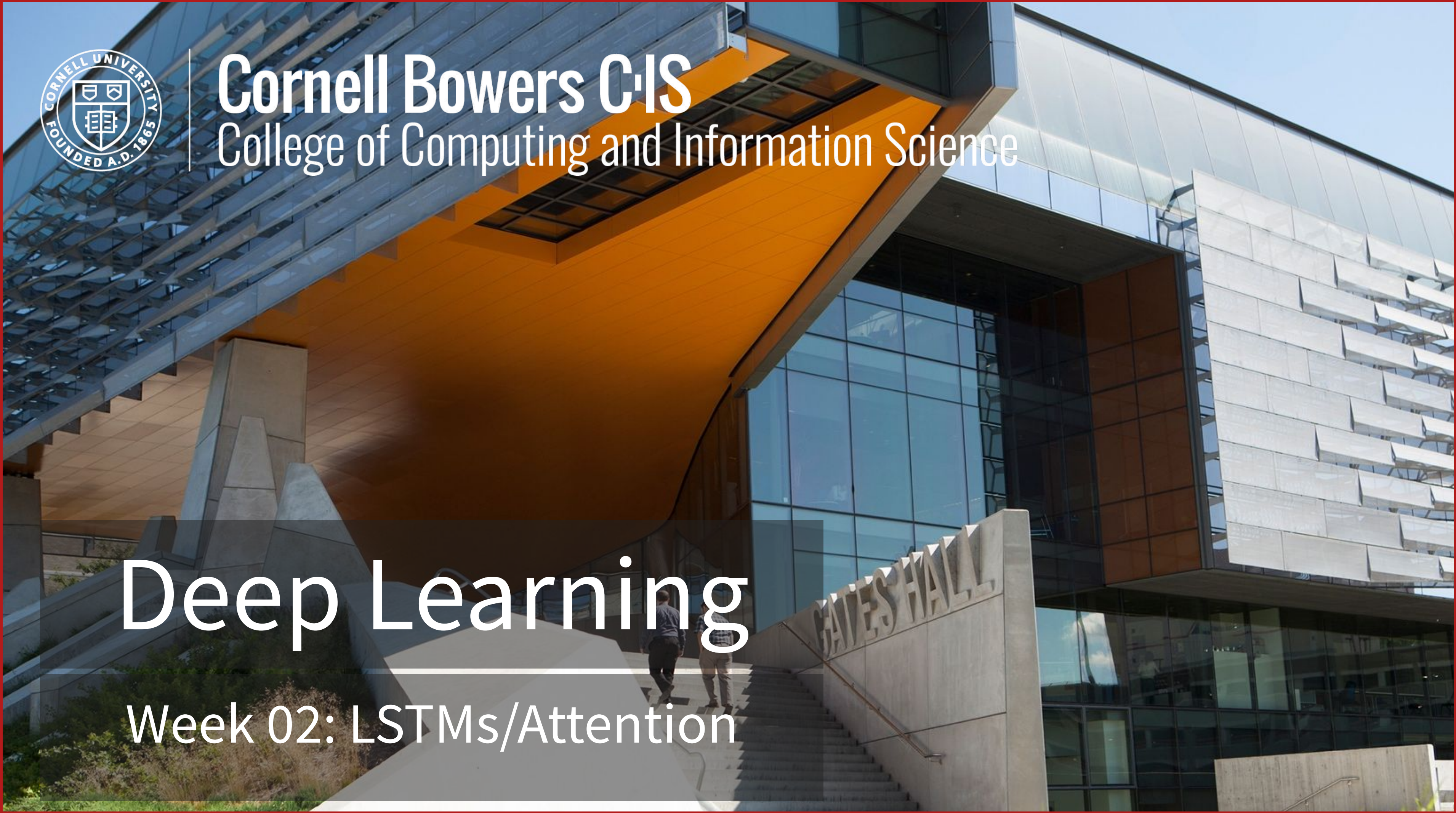
Cornell Bowers C·IS
College of Computing and Information Science

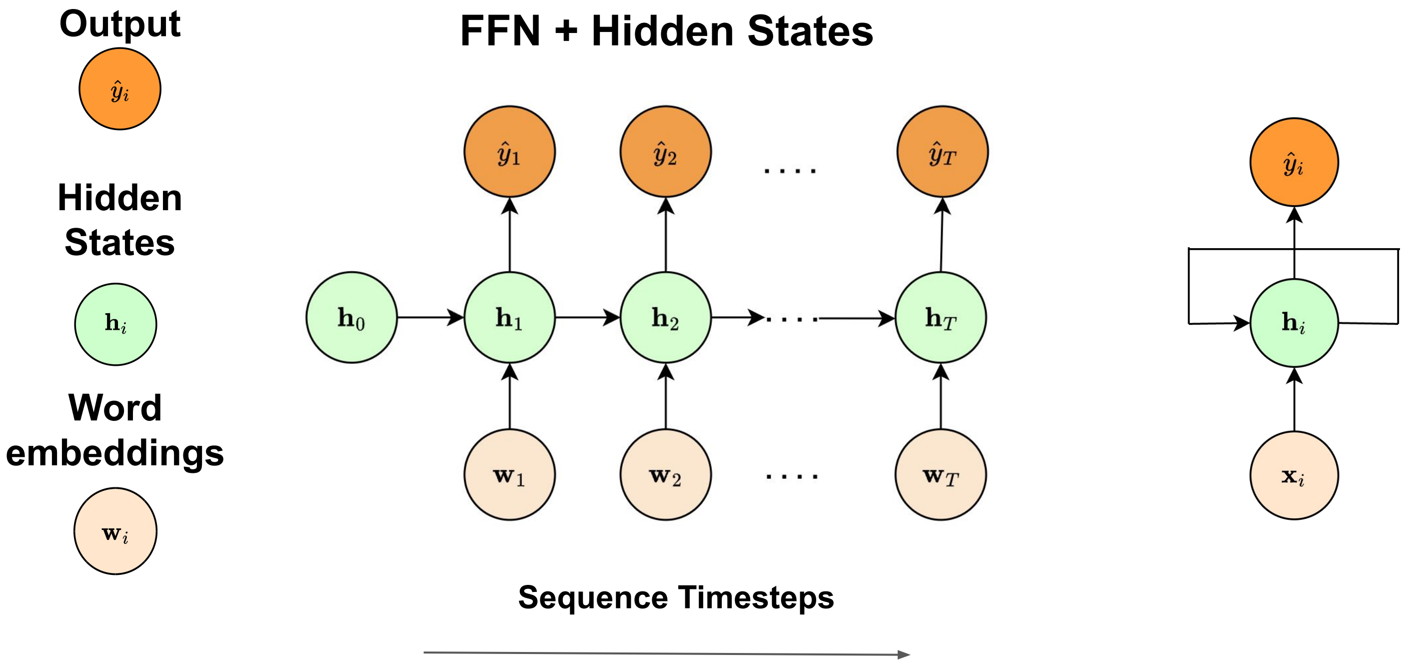
Deep Learning

Week 02: LSTMs/Attention



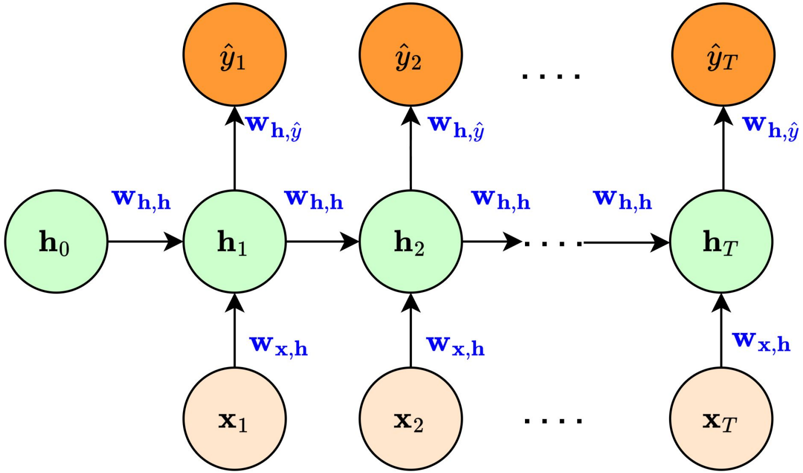
Big Question: How to model sequences of words?

Recurrent neural network (RNN)



RNN w/ parameter-sharing

Use **the same parameters** across different timesteps.



