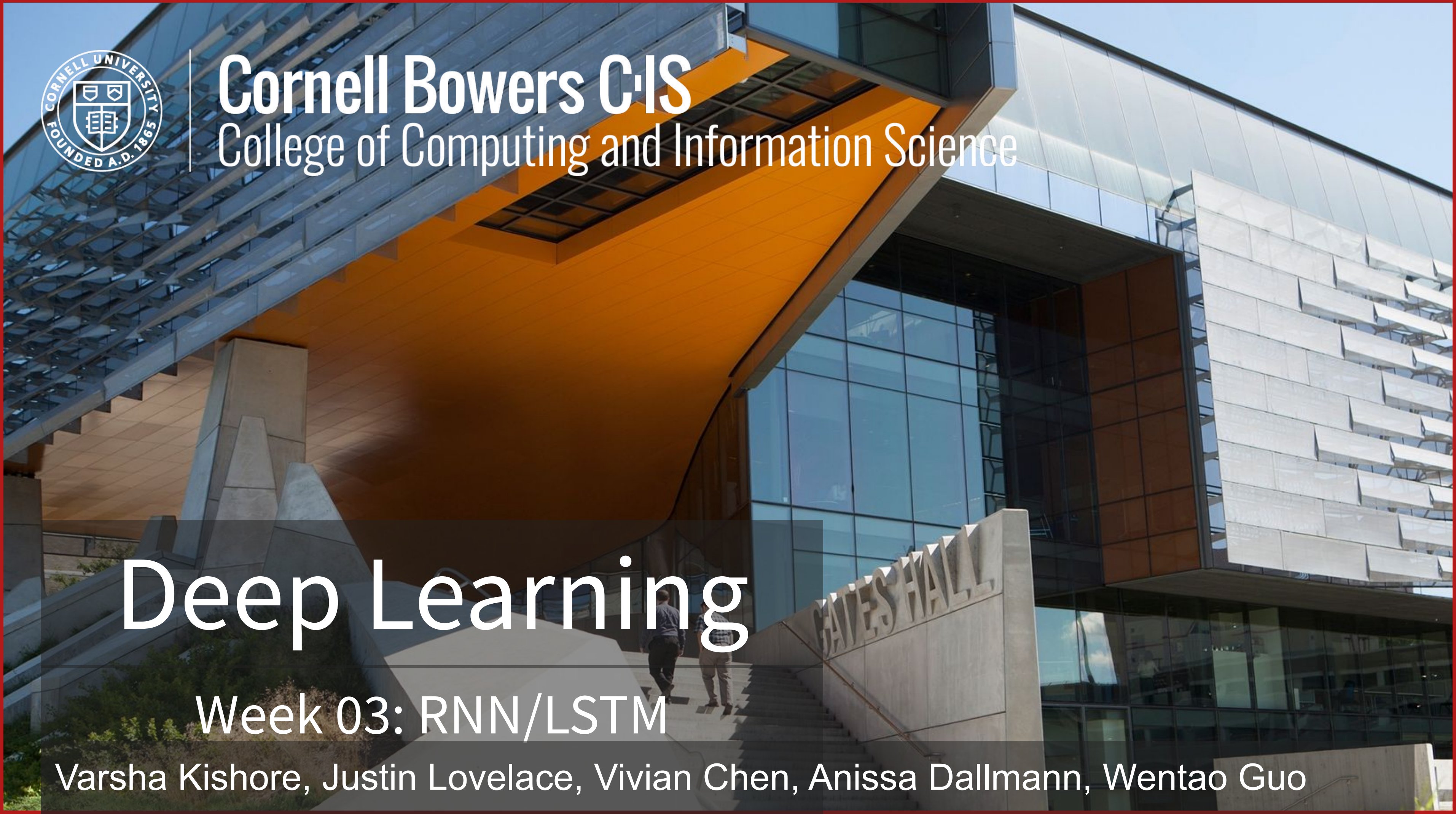


Cornell Bowers C·IS
College of Computing and Information Science

Deep Learning

Week 03: RNN/LSTM

Varsha Kishore, Justin Lovelace, Vivian Chen, Anissa Dallmann, Wentao Guo

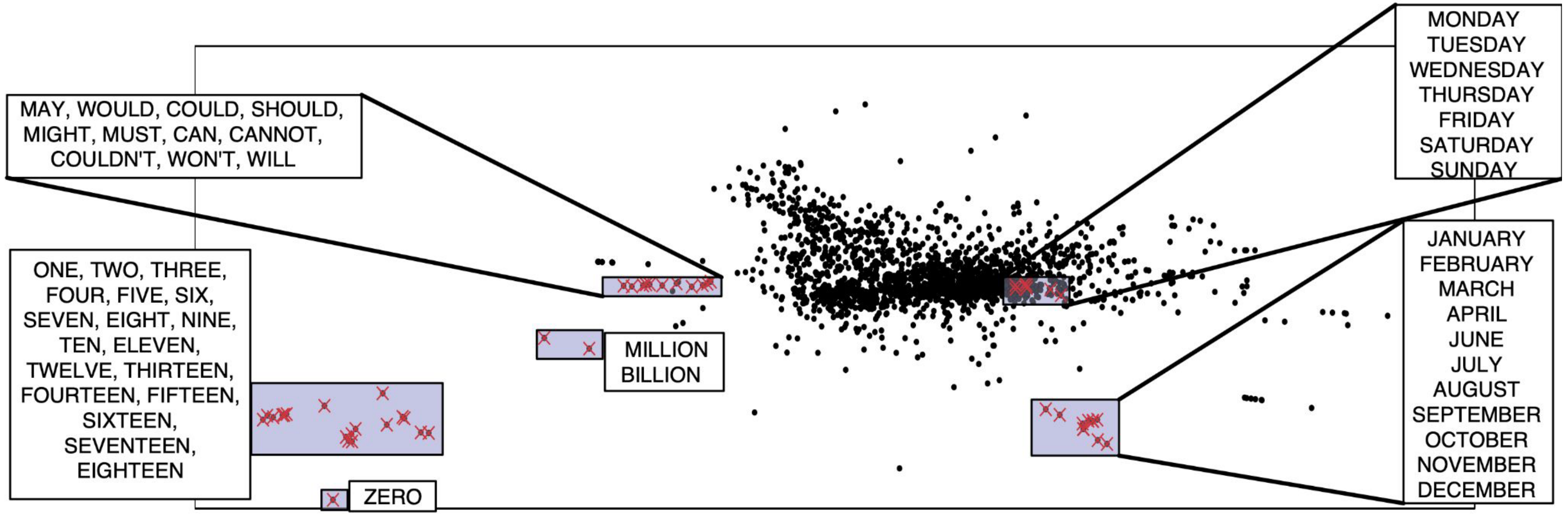
Logistics

- P2 released last Thursday
- Submit on gradescope
  - If you worked in a group, create a group and then submit
- HW2 due on **03/05**

Cornell Bowers C·IS

What are word embeddings

- What are Word Embeddings?
  - vector representations of words that capture semantic relationships
  - Latent Semantic Analysis / Indexing [S. Deerwester et al 1988]



[Blitzer et al. Neurips, 2004]

Cornell Bowers C·IS

Word2Vec

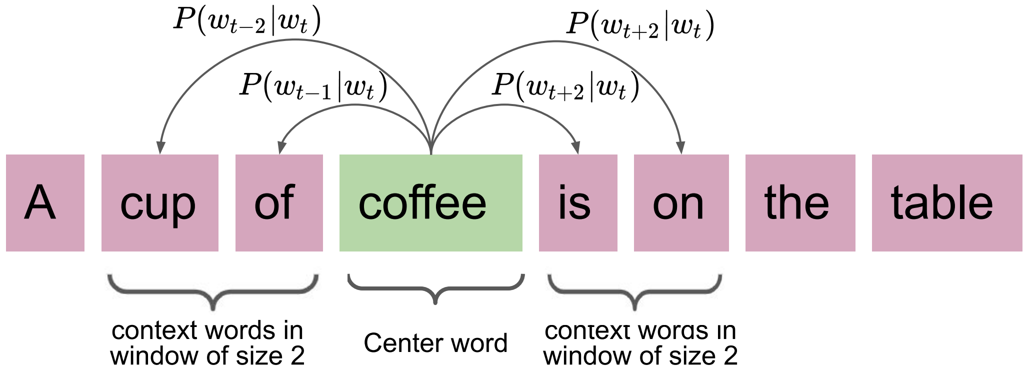
- We want vectors for words so that the context of a word can suggest the vector of this word, and vice versa
- Idea: **Similar words appear in similar contexts**

A cup of **coffee** is on the table.
Coffee helps me focus.
Espresso is my favorite type of **coffee**.

