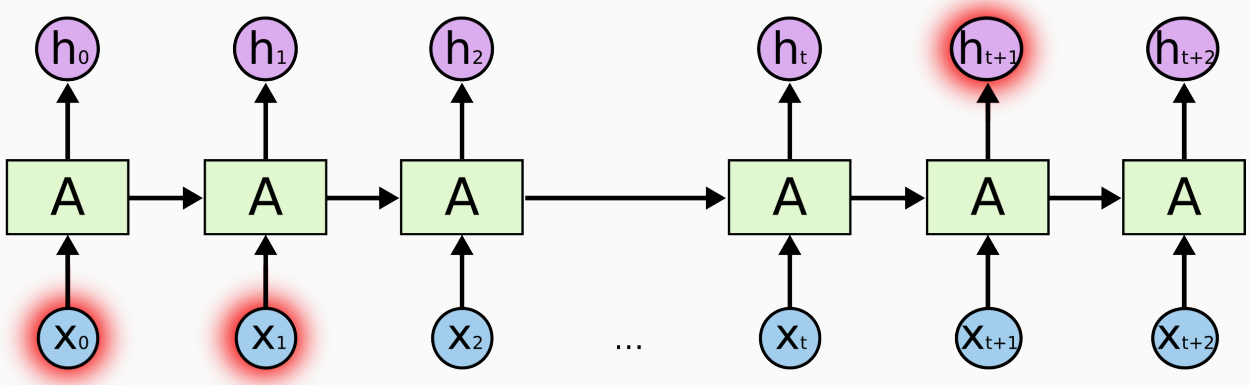
Vanishing Gradients in RNNs

ML@B

- Basic RNNs have a glaring problem: while the tanh activation can prevent **exploding gradients**, it cannot mitigate **vanishing gradients**!

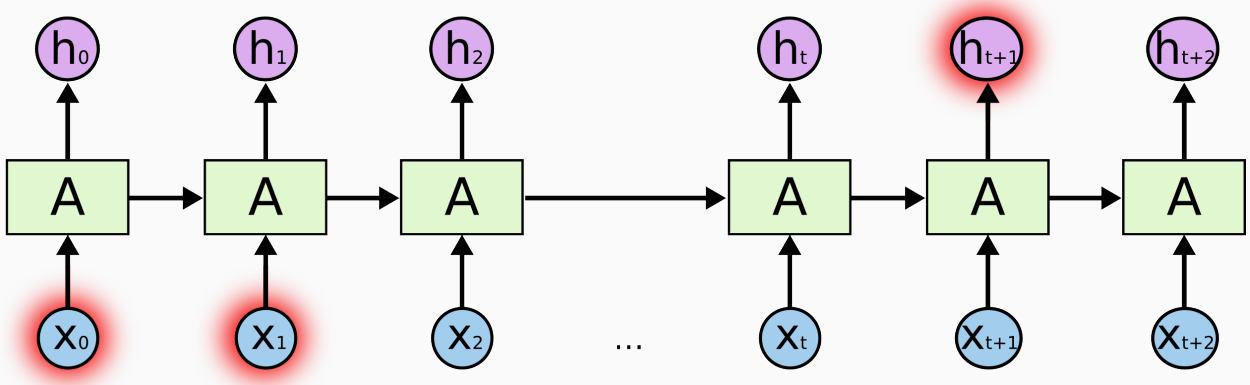
Vanishing Gradients in RNNs

- Basic RNNs have a glaring problem: while the tanh activation can prevent **exploding gradients**, it cannot mitigate **vanishing gradients**!