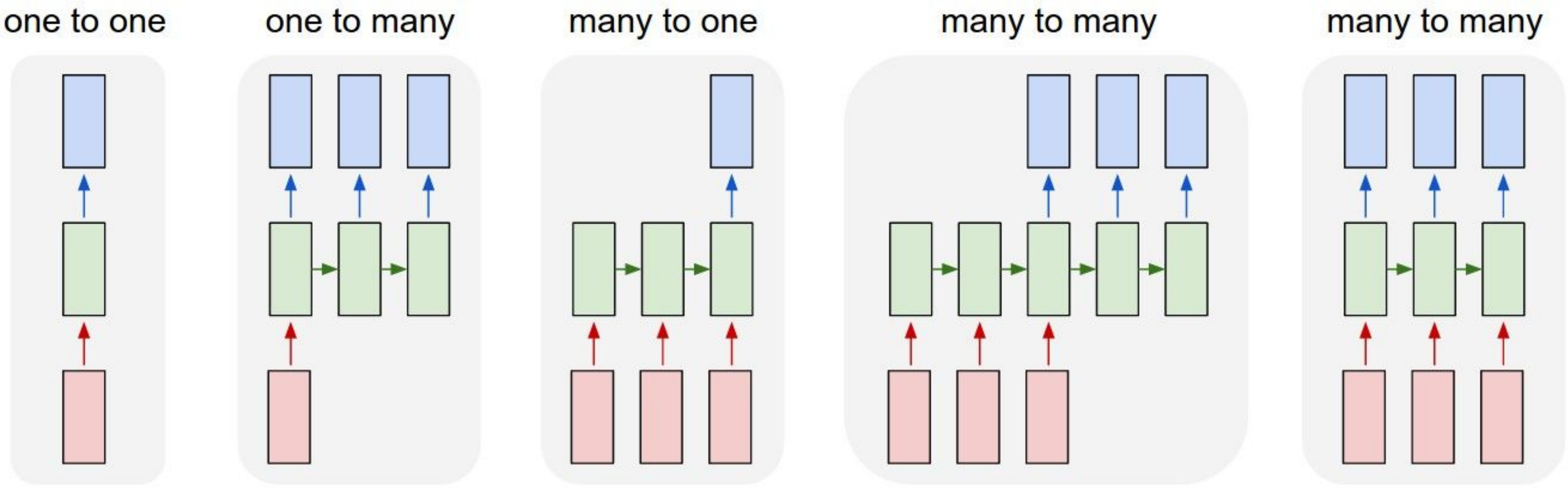
Hidden State

$$\begin{aligned}\mathbf{h}_i &= f(\mathbf{x}_i, \mathbf{h}_{i-1}) \\ &= \sigma(\mathbf{w}_{\mathbf{x},h}\mathbf{x}_i + \mathbf{w}_{\mathbf{h},\mathbf{h}}\mathbf{h}_{i-1})\end{aligned}$$

Output

$$\hat{y}_i = \mathbf{w}_{\mathbf{h},\hat{y}}\mathbf{h}_i$$

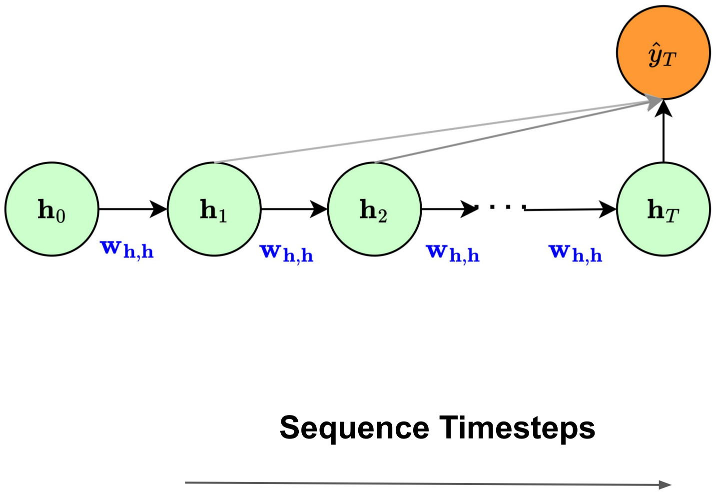
Discuss: What types of tasks can you perform with RNNs?



https://www.analyticsvidhya.com/blog/2021/06/time-series-analysis-recurrence-neural-network-in-python/

Cornell Bowers C·IS

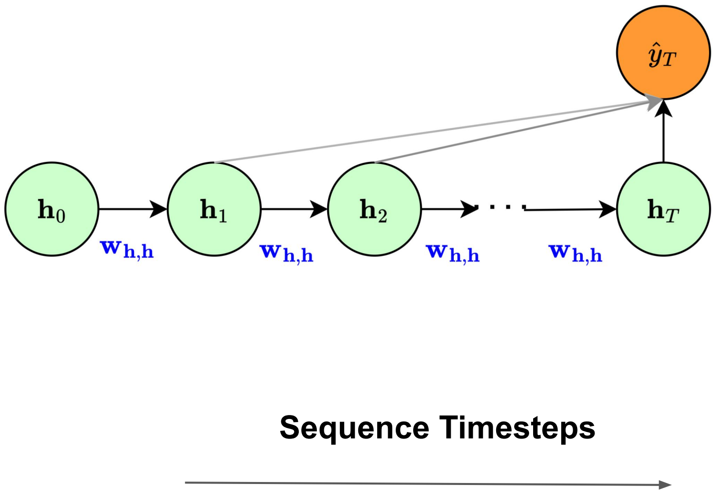
RNN: Issues under **Looooooooong** Context

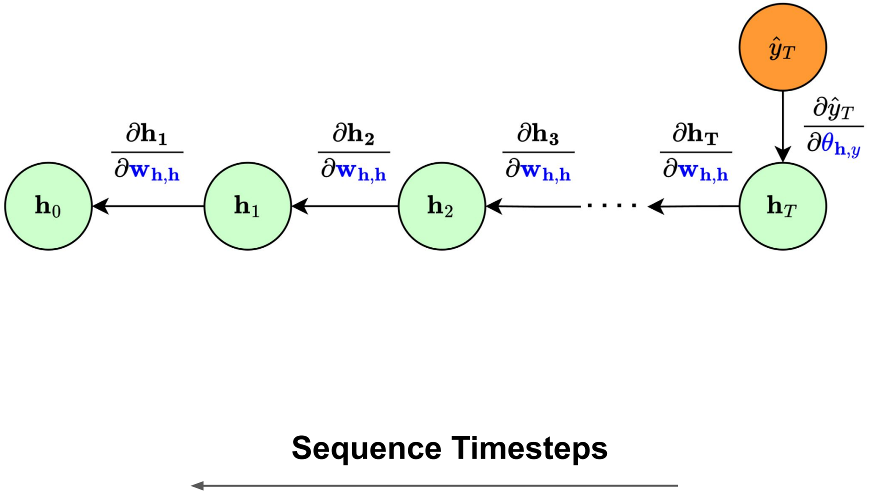


- Recurrent forward will **rewrite** the hidden states on every timestep!
  - What will happen? Let's discuss!

Cornell Bowers C·IS

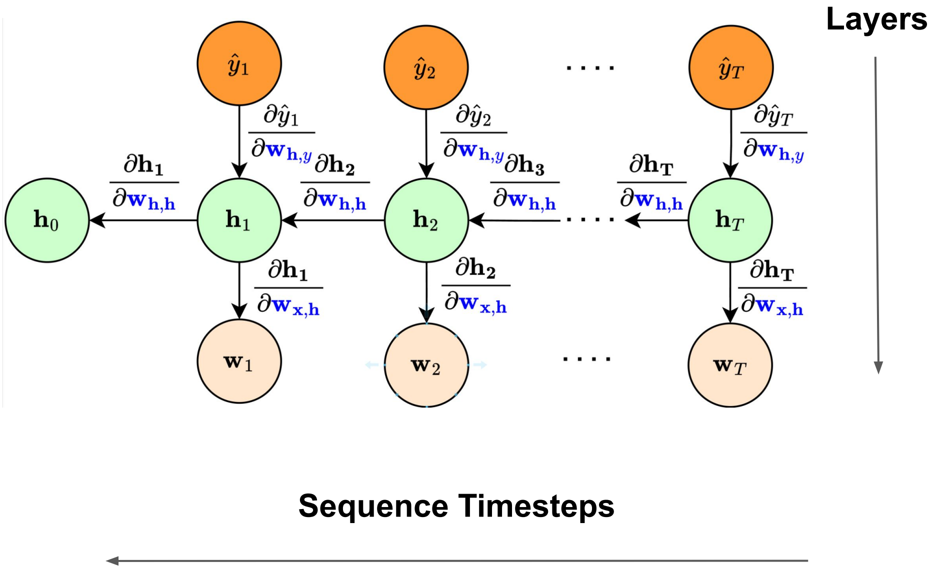
RNN: Issues under **Looooooooong** Context





Cornell Bowers C·IS

Backpropagation through the Time (BPTT)



- Unfold a recurrent neural network in time
- Gradients are accumulated across all time steps by applying the chain rule
- Propagate gradients backwards through time steps

Cornell Bowers CIS

Backpropagation through the Time (BPTT)