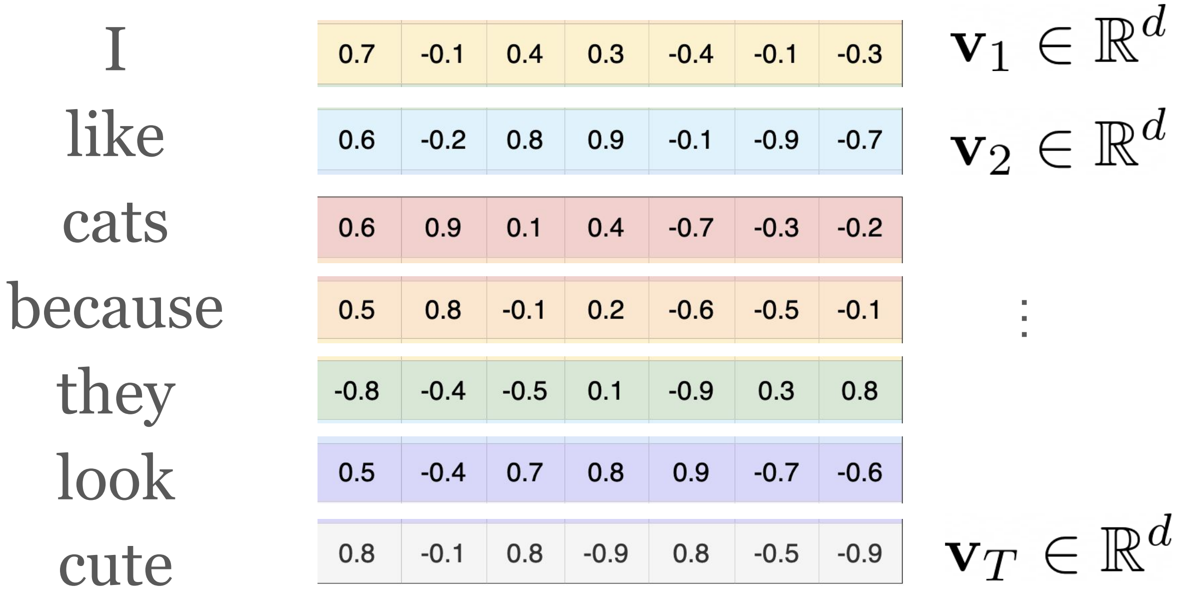
Cornell Bowers C·IS

Word2Vec - Training

SkipGram - Predict context from target

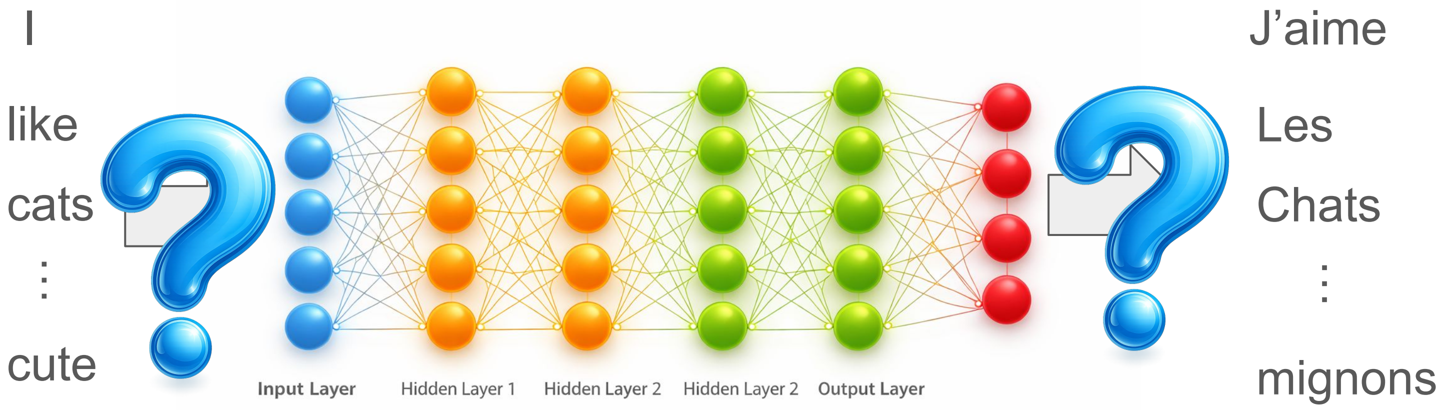


Big Question: How to model sequences of words?



I	0.7	-0.1	0.4	0.3	-0.4	-0.1	-0.3	$\mathbf{v}_1 \in \mathbb{R}^d$
like	0.6	-0.2	0.8	0.9	-0.1	-0.9	-0.7	$\mathbf{v}_2 \in \mathbb{R}^d$
cats	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2	
because	0.5	0.8	-0.1	0.2	-0.6	-0.5	-0.1	$\vdots$
they	-0.8	-0.4	-0.5	0.1	-0.9	0.3	0.8	
look	0.5	-0.4	0.7	0.8	0.9	-0.7	-0.6	
cute	0.8	-0.1	0.8	-0.9	0.8	-0.5	-0.9	$\mathbf{v}_T \in \mathbb{R}^d$

How to use word vectors with neural networks?



- Inputs and outputs don't have fixed lengths
- Weights are not shared

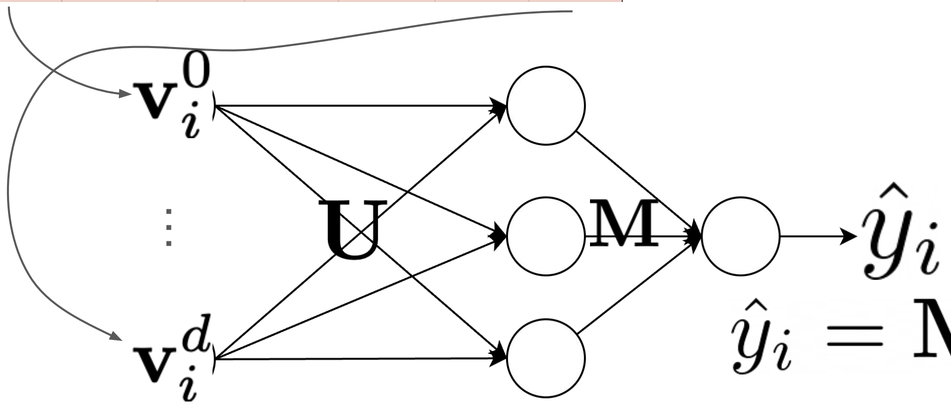
Let’s simplify!

What if we have a single word and a single output?

0 = not animal
1 = animal

cat	0.6	0.9	0.1	0.4	-0.7	-0.3	-0.2

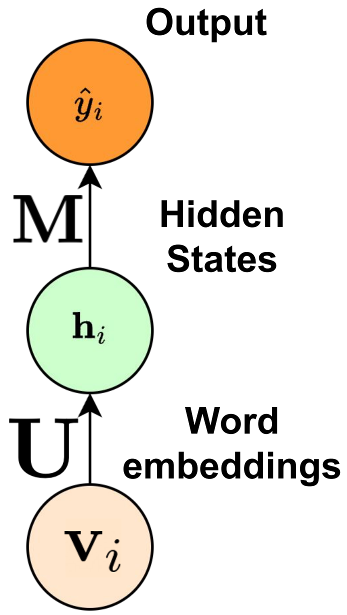
$\mathbf{v}_i \in \mathbb{R}^d$



$\hat{y}_i = \mathbf{M}_i \mathbf{h}_i$

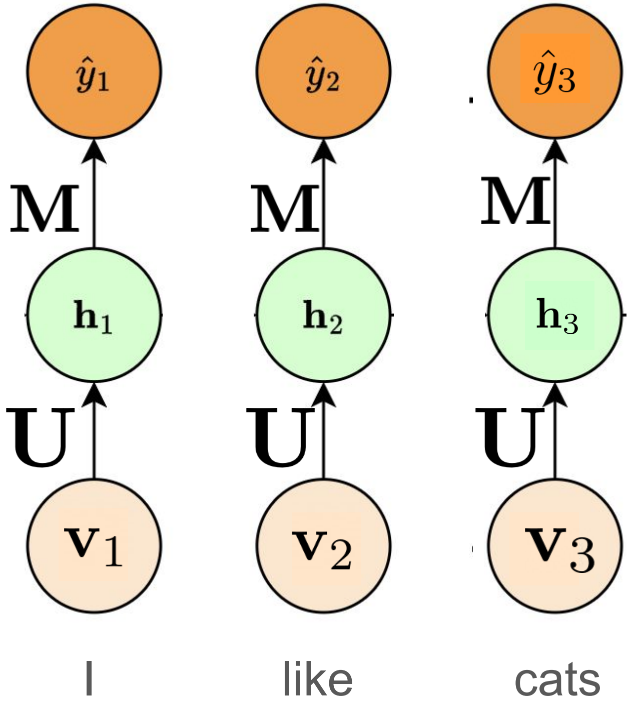
$\mathbf{h}_i = \sigma\left(\mathbf{U}\mathbf{v}_i\right)$

typically a tanh()