Vanishing Gradients in RNNs

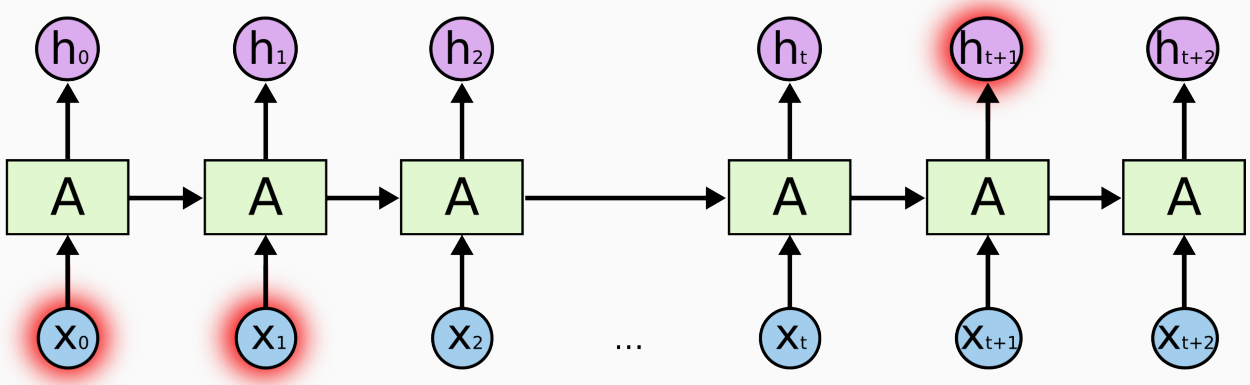
- Basic RNNs have a glaring problem: while the tanh activation can prevent **exploding gradients**, it cannot mitigate **vanishing gradients**!



- The gradients of tanh are always $\leqslant 1$, so when applying chain rule for backpropagation, the gradients can only get smaller.

Vanishing Gradients in RNNs

- Basic RNNs have a glaring problem: while the tanh activation can prevent **exploding gradients**, it cannot mitigate **vanishing gradients**!



- The gradients of tanh are always $\leqslant 1$, so when applying chain rule for backpropagation, the gradients can only get smaller.
- For very long dependencies, these gradients approach zero!

Long Short-Term Memory

Introducting LSTMs

ML@B

- Long-term memory, when talking about ML algorithms, are the learned parameters of the model, since they are held indefinitely once trained.

Introducting LSTMs

ML@B

- Long-term memory, when talking about ML algorithms, are the learned parameters of the model, since they are held indefinitely once trained.
- The hidden states in RNNs are short-term memory, since they last only for the current prediction and are then cleared,

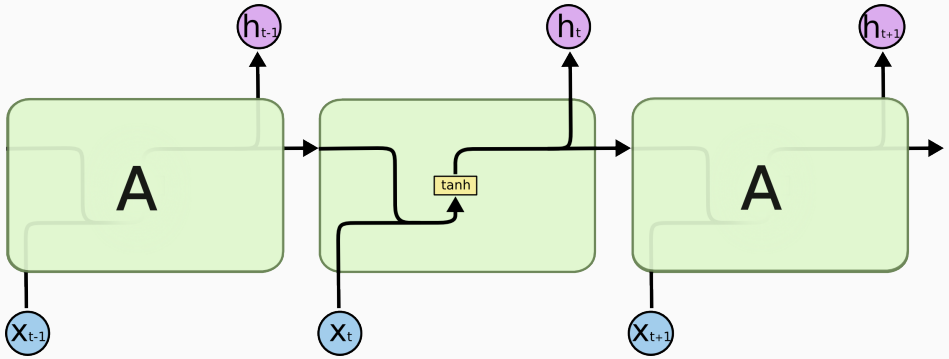
Introducting LSTMs

ML@B

- Long-term memory, when talking about ML algorithms, are the learned parameters of the model, since they are held indefinitely once trained.
- The hidden states in RNNs are short-term memory, since they last only for the current prediction and are then cleared,
- LSTMs allow for the short-term memory of RNNs to be held for much longer than the basic RNN cell.