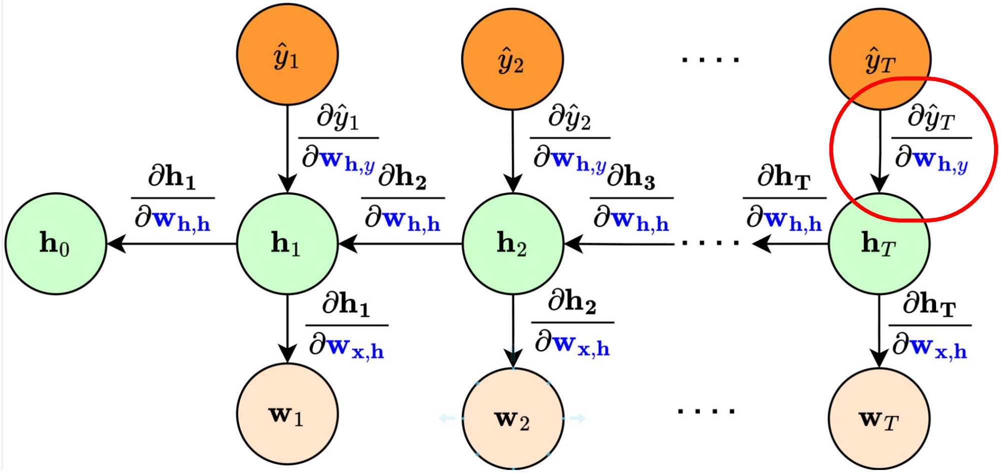
Hidden State

$$\mathbf{h}_i = f(\mathbf{x}_i, \mathbf{h}_{i-1})$$
$$= \sigma(\mathbf{w}_{\mathbf{x},h}\mathbf{x} + \mathbf{w}_{\mathbf{h},\mathbf{h}}\mathbf{h}_{i-1})$$

Output

$$\hat{y}_i = \mathbf{w}_{\mathbf{h},\hat{y}}\mathbf{h}_i$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\mathbf{h},\hat{y}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{w}_{\mathbf{h},\hat{y}}}$$



Cornell Bowers C·IS

Backpropagation through the Time (BPTT)

Hidden State

$$\mathbf{h}_i = f(\mathbf{x}_i, \mathbf{h}_{i-1})$$
$$= \sigma(\mathbf{w}_{\mathbf{x},h}\mathbf{x} + \mathbf{w}_{\mathbf{h},\mathbf{h}}\mathbf{h}_{i-1})$$

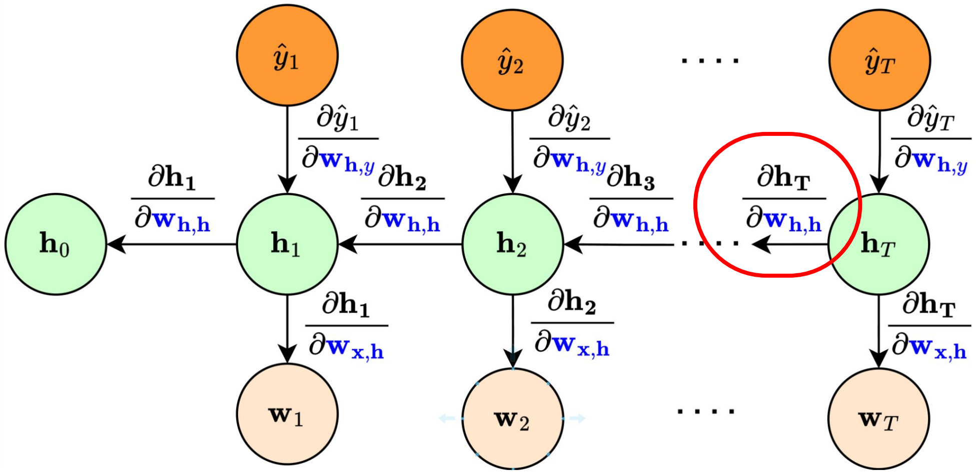
Output

$$\hat{y}_i = \mathbf{w}_{\mathbf{h},\hat{y}}\mathbf{h}_i$$

Assume we only compute the loss on the last time step

Last time step:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\mathbf{h},\mathbf{h}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{w}_{h,h}}$$



Backpropagation through the Time (BPTT)

Hidden State

$$\mathbf{h}_i = f(\mathbf{x}_i, \mathbf{h}_{i-1}) = \sigma(\mathbf{w}_{\mathbf{x},h}\mathbf{x} + \mathbf{w}_{\mathbf{h},\mathbf{h}}\mathbf{h}_{i-1})$$

Output

$$\hat{y}_i = \mathbf{w}_{\mathbf{h},\hat{y}}\mathbf{h}_i$$

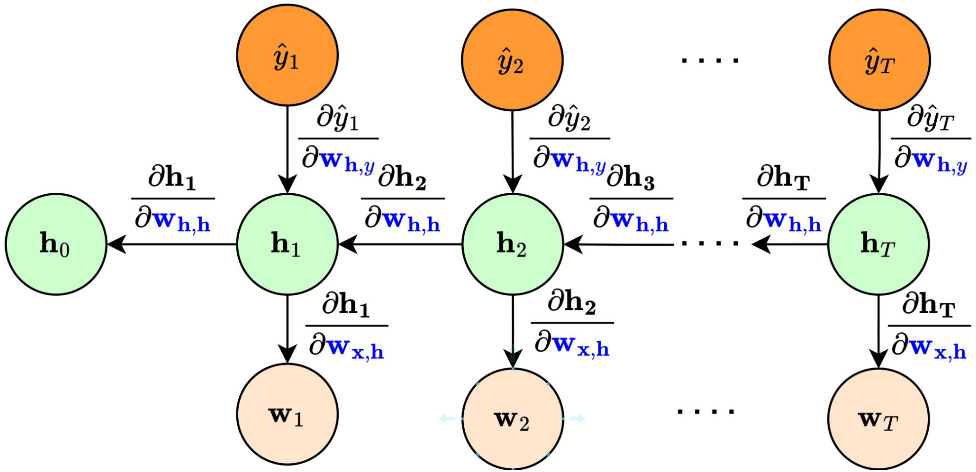
Assume we only compute the loss on the last time step

Last time step:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w_{h,h}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{w}_{h,h}}$$

T-1th time step:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w_{h,h}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_{T-1}} \cdot \frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{w}_{h,h}}$$



Cornell Bowers CIS

Backpropagation through the Time (BPTT)

Hidden State

$$\mathbf{h}_i = f(\mathbf{x}_i, \mathbf{h}_{i-1})$$
$$= \sigma(\mathbf{w}_{\mathbf{x},h}\mathbf{x} + \mathbf{w}_{\mathbf{h},\mathbf{h}}\mathbf{h}_{i-1})$$

Output

$$\hat{y}_i = \mathbf{w}_{\mathbf{h},\hat{y}}\mathbf{h}_i$$

Assume we only compute the loss on the last time step

Last time step:

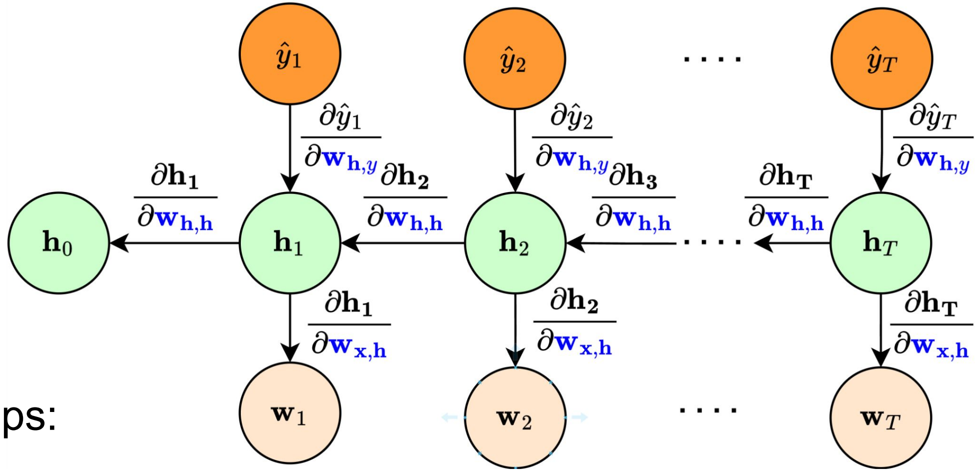
$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\mathbf{h},\mathbf{h}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{w}_{h,h}}$$

T-1th time step:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\mathbf{h},\mathbf{h}}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_{T-1}} \cdot \frac{\partial \mathbf{h}_{T-1}}{\partial \mathbf{w}_{h,h}}$$