Cornell Bowers C·IS

Towards RNNs



Output

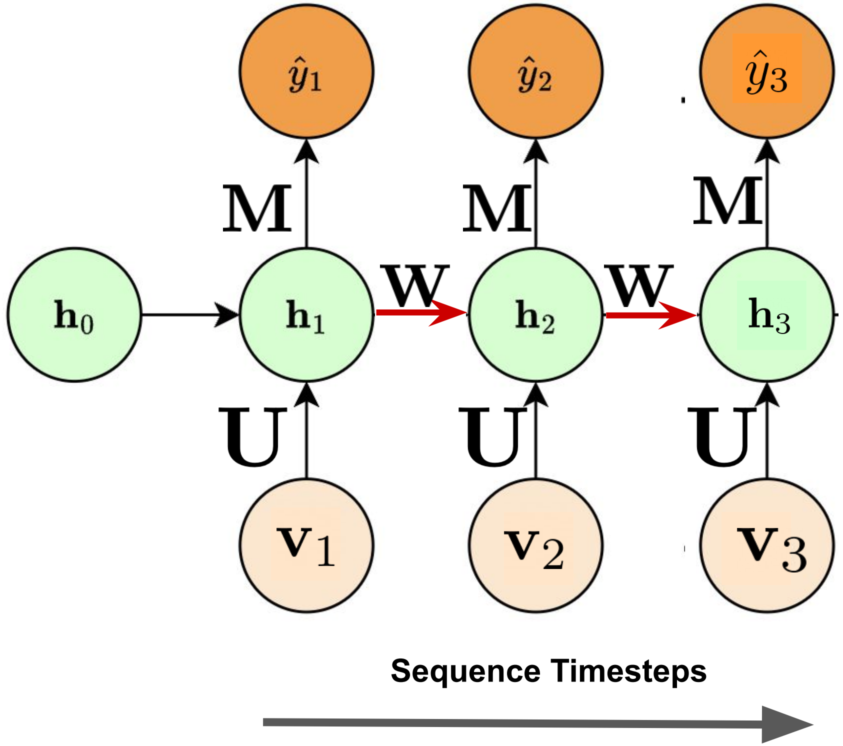
$$\hat{y}_i = \mathbf{M}\mathbf{h}_i$$

Hidden State

$$\mathbf{h}_i = \sigma\left(\mathbf{U}\mathbf{v}_i\right)$$

Cornell Bowers C·IS

Towards RNNs



Output

$$\hat{y}_i = \mathbf{M}\mathbf{h}_i$$

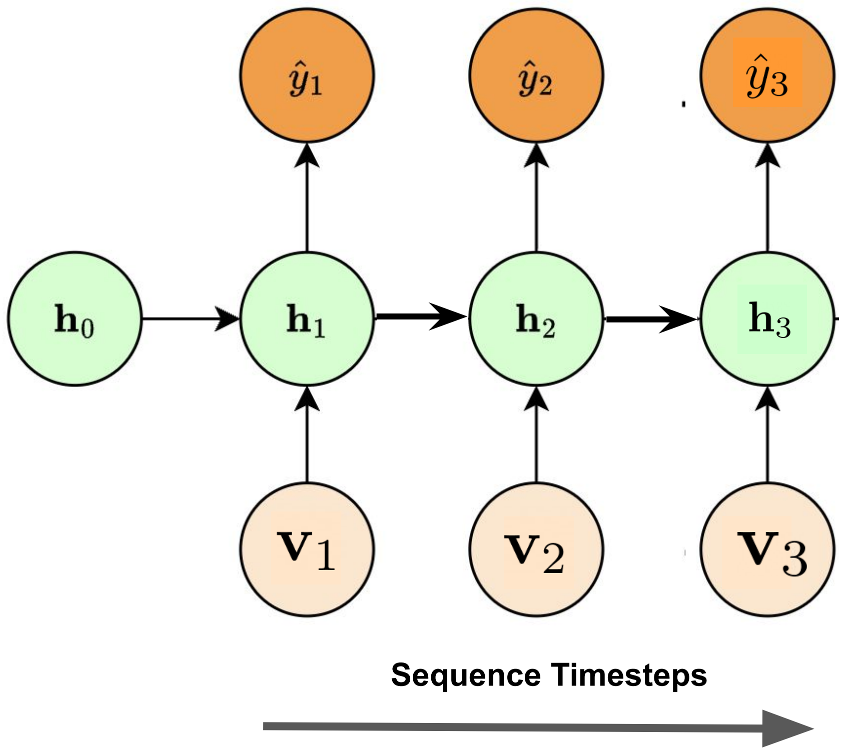
Hidden State

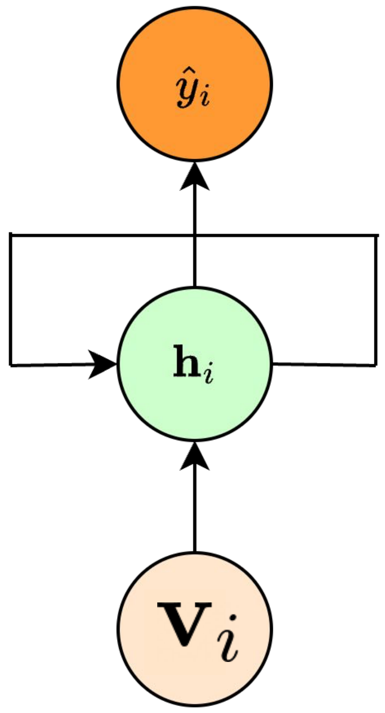
$$\mathbf{h}_i = \sigma\left(\mathbf{U}\mathbf{v}_i + \mathbf{W}\mathbf{h}_{i-1}\right)$$

Cornell Bowers C·IS

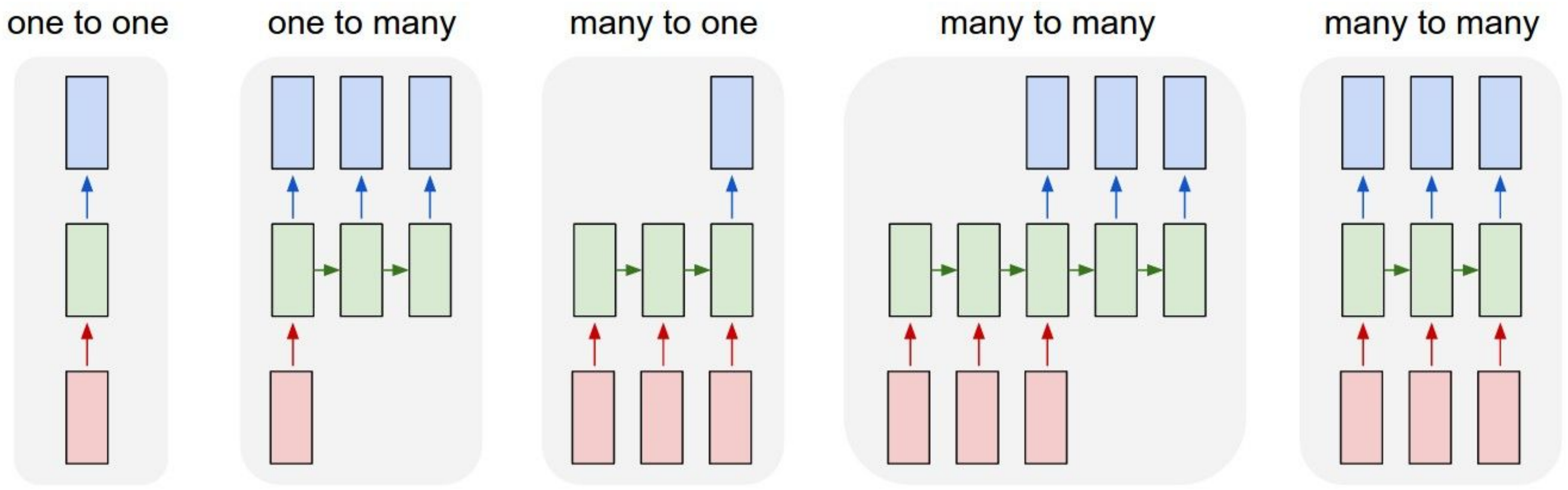
Recurrent neural network (RNN)

Use the same parameters across different timesteps.





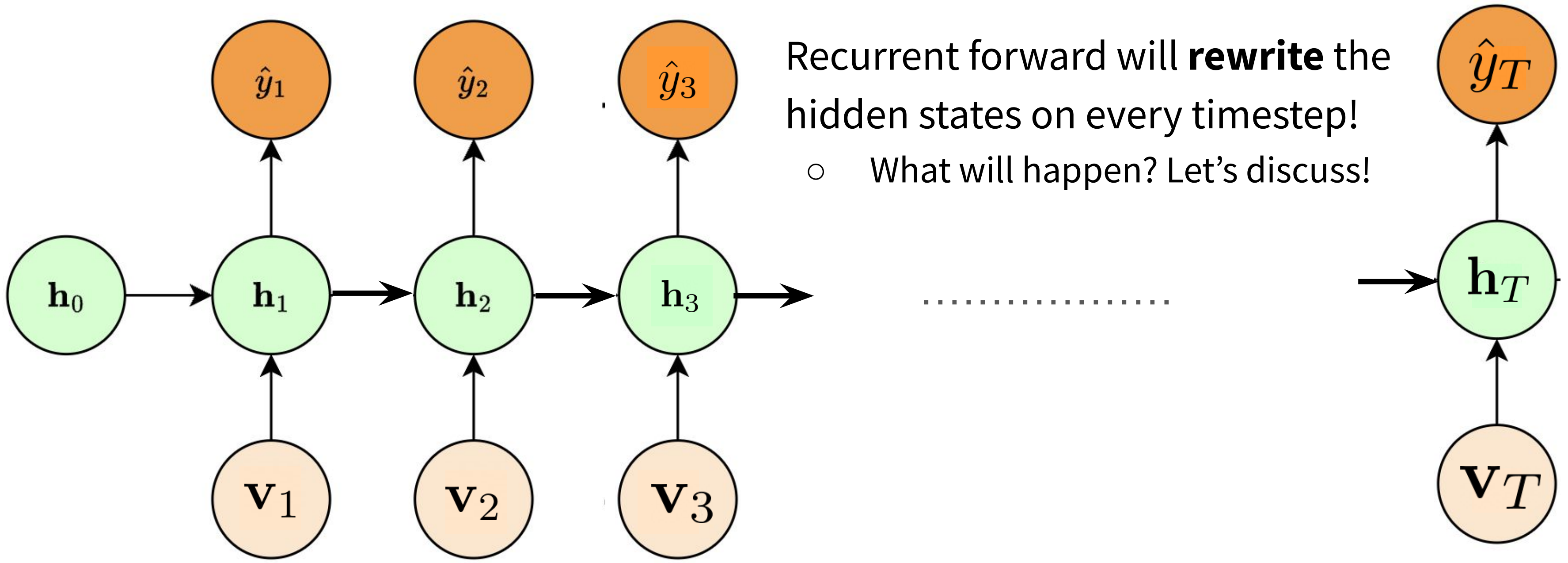
Discuss: Which tasks can you perform with RNNs? Can you find an example of each task?