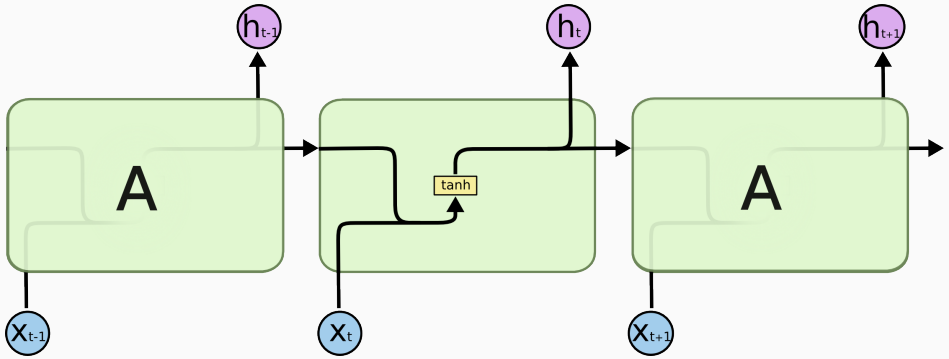
Introducting LSTMs

We go from this

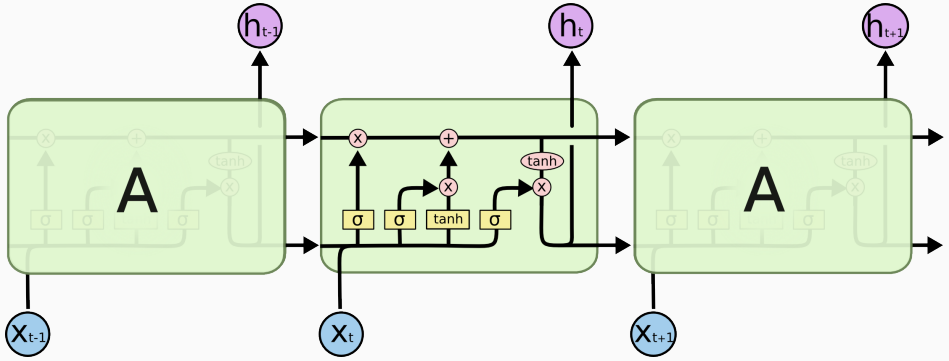


Introducting LSTMs

We go from this

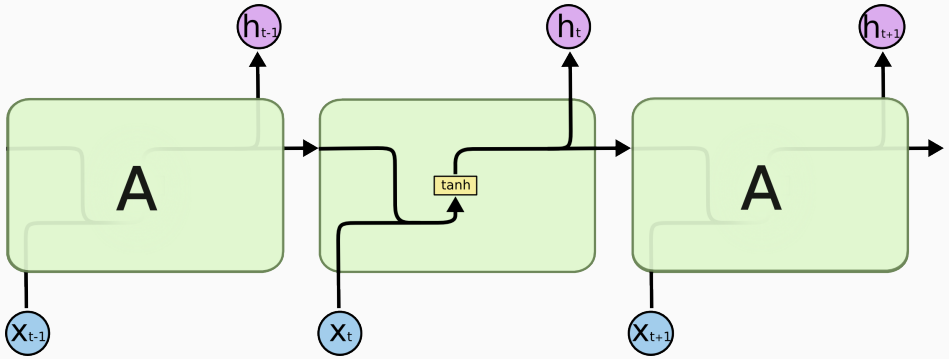


to this

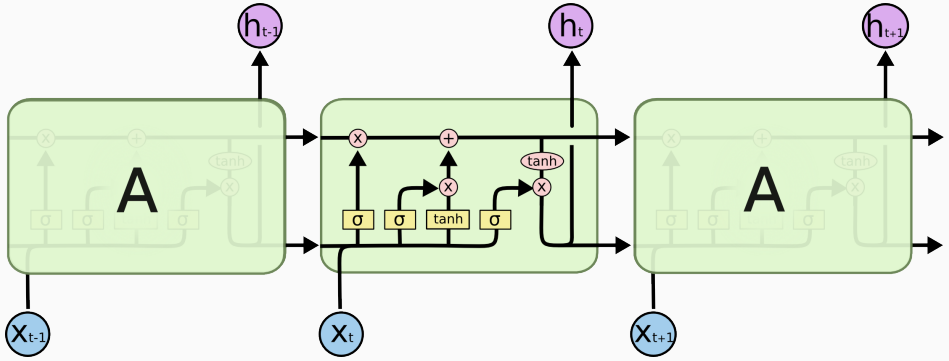


Introducting LSTMs

We go from this

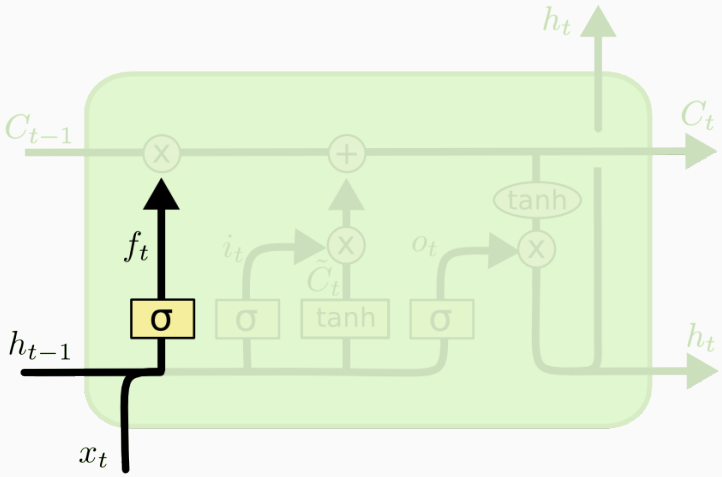


to this



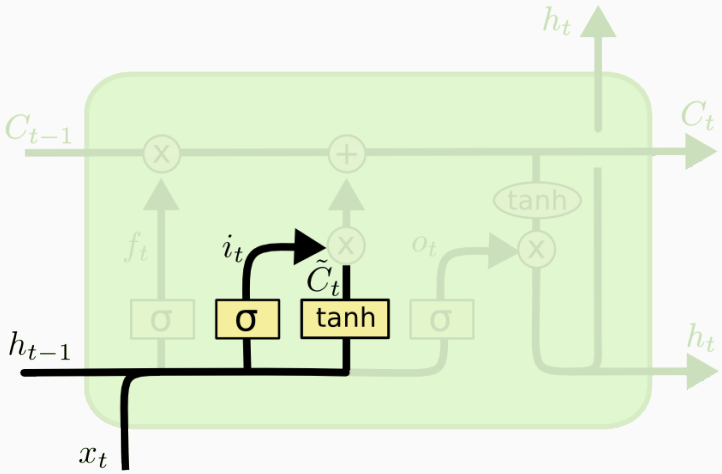
Let's break it down...

LSTMs: The Forget Gate



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

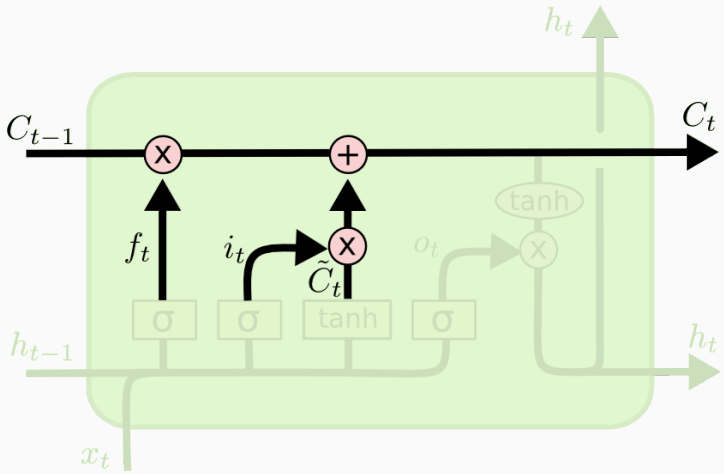
LSTMs: The Remember Gate