Generalizing and summing over all time steps:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}_{\mathbf{h},\mathbf{h}}} = \sum_{k=1}^{T} \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_k} \cdot \frac{\partial \mathbf{h}_k}{\partial \mathbf{w}_{h,h}}$$
$$= \sum_{k=1}^{T} \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \cdot \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \cdot \left(\prod_{j=k}^{T-1} \frac{\partial \mathbf{h}_{j+1}}{\partial \mathbf{h}_j}\right) \cdot \frac{\partial \mathbf{h}_k}{\partial \mathbf{w}_{h,h}}$$



Cornell Bowers C·IS

RNN: Issues under **Looooooooong** Context

$$\frac{\partial \mathcal{L}(\hat{y}_T)}{\partial \mathbf{h}_1} = \frac{\partial \mathcal{L}(\hat{y}_T)}{\partial \mathbf{h}_T} \prod_{1<t\leq T} \frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}}$$

$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \mathrm{diag}\left(\sigma^{'}(\mathbf{w}_{\mathbf{h},\mathbf{h}}\mathbf{h}_{t-1} + \mathbf{w}_{\mathbf{h},\mathbf{x}}\mathbf{x}_t)\right)\boxed{\mathbf{w}_{\mathbf{h},\mathbf{h}}}$$

- **Vanishing gradients: grad to *0***

If $\|\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}\| < 1$ and T is large, $\|\frac{\partial \mathcal{L}(\hat{y}_T)}{\partial \mathbf{h}_1}\| \to 0.$

- **Exploding gradients: grad to *inf***

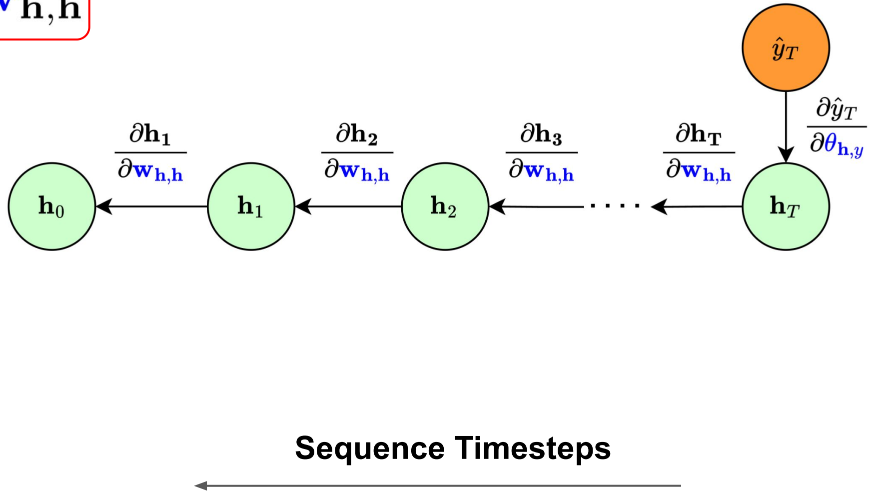
If $\|\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}\| > 1$ and T is large, $\|\frac{\partial \mathcal{L}(\hat{y}_T)}{\partial \mathbf{h}_1}\| \to \mathrm{inf}.$

Hidden State

$$\mathbf{h}_i = f(\mathbf{x}_i, \mathbf{h}_{i-1})$$
$$= \sigma(\mathbf{w}_{\mathbf{x},h}\mathbf{x} + \mathbf{w}_{\mathbf{h},\mathbf{h}}\mathbf{h}_{i-1})$$

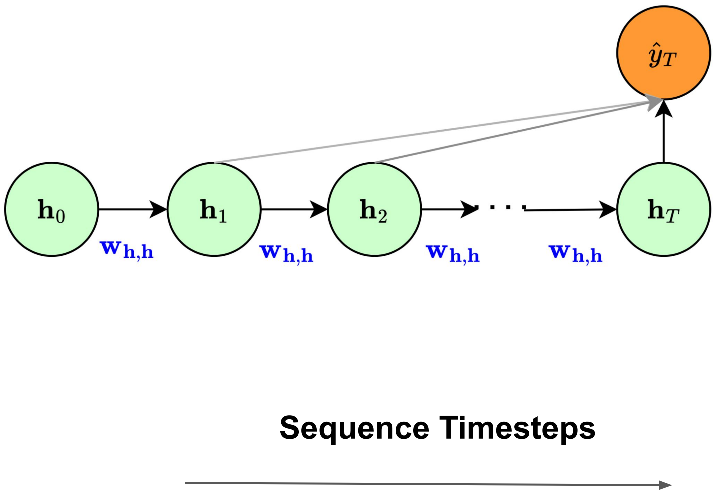
Output

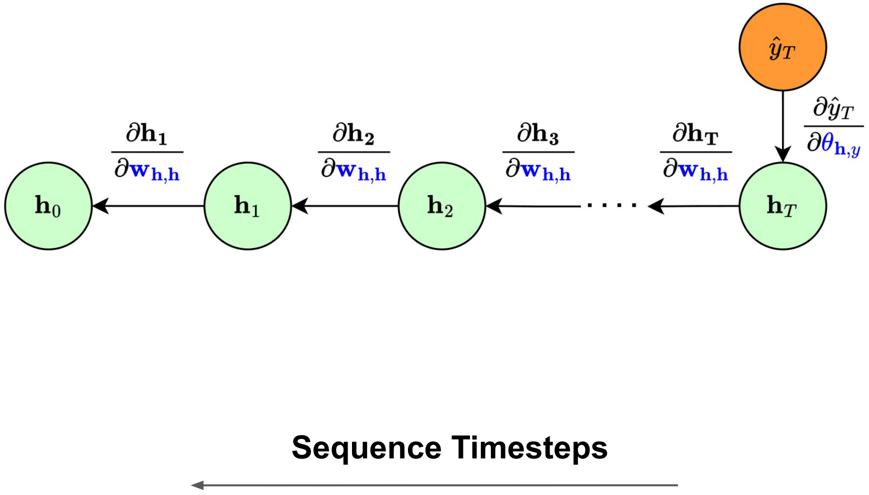
$$\hat{y}_i = \mathbf{w}_{\mathbf{h},\hat{y}}\mathbf{h}_i$$



Cornell Bowers C·IS

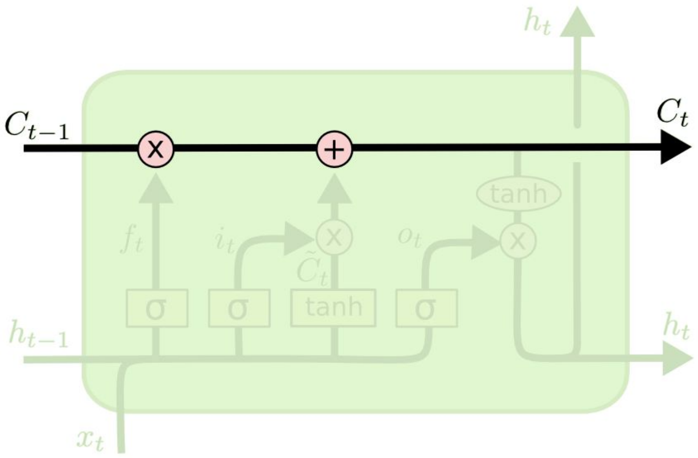
RNN: Issues under **Looooooooong** Context





Long-short Term Memory (LSTM)

- Main idea: add a “cell” state that allows information to flow easily
  - Similar to residual connections
  - No repeated matrix multiplications!



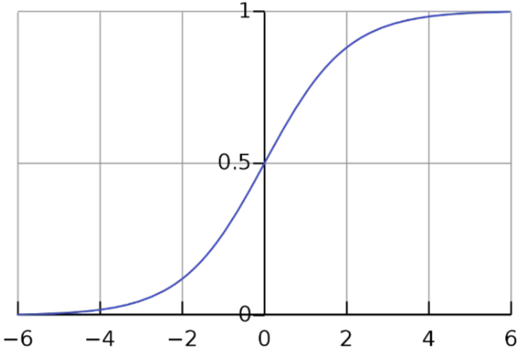
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

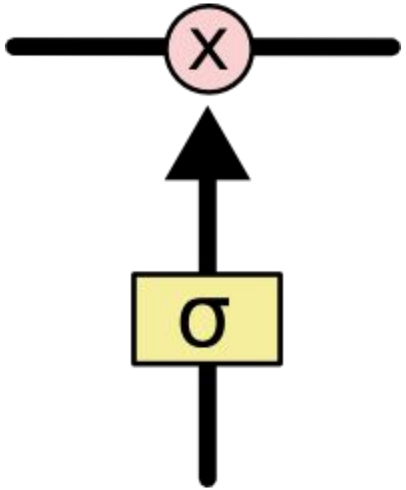
LSTMs- Gates

- Control the flow of information with “gates”
  - Element-wise product with the output of a sigmoid activation