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

Cornell Bowers C·IS

RNN: Issues under **Looooooooong** Context

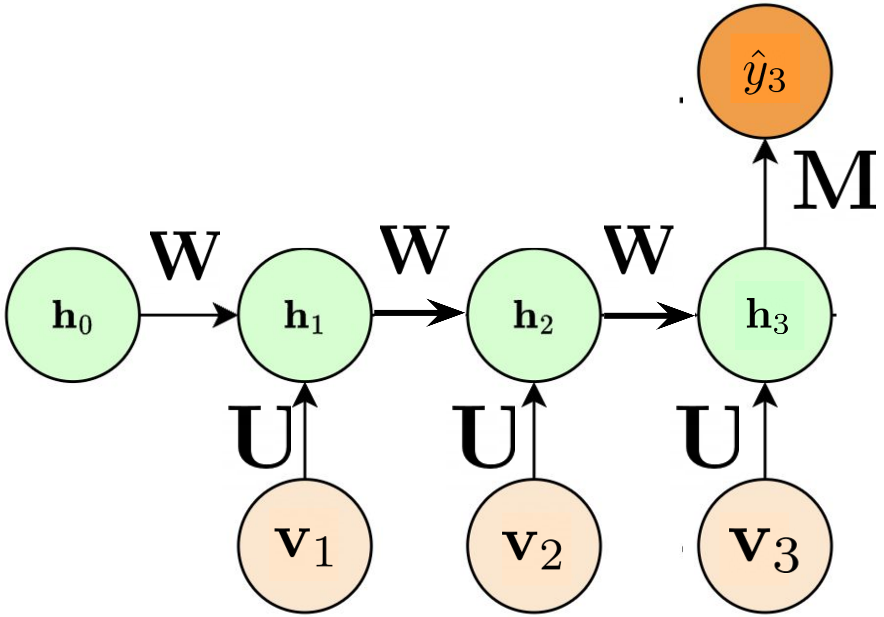


Recurrent forward will **rewrite** the hidden states on every timestep!

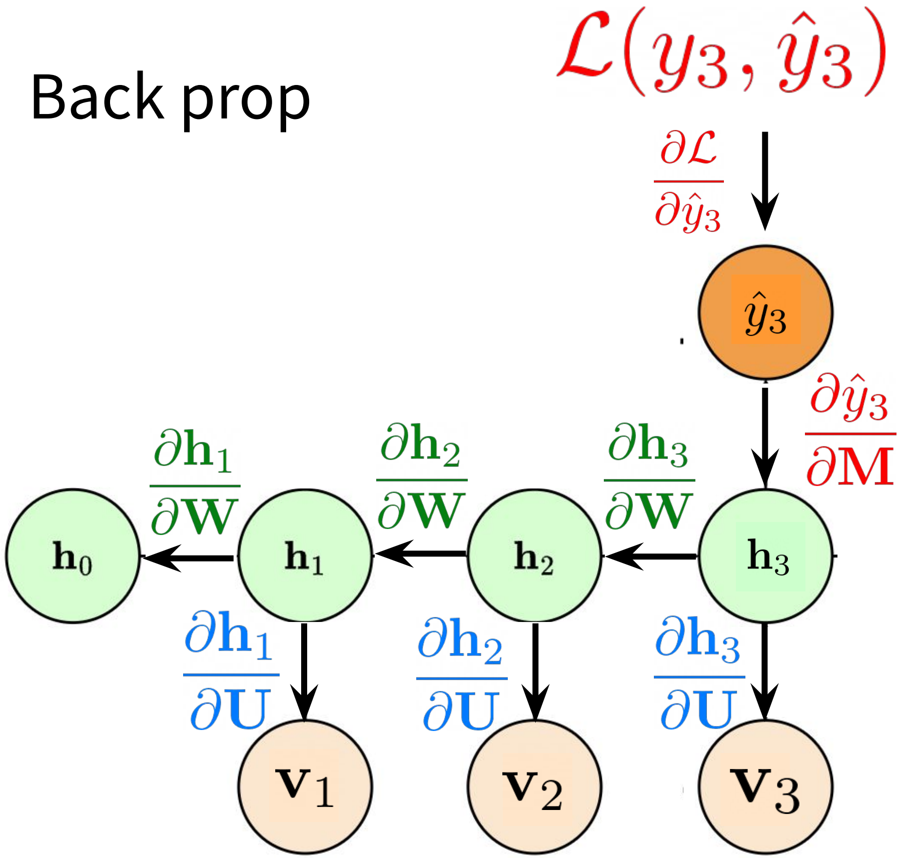
- What will happen? Let’s discuss!

Cornell Bowers CIS

Forward prop



Back prop

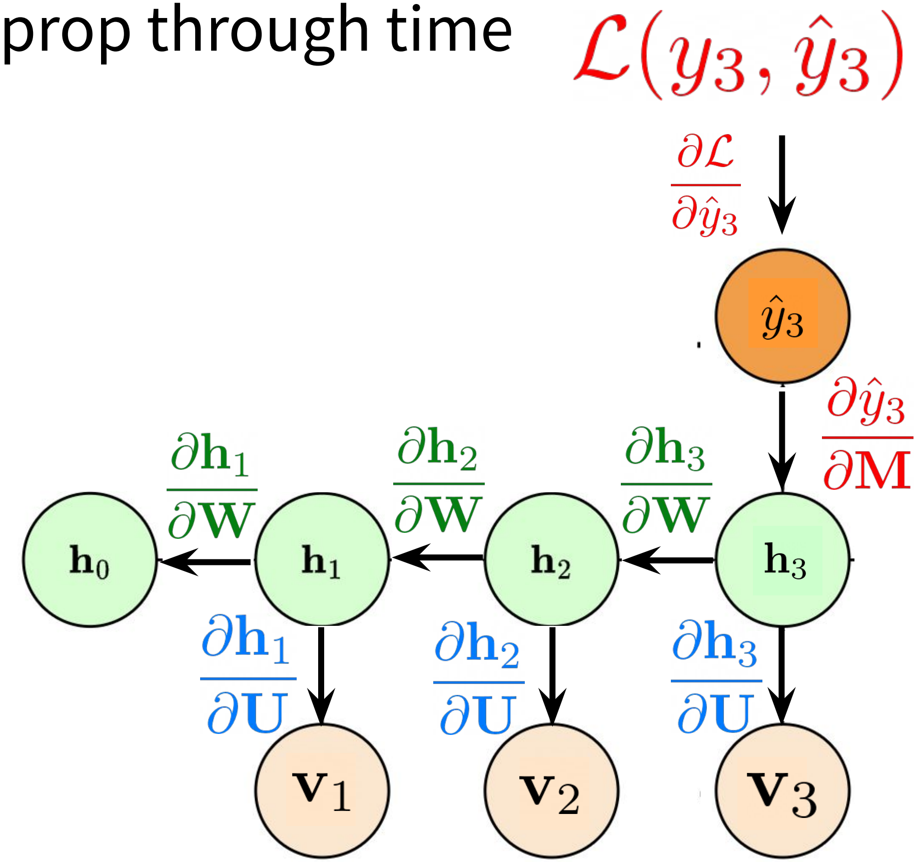


$\mathcal{L}(y_3, \hat{y}_3)$
$\frac{\partial \mathcal{L}}{\partial \hat{y}_3}$
$\hat{y}_3$
$\frac{\partial \hat{y}_3}{\partial \mathbf{M}}$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{W}}$
$\mathbf{h}_0$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_3$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{U}}$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$

Cornell Bowers C·IS

Back prop through time

- 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



$\mathcal{L}(y_3, \hat{y}_3)$
$\frac{\partial \mathcal{L}}{\partial \hat{y}_3}$
$\hat{y}_3$
$\frac{\partial \hat{y}_3}{\partial \mathbf{M}}$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{W}}$
$\mathbf{h}_0$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_3$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{U}}$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$

Cornell Bowers C·IS

Back prop through time

$$\frac{\partial \mathcal{L}}{\partial \mathbf{M}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_3} \cdot \frac{\partial \hat{y}_3}{\partial \mathbf{M}}$$

Gradients wrt W from last time step:

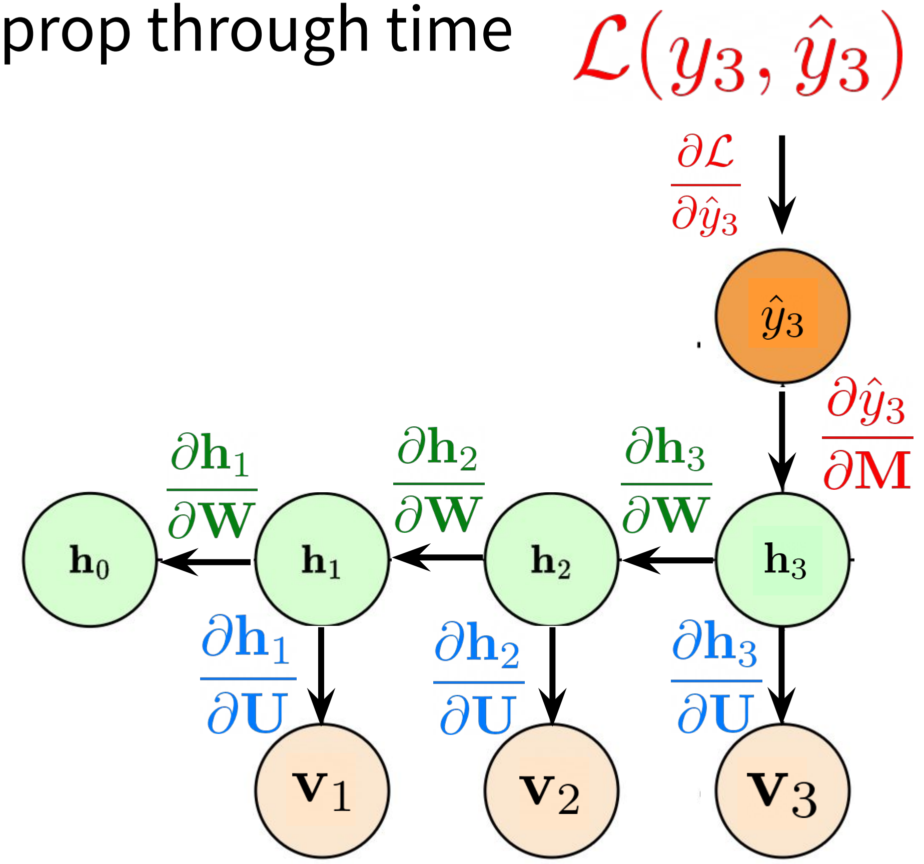
$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{W}}$$

Gradients wrt W from time step 2:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2}{\partial \mathbf{W}}$$

Gradients wrt W from time step 1:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$$



$\mathcal{L}(y_3, \hat{y}_3)$
$\frac{\partial \mathcal{L}}{\partial \hat{y}_3}$
$\hat{y}_3$
$\frac{\partial \hat{y}_3}{\partial \mathbf{M}}$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{W}}$
$\mathbf{h}_0$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_3$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{U}}$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$

Cornell Bowers C·IS

T = 3

What is the general form of $\frac{\partial \mathcal{L}}{\partial \mathbf{W}}$ with **T = T**, at time step t?

Gradients wrt W from t=3:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{W}}$$

Gradients wrt W from t = 2:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2}{\partial \mathbf{W}}$$

Gradients wrt W from t = 1:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_3} \frac{\partial \hat{y}_3}{\partial \mathbf{h}_3} \frac{\partial \mathbf{h}_3}{\partial \mathbf{h}_2} \frac{\partial \mathbf{h}_2}{\partial \mathbf{h}_1} \frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \boxed{\frac{\partial \mathbf{h}_T}{\partial \mathbf{h}_{T-1}} \cdots\cdots \frac{\partial \mathbf{h}_{t+1}}{\partial \mathbf{h}_t}} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \boxed{\left(\prod_{i=t}^{T-1} \frac{\partial \mathbf{h}_{i+1}}{\partial \mathbf{h}_i}\right)} \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}}$$

Cornell Bowers CIS

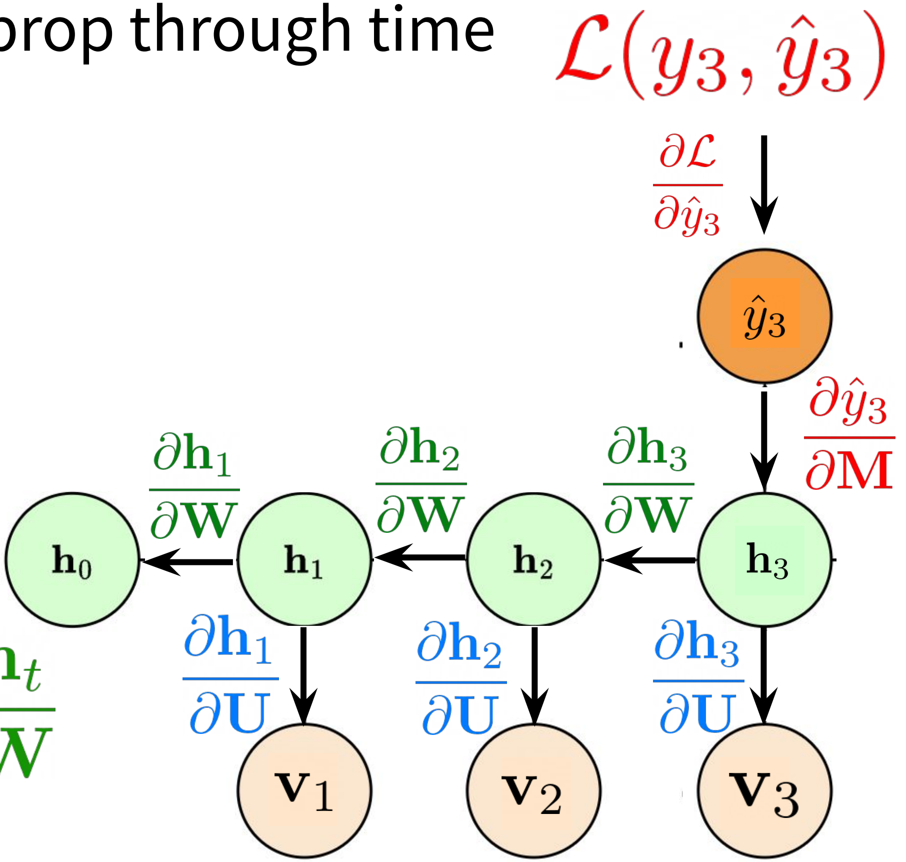
Back prop through time

Gradients wrt W from time step t:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \left(\prod_{i=t}^{T-1} \frac{\partial \mathbf{h}_{i+1}}{\partial \mathbf{h}_i}\right) \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}}$$

Each timestamp contributes to the gradient!
Summing over all timestamps:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial \hat{y}_T} \frac{\partial \hat{y}_T}{\partial \mathbf{h}_T} \left(\prod_{i=t}^{T-1} \frac{\partial \mathbf{h}_{i+1}}{\partial \mathbf{h}_i}\right) \frac{\partial \mathbf{h}_t}{\partial \mathbf{W}}$$


$\mathcal{L}(y_3, \hat{y}_3)$
$\frac{\partial \mathcal{L}}{\partial \hat{y}_3}$
$\hat{y}_3$
$\frac{\partial \hat{y}_3}{\partial \mathbf{M}}$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{W}}$
$\mathbf{h}_0$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_3$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{U}}$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$

Cornell Bowers C·IS

RNN: Issues under **Looooooooong** Context

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

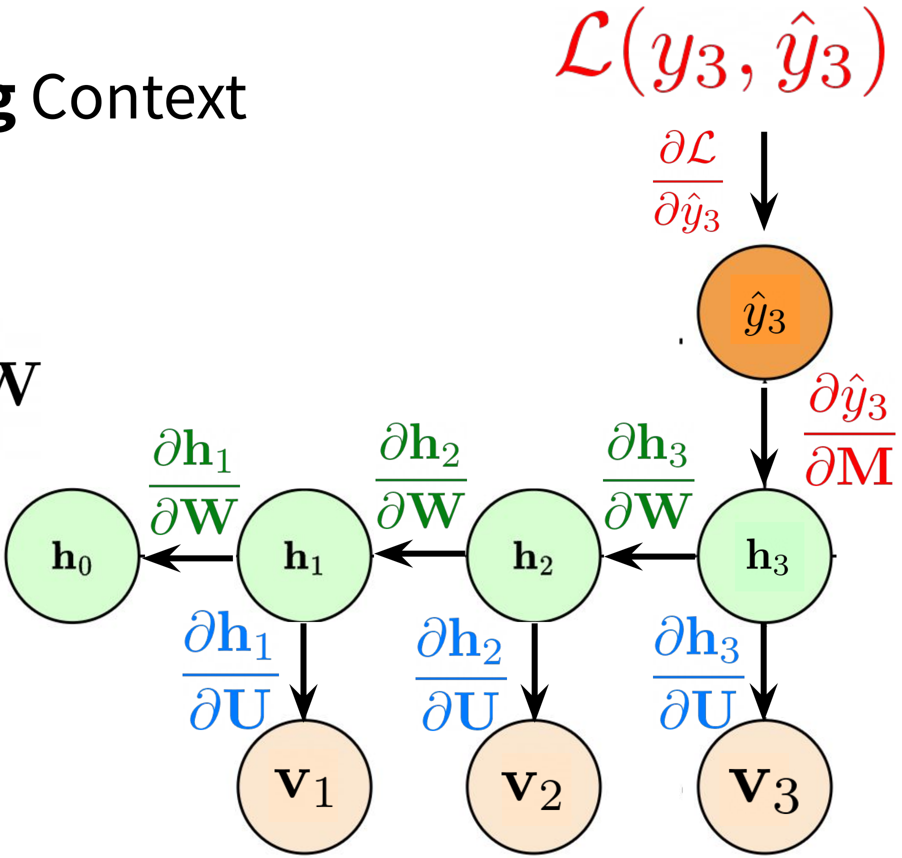
$$\frac{\partial \mathbf{h}_t}{\partial \mathbf{h}_{t-1}} = \operatorname{diag}(\sigma'(\mathbf{W}\mathbf{h}_{t-1} + \mathbf{U}\mathbf{v}_t))\mathbf{W}$$