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

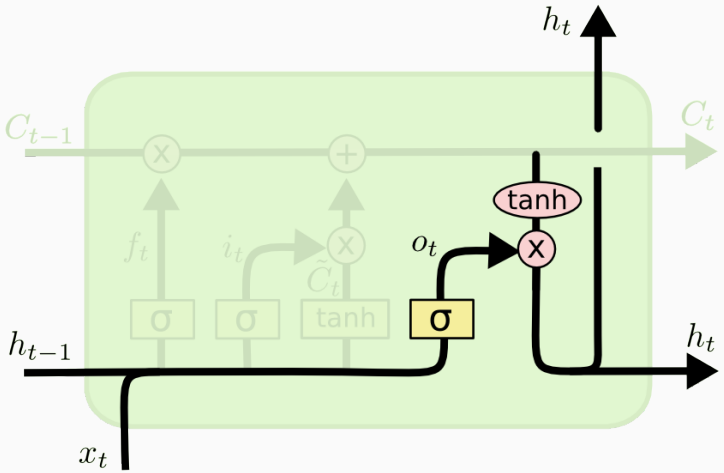
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

LSTMs: Updating the Cell State



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

LSTMs: Updating the Hidden State



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

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

Demo: Language Modeling

The Language Modeling Task

ML@B

- **Language modeling** is the problem of predicting the next word or letter given the previous words/letter.