Sigmoid Function

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

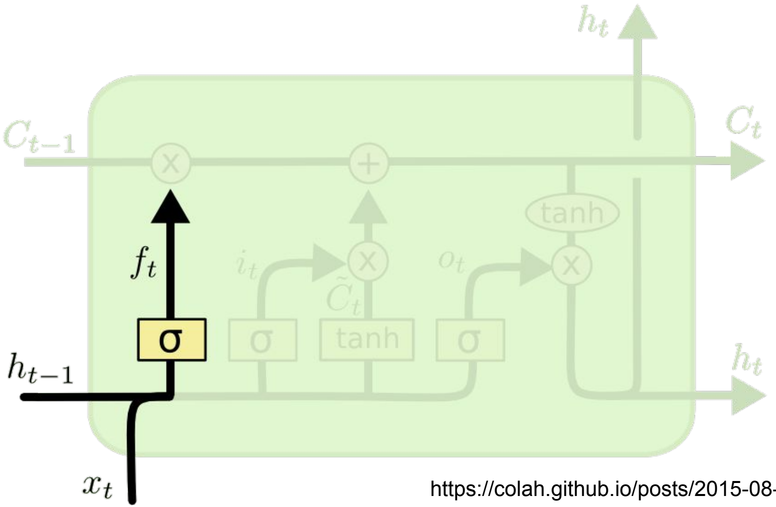




https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs- Forget Gate

- Forget gate- function of current input and previous hidden state
- Controls what should be remembered in the cell state

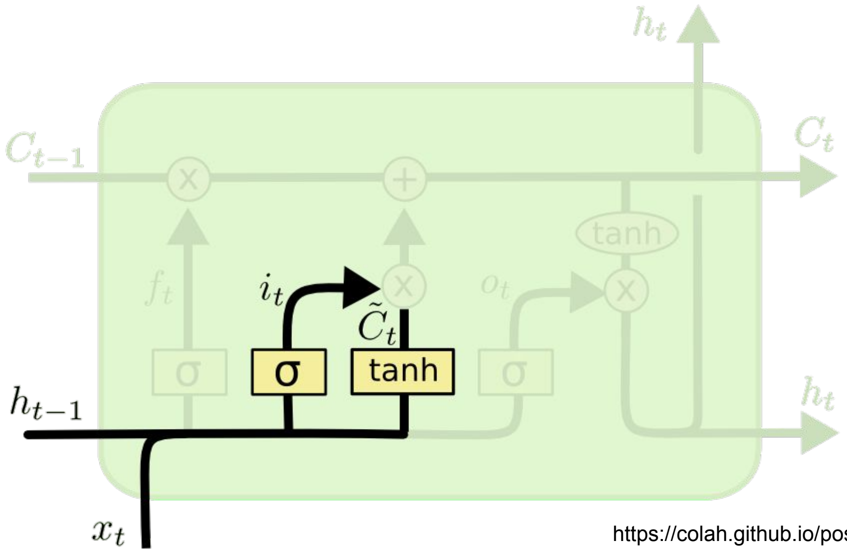


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs- Input Gate

- Input gate- function of current input and previous hidden state
- Decides what information to write to the cell state



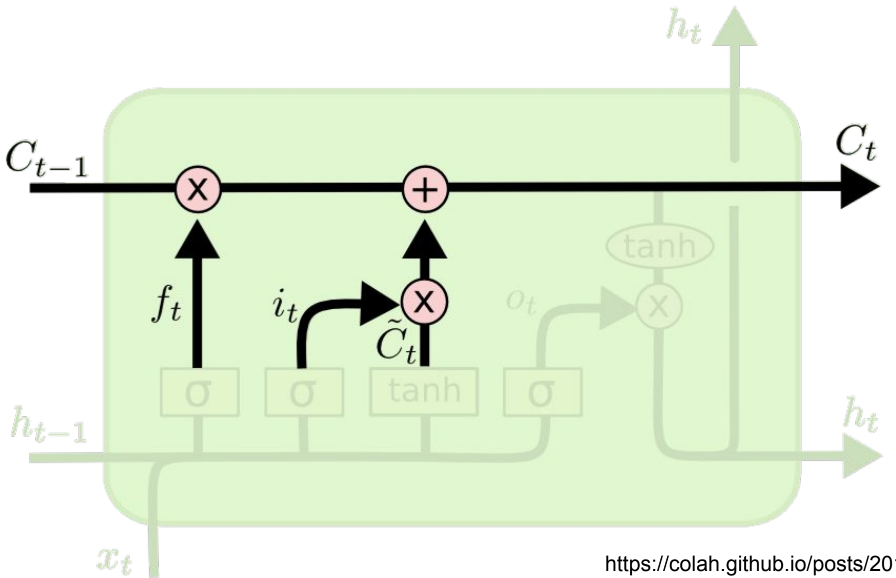
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM- Cell Update

- Forget irrelevant information
- Add new information from the current token

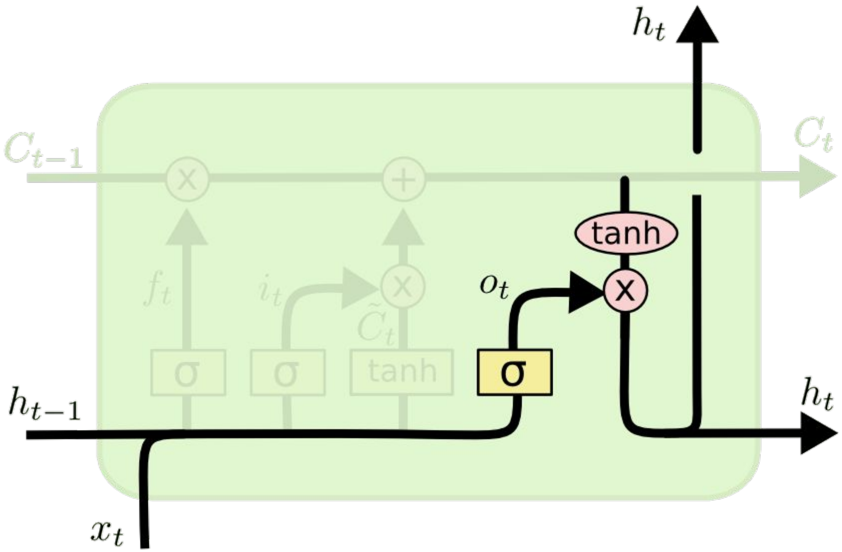


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTM- Output Gate

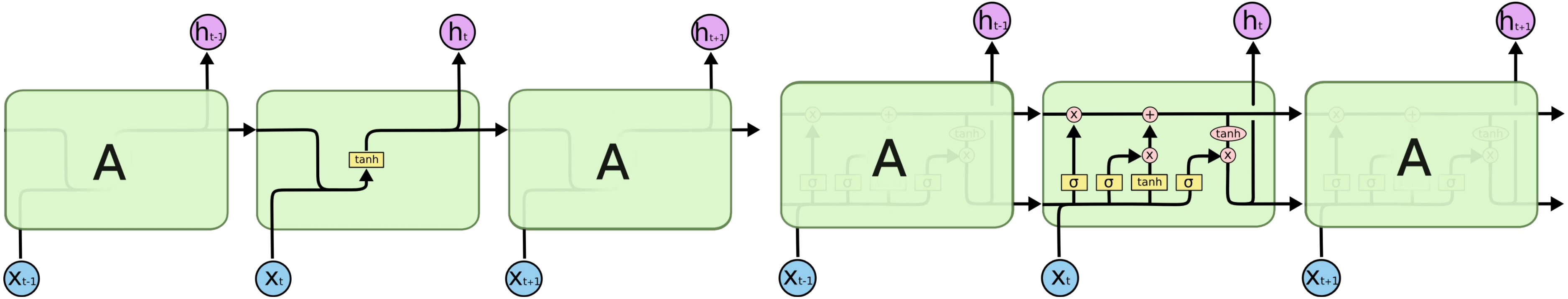
- Output gate- function of current input and previous hidden state
- Controls flow of information from the cell state to the hidden state
- Discuss: Given some weight matrix W_o, how could we write the update?