- **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***

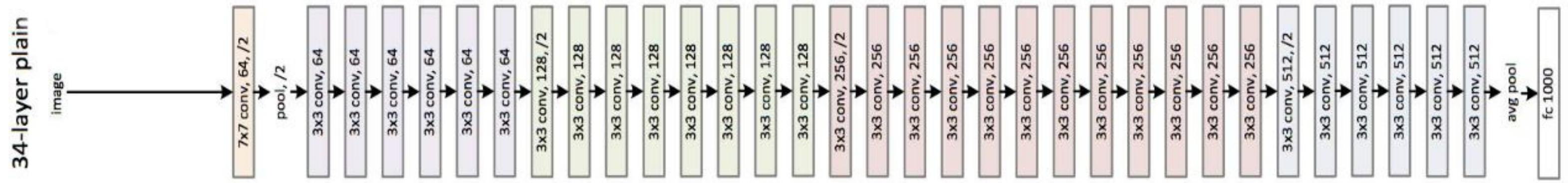
If $\|\frac{\partial \mathbf{h}_i}{\partial \mathbf{h}_{i-1}}\| > 1$ and T is large, $\|\frac{\partial \mathcal{L}(\hat{y}_T)}{\partial \mathbf{h}_1}\| \to \text{inf}.$


$\mathcal{L}(y_3, \hat{y}_3)$
$\frac{\partial \mathcal{L}}{\partial \hat{y}_3}$
$\hat{y}_3$
$\frac{\partial \hat{y}_3}{\partial \mathbf{M}}$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{W}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{W}}$
$\mathbf{h}_0$
$\mathbf{h}_1$
$\mathbf{h}_2$
$\mathbf{h}_3$
$\frac{\partial \mathbf{h}_1}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_2}{\partial \mathbf{U}}$
$\frac{\partial \mathbf{h}_3}{\partial \mathbf{U}}$
$\mathbf{v}_1$
$\mathbf{v}_2$
$\mathbf{v}_3$

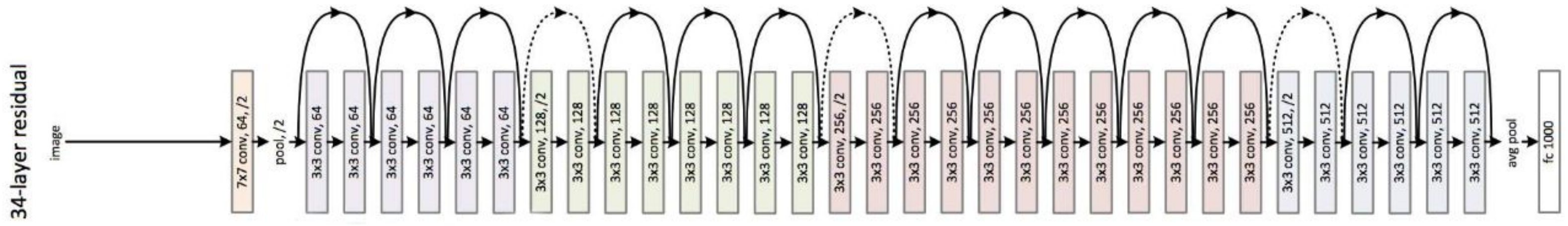
Cornell Bowers C·IS

Recall: ResNet

“Plain” Network

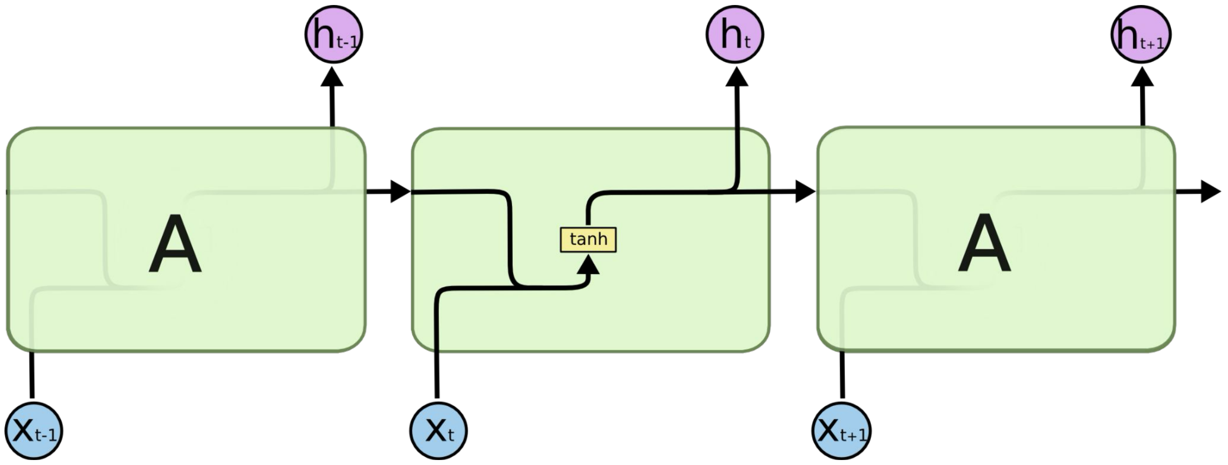


ResNet

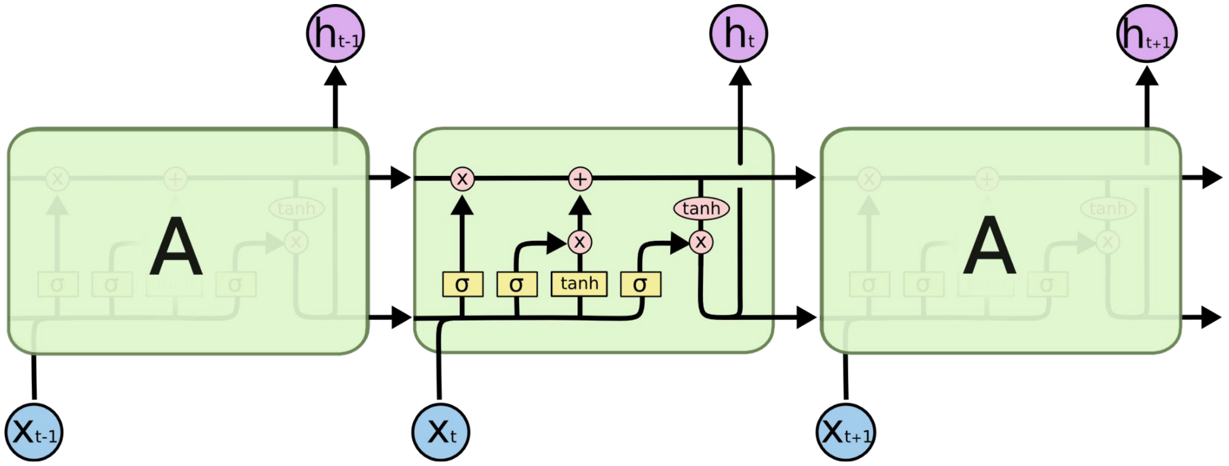


[He, Kaiming, et al. "Deep residual learning for image recognition." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.]

RNN

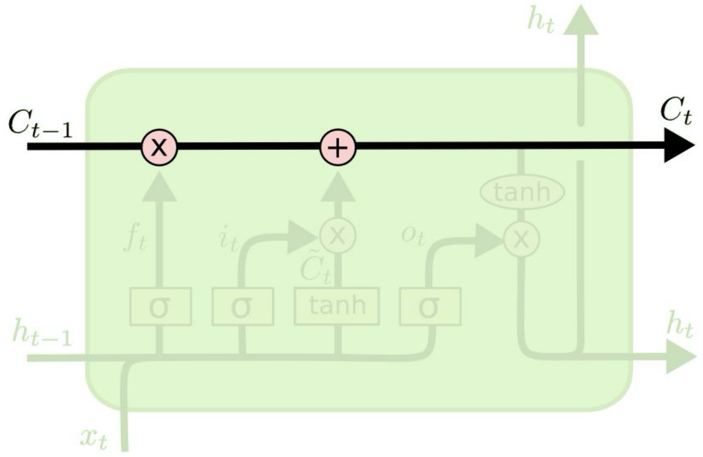


Long-short Term Memory (LSTM)



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!