The Language Modeling Task

ML@B

- **Language modeling** is the problem of predicting the next word or letter given the previous words/letter.
- Mathematically, it is predicting

$$P(W_n = w_n | W_{n-1} = w_{n-1}, W_{n-2} = w_{n-2}, \ldots)$$

The Language Modeling Task

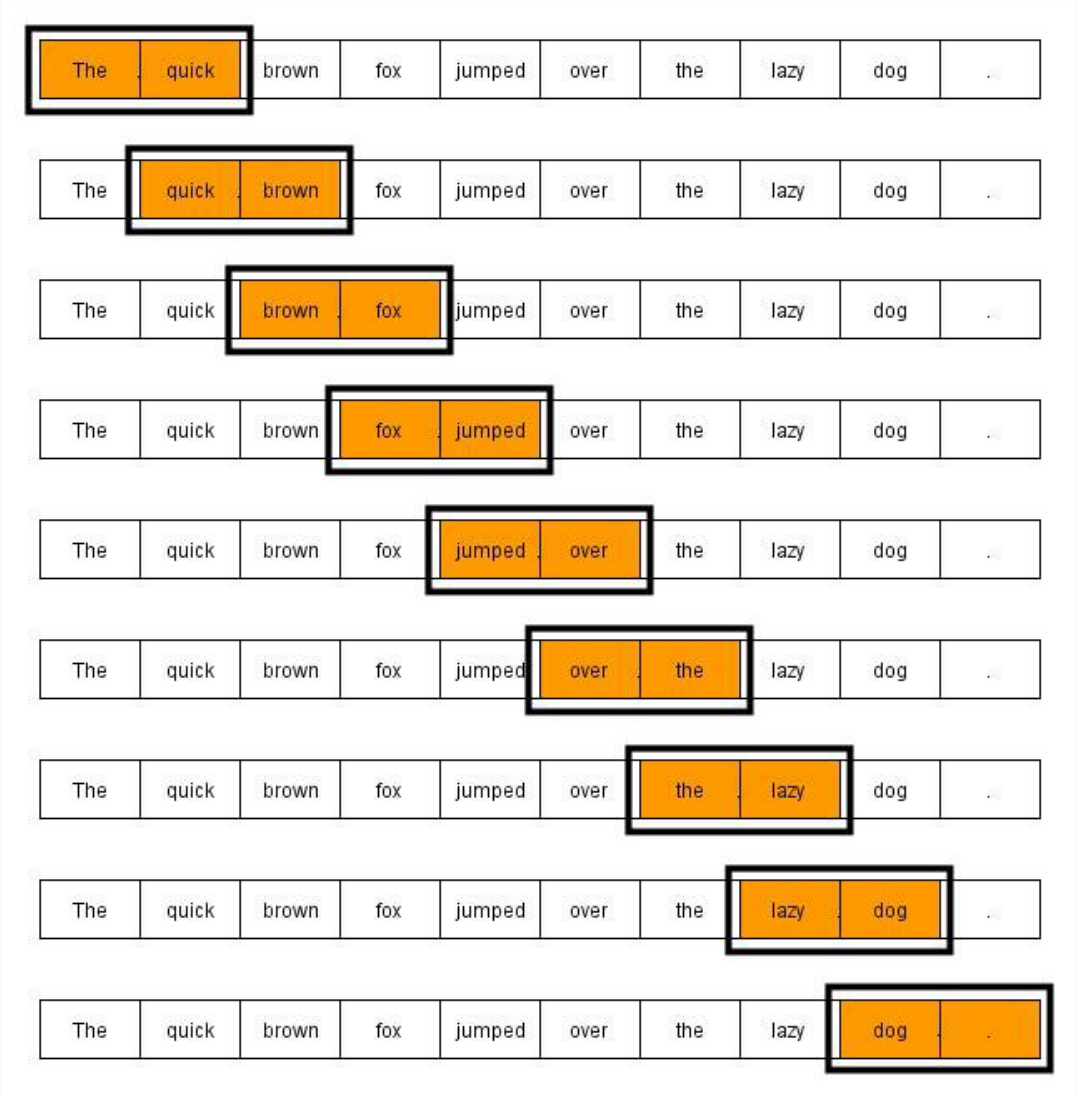
ML@B

- **Language modeling** is the problem of predicting the next word or letter given the previous words/letter.
- Mathematically, it is predicting

$$P(W_n = w_n | W_{n-1} = w_{n-1}, W_{n-2} = w_{n-2}, \ldots)$$

- We can train an RNN to perform language modeling by giving previous words/letters as input, and outputting the next word/letter