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

RNN vs. LSTM

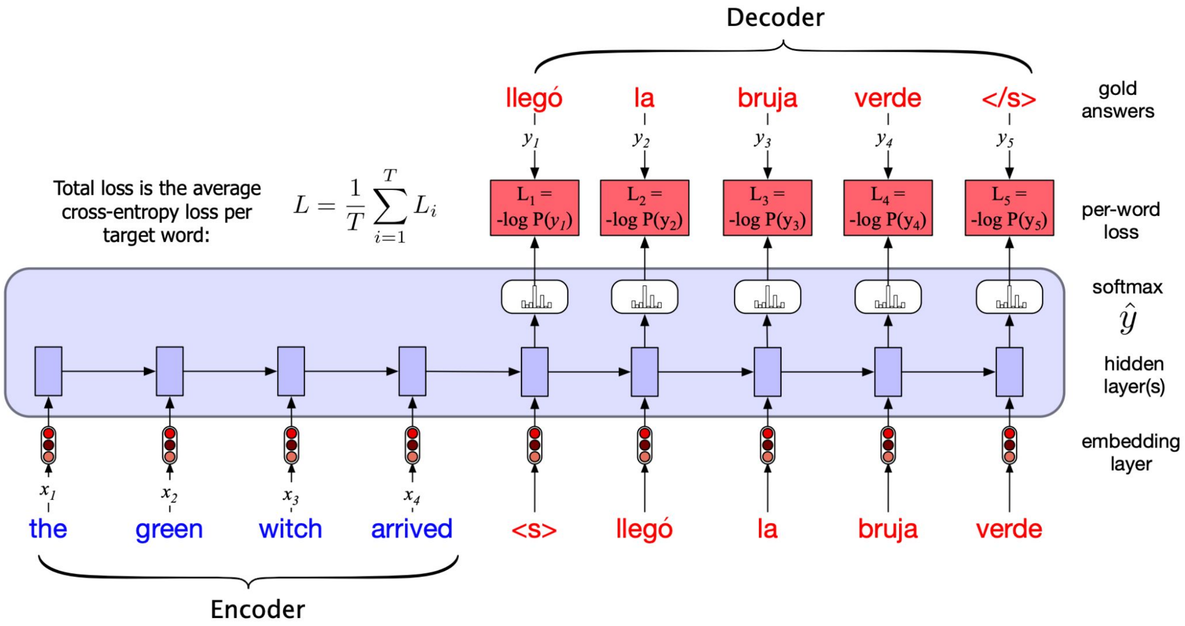
- **RNN**
  - Can be applied to variable-length sequences
  - Share parameters across time
  - Hard to train!
- **LSTM**
  - Mitigates the vanishing gradient problem with the cell state
  - Better for long sequences



https://colah.github.io/posts/2015-08-Understanding-LSTMs/

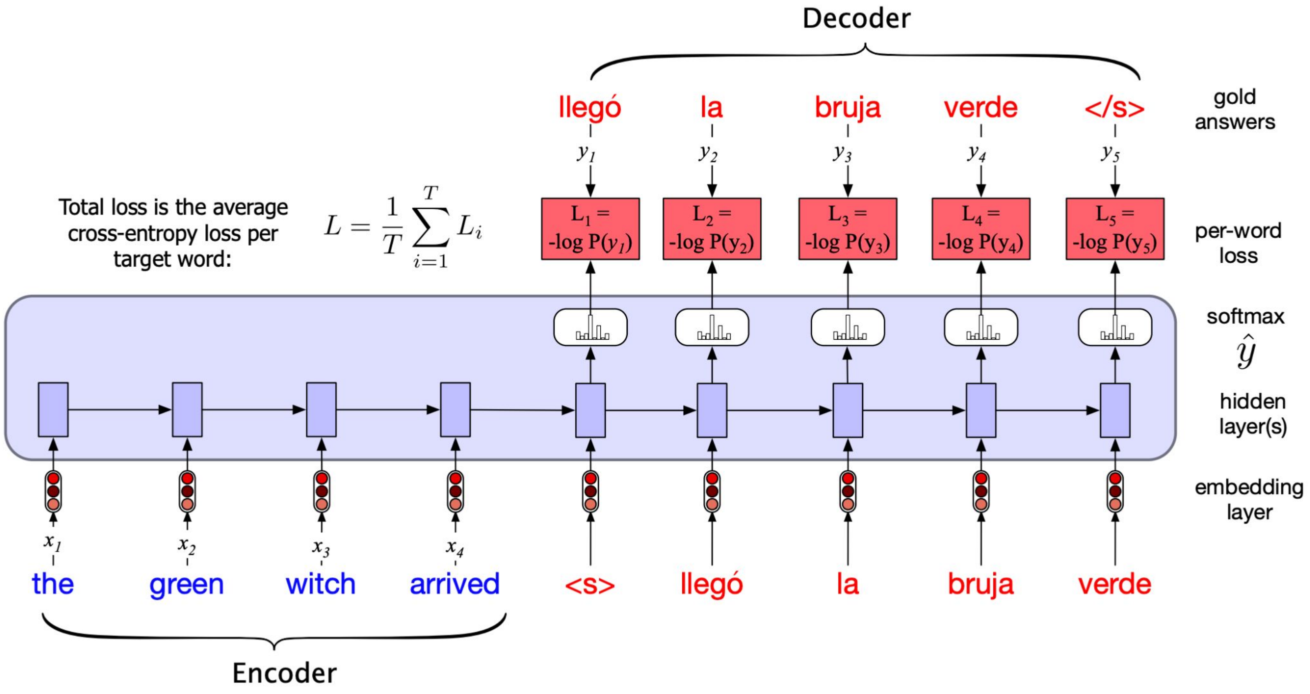
Sequence-to-Sequence Generation

- Map some input sequence to a target sequence
- Applications:
    - Machine translation
    - News summarization
    - ChatGPT!



https://web.stanford.edu/~jurafsky/slp3/

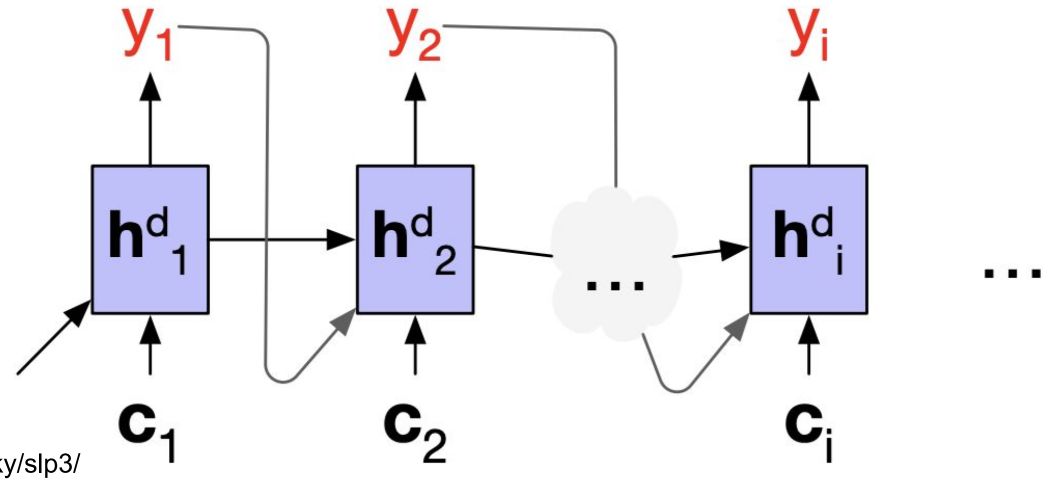
Discuss: Potential problems with Sequence to Sequence Models



https://web.stanford.edu/~jurafsky/slp3/

Attention

- Attention gives the network a way to “look back” at all previous hidden states
  - Introduced to handle long source sentences in neural machine translation (NMT)



https://web.stanford.edu/~jurafsky/slp3/

Attention Mechanism

Consists of 3 "general" steps:

Step 1: Compute score of each embedding/input

$$\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i)$$

Step 2: Compute attention weights according to alignment with outputs (general attention) or inputs (self attention)