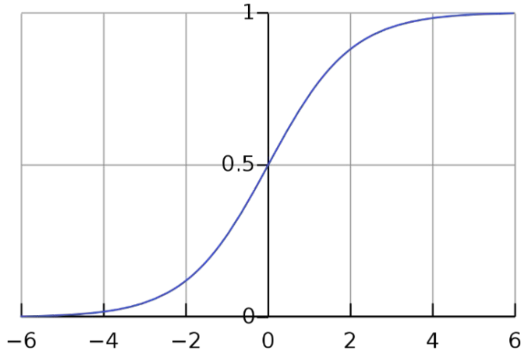
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

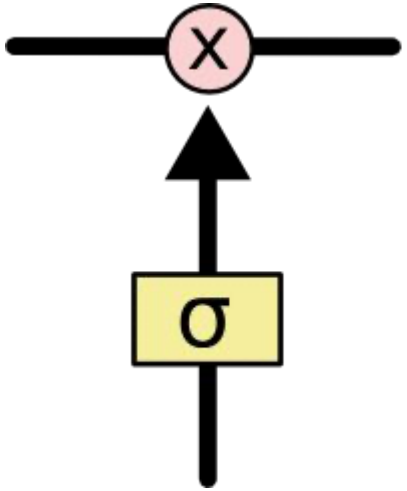
LSTMs- Gates

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

Sigmoid Function

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

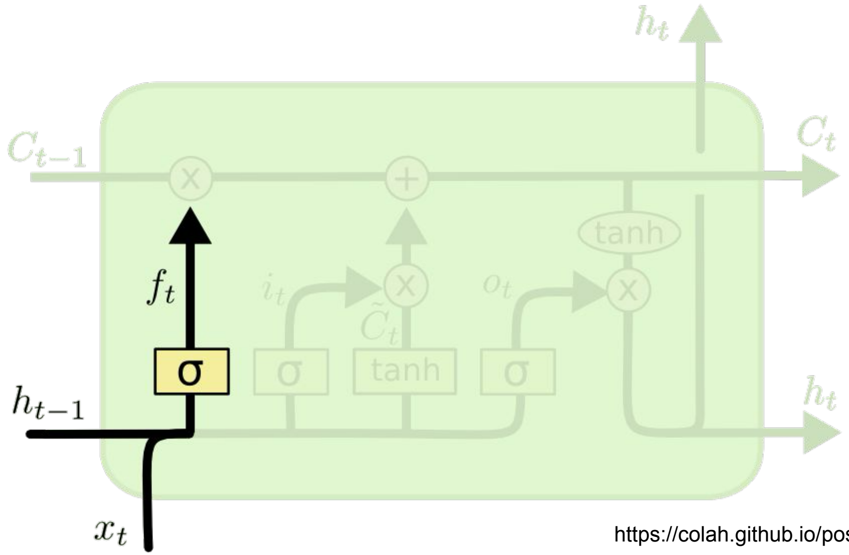




https://colah.github.io/posts/2015-08-Understanding-LSTMs/

LSTMs- Forget Gate

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

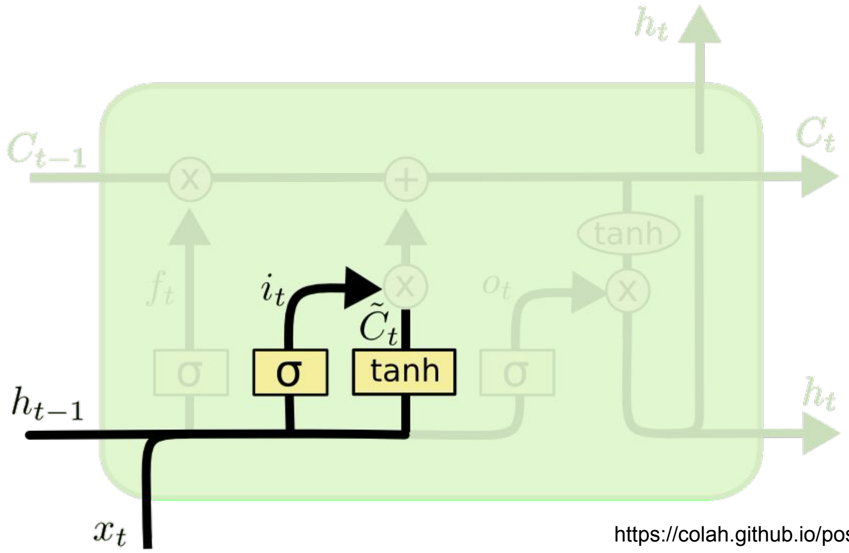


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

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



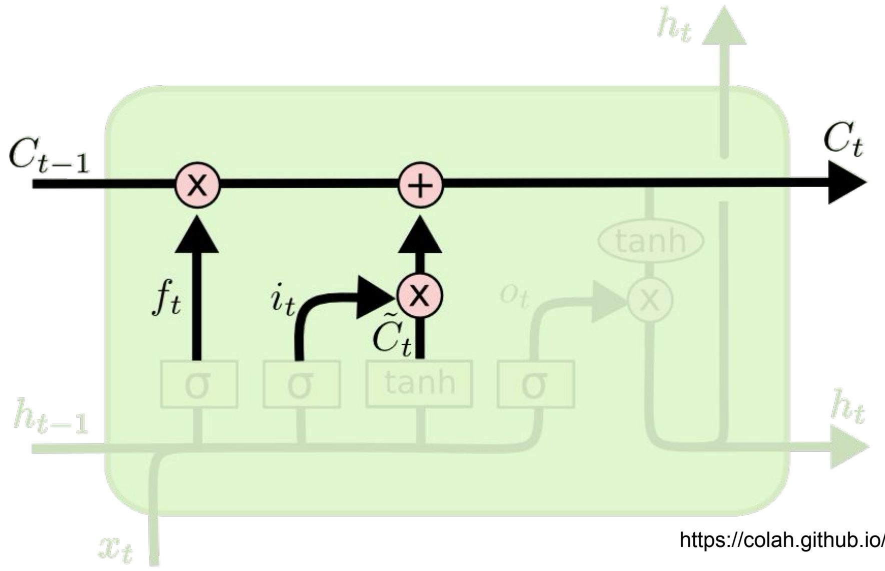
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

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

LSTM- Cell Update

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

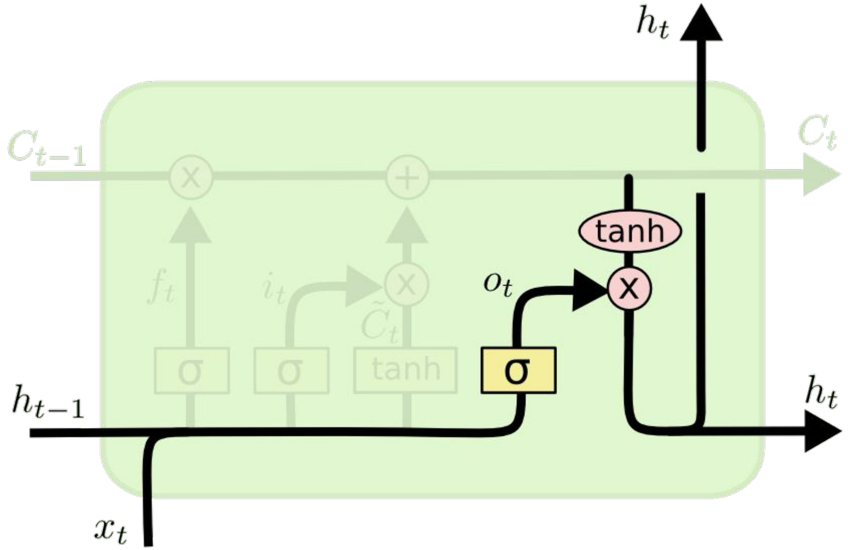


$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

LSTM- Output Gate

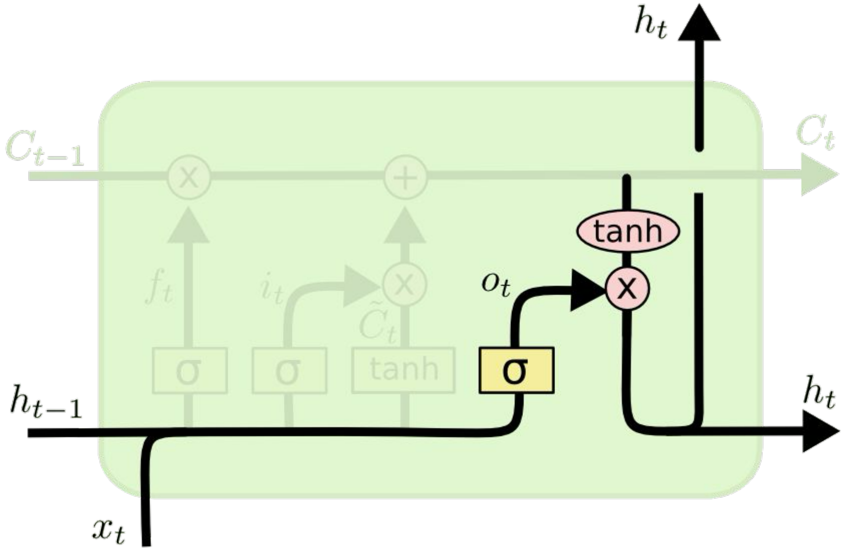
- Output gate- function of current input and previous hidden state
- Controls flow of information from the cell state to the hidden state
- Given some weight matrix W_o, how do we write to o_t and h_t?