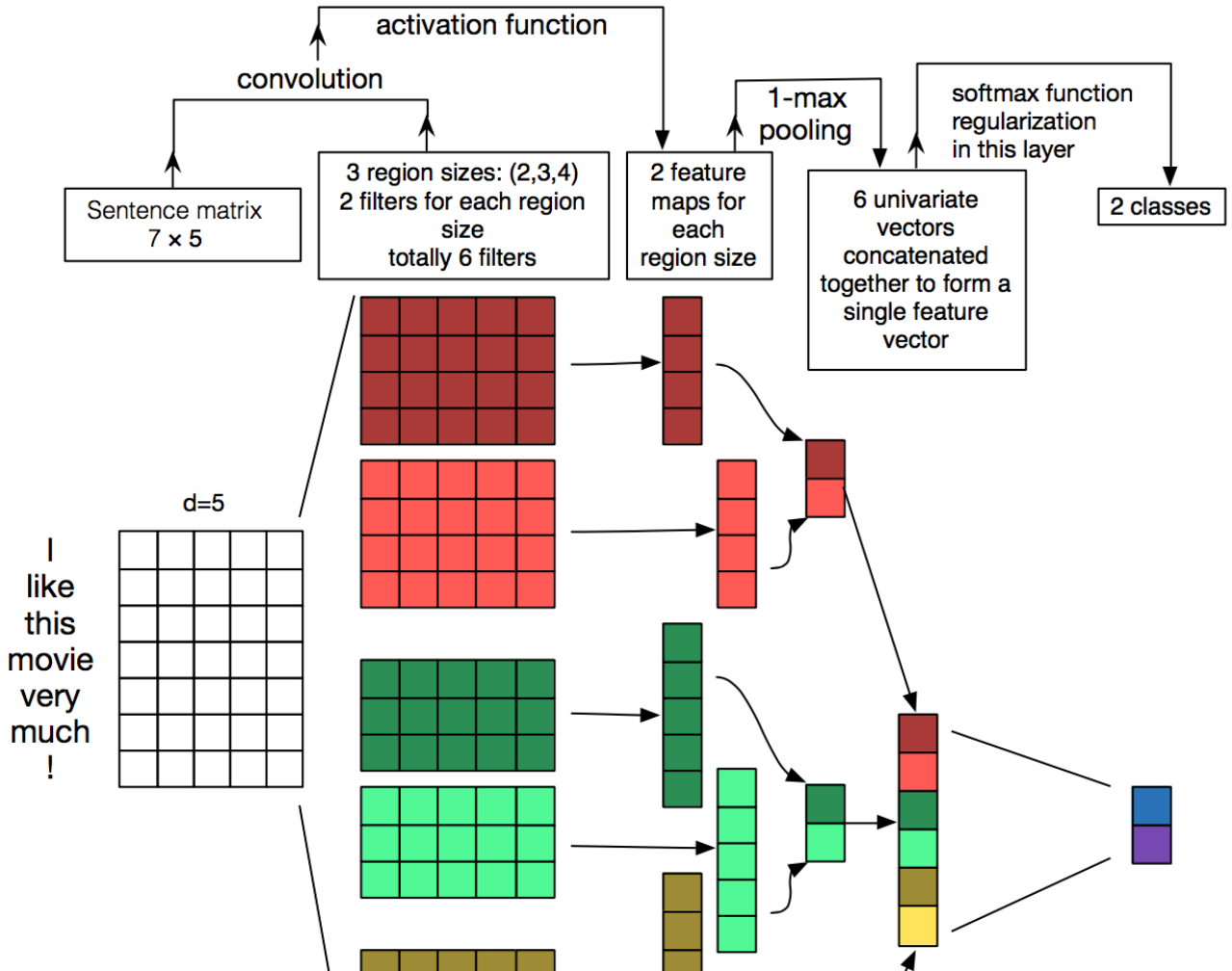
Using a Sliding Window for Input



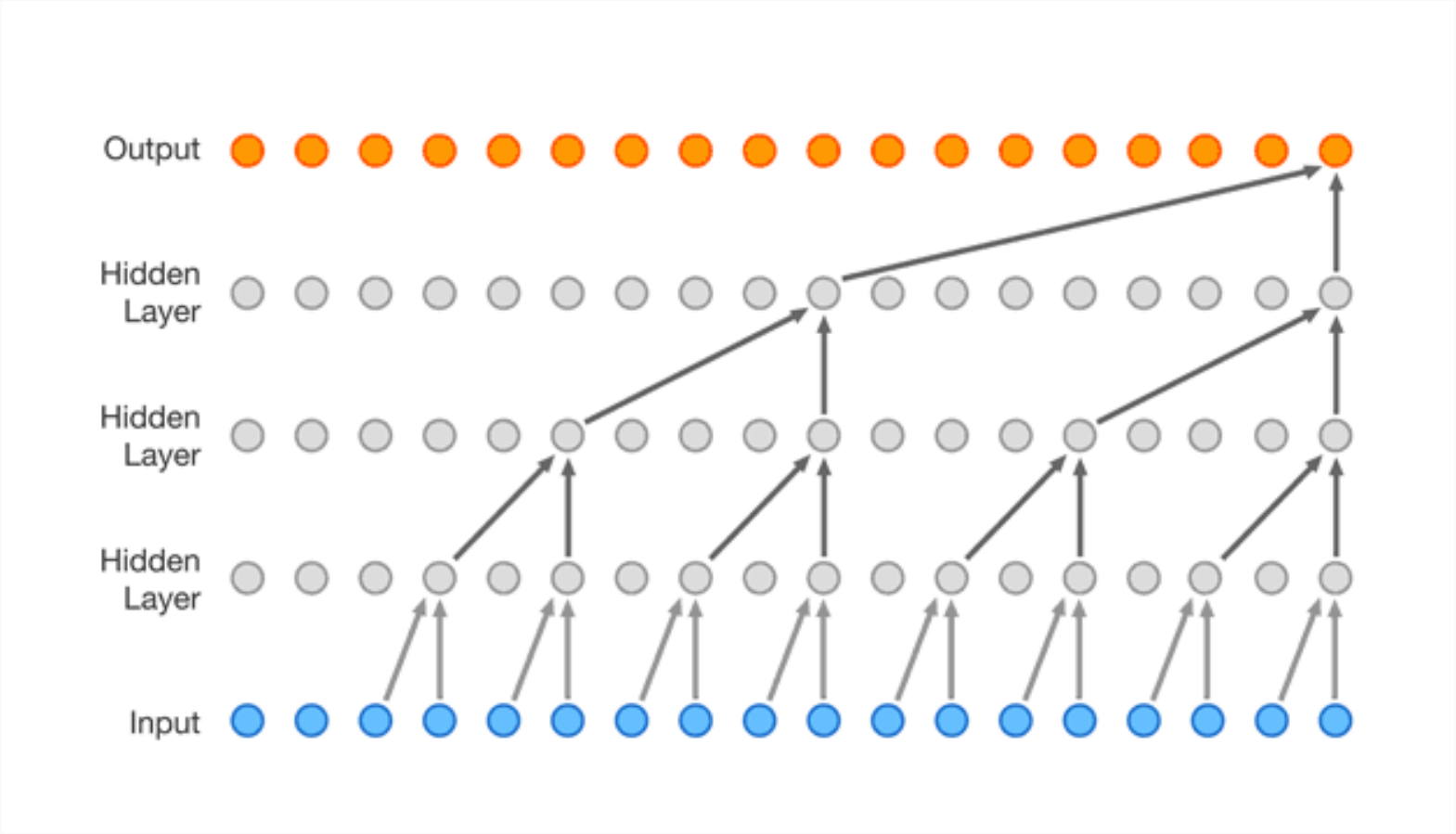
RNNs vs CNNs

Comparings RNNs and CNNs

Oftentimes, you can use CNNs to perform the same tasks as RNNs with equal or better results



Wavenet



Questions

Questions?