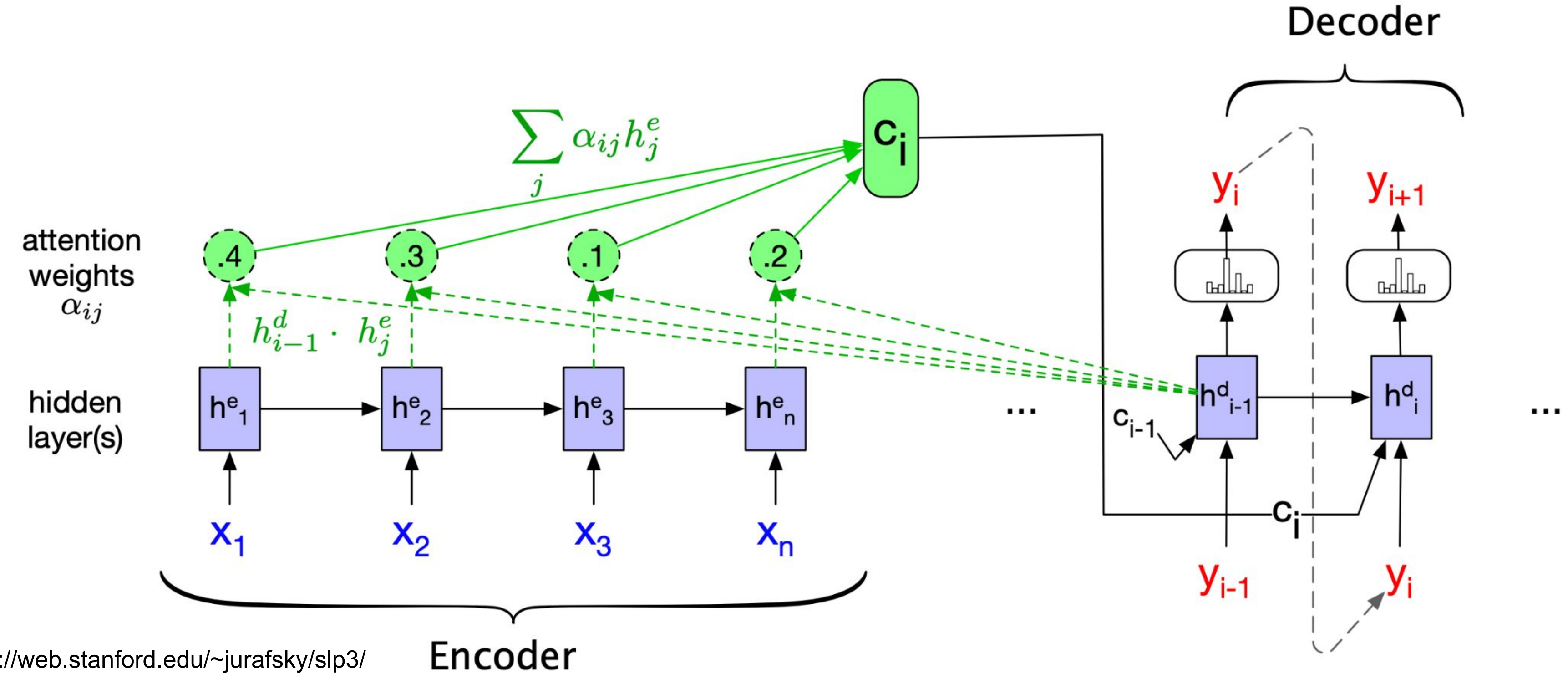
$$\begin{aligned}\alpha_{t,i} &= \text{align}(y_t, x_i) \\ &= \frac{\exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_i))}{\sum_{i'=1}^{n} \exp(\text{score}(\boldsymbol{s}_{t-1}, \boldsymbol{h}_{i'}))}\end{aligned}$$

Step 3: Compute the context vector, scaled according to attention weights

$$\mathbf{c}_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$

https://lilianweng.github.io/posts/2018-06-24-attention/

Attention Mechanism



https://web.stanford.edu/~jurafsky/slp3/

Popular Attention Formulations

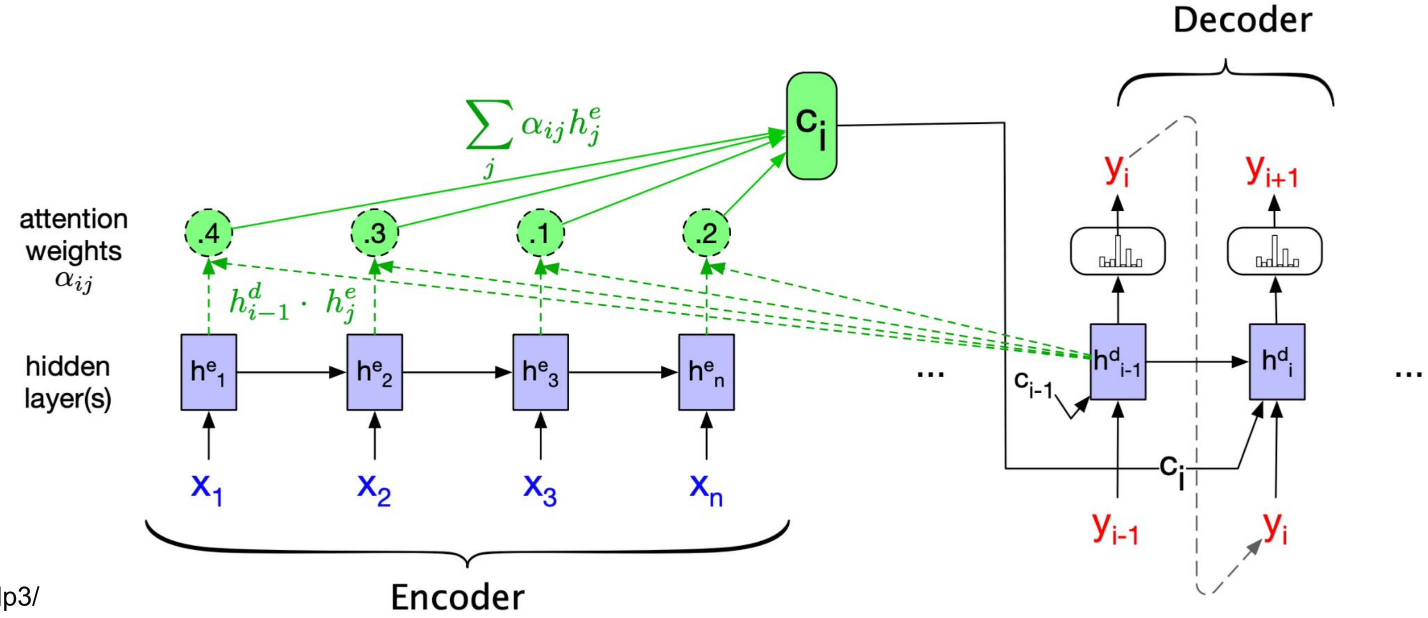
- Different score functions have been introduced
  - In practice, the dot-product is simple and effective

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_{t-1}; \boldsymbol{h}_i])$	Bahdanau2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

https://lilianweng.github.io/posts/2018-06-24-attention/

Attention

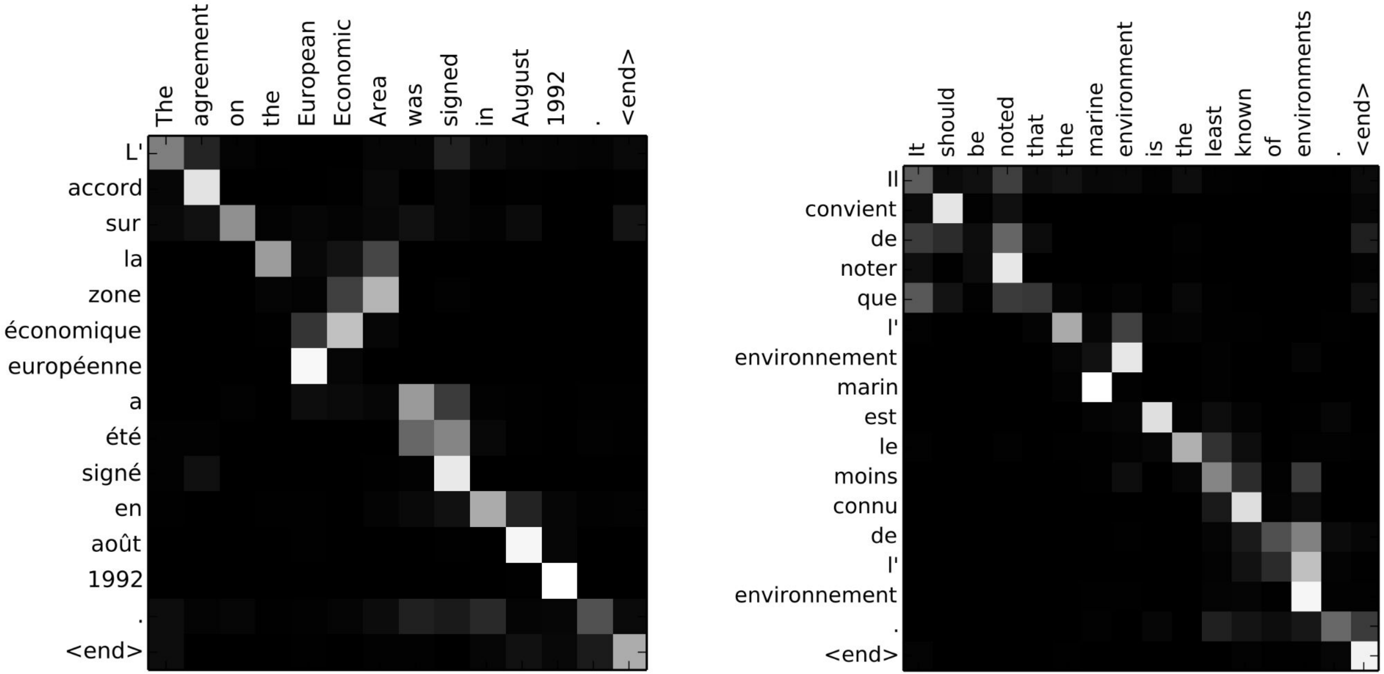
- Computes time-dependent weighted averages over previous vectors
- Can focus on different aspects of the past sequence



https://web.stanford.edu/~jurafsky/slp3/

Visualizing Attention

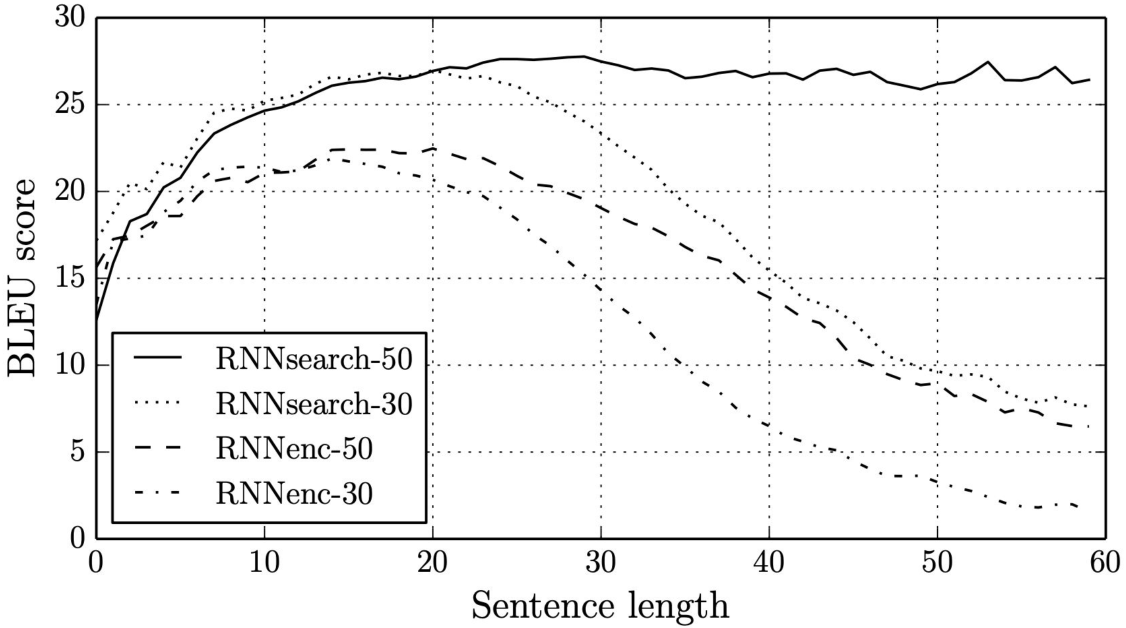
- Plot attention weights to see where the model is “looking”
  - Learns language alignment for translation!



Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate."

Impact of Attention

- Really helpful for long sequences
  - Helps solve bottleneck problem!

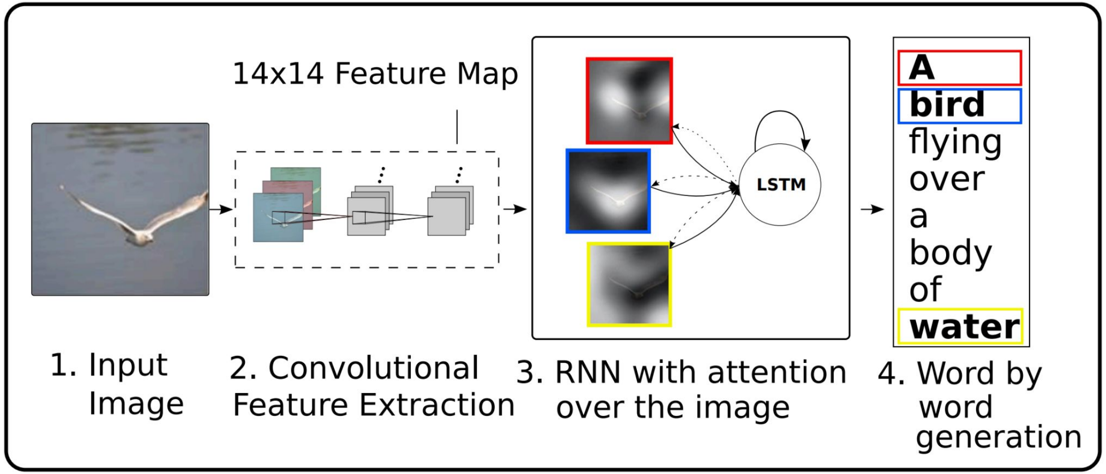


Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate."

Attention Application- Image Captioning!

- Extract image features with a CNN
- Use an LSTM with attention to generate image captions

Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4



Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International conference on machine learning. PMLR, 2015.

Visualize Attention Weights

- Learns to focus on relevant regions of the image

Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)

A woman is throwing a frisbee in a park.

A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

A little girl sitting on a bed with a teddy bear.

A group of people sitting on a boat in the water.

A giraffe standing in a forest with trees in the background.

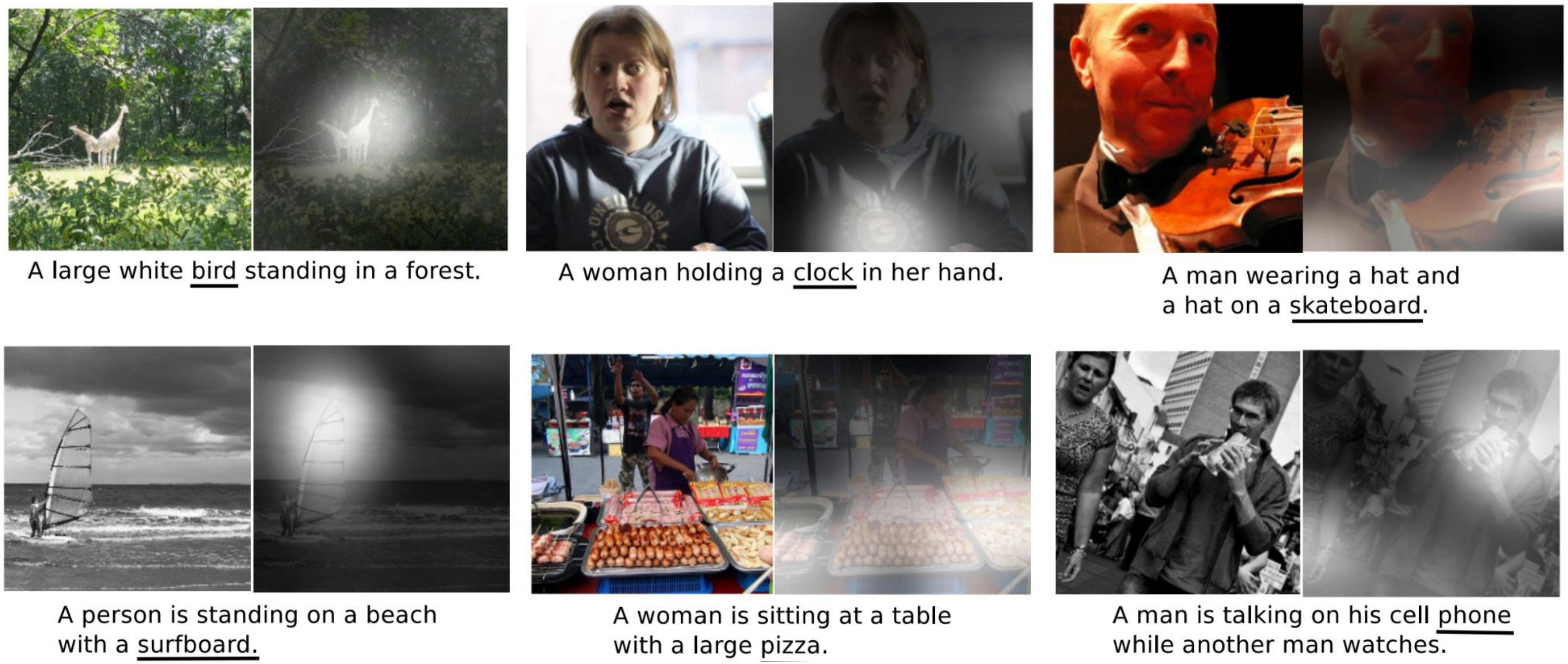
Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International conference on machine learning. PMLR, 2015.

(b) A woman is throwing a frisbee in a park.

Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International conference on machine learning. PMLR, 2015.

Error Analysis!

Figure 5. Examples of mistakes where we can use attention to gain intuition into what the model saw.



Xu, Kelvin, et al. "Show, attend and tell: Neural image caption generation with visual attention." International conference on machine learning. PMLR, 2015.

Recap

- RNNs can be applied to arbitrary length sequences
  - Run into vanishing/exploding gradient problems
- LSTMs add a cell state to RNNs to improve gradient flow
  - Better a handling long sequences
- Attention can look back at past feature vectors!
  - Scales better to long sequences
  - Can incorporate image features
  - Many, many more applications!