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



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

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

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

LSTMs

- Add a cell state to store information
  - Gradient flows along the cell state
- Update cell state with parameterized gating functions
- Performs better with long sequences

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

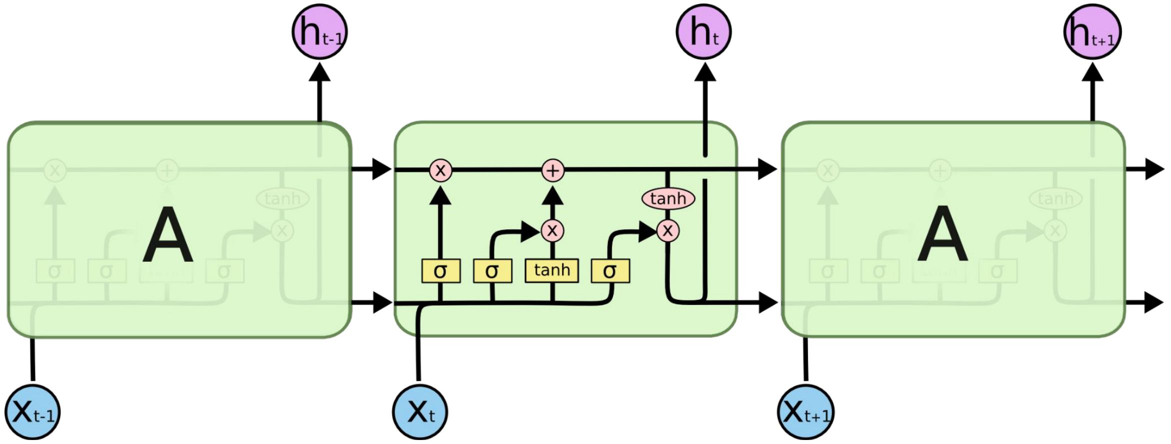
$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \ + \ b_i\right)$$

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

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

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

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

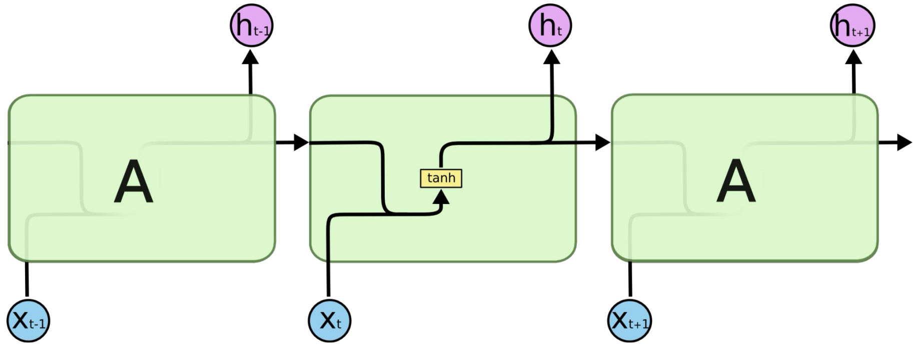


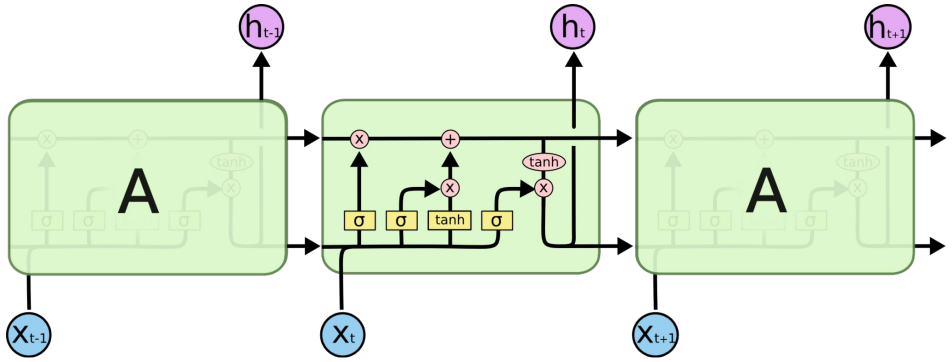
https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Cornell Bowers C·IS

RNN vs. LSTM

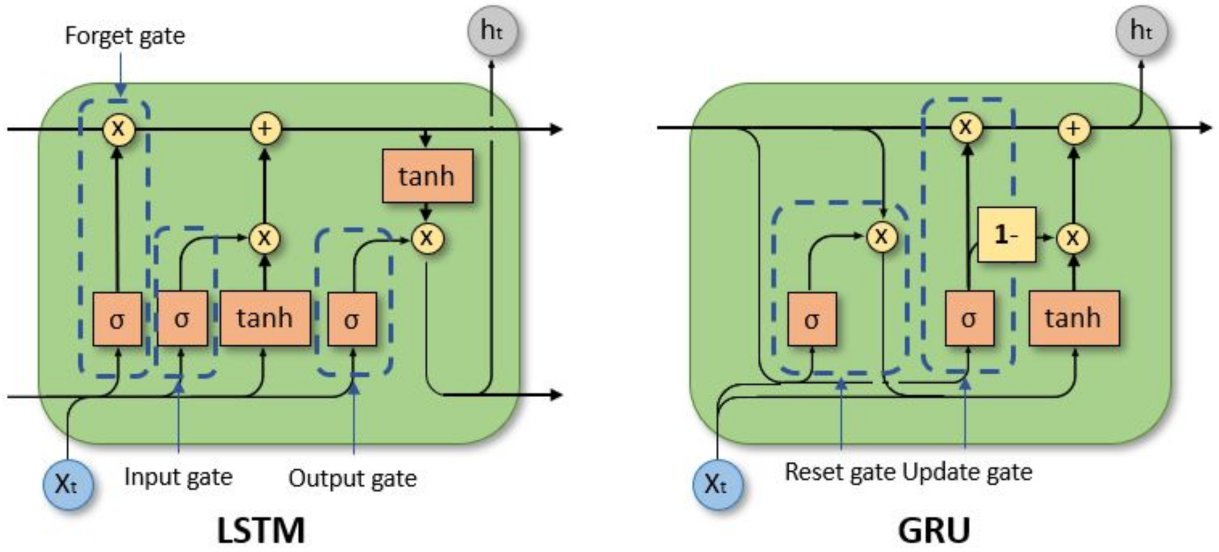
- **RNN**
  - Can be applied to variable-length sequences
  - Share parameters across time
  - Hard to train!
- **LSTM**
  - Mitigates the vanishing gradient problem with the cell state
  - Better for long sequences





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

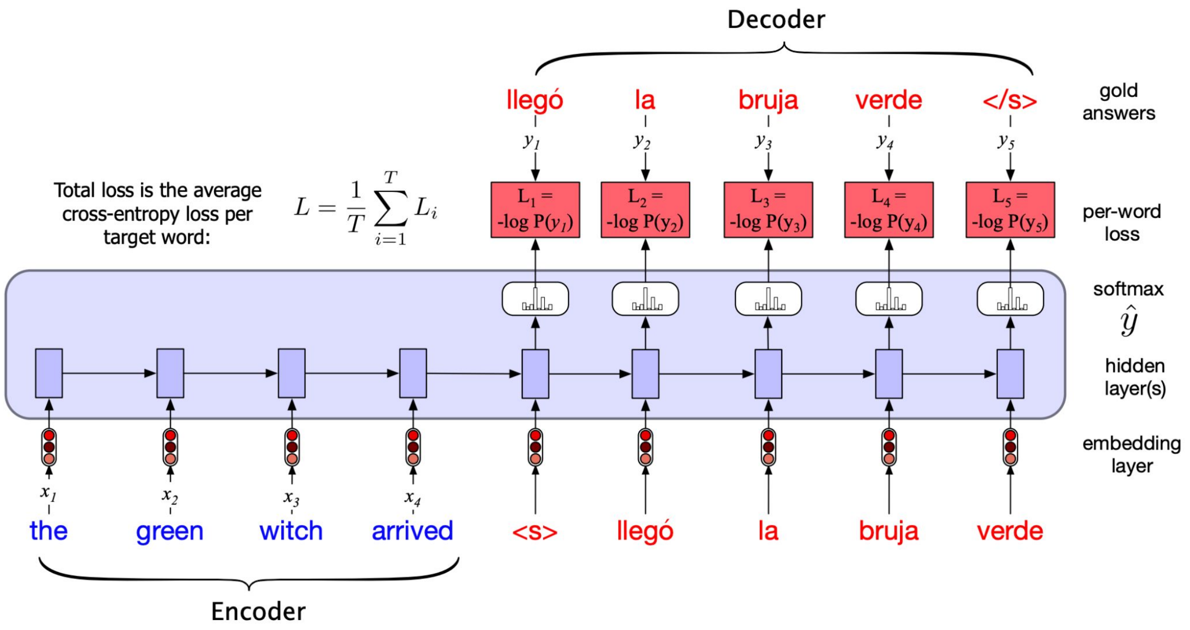
Gated recurrent units (GRUs)



https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Sequence-to-Sequence Generation

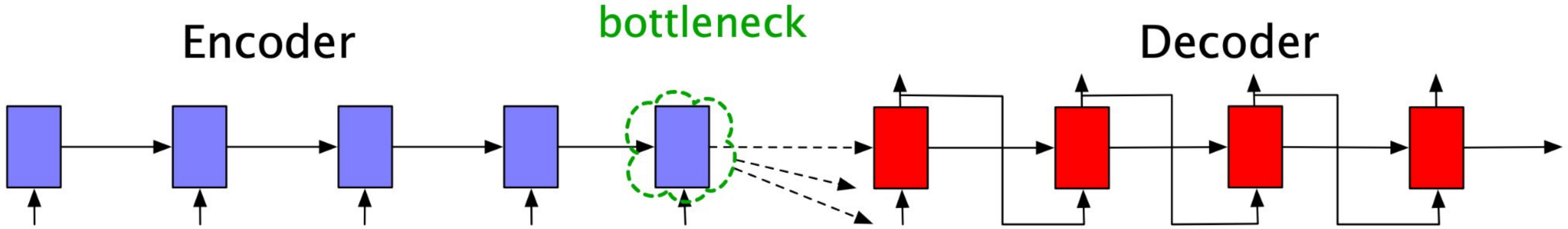
- Map some input sequence to a target sequence
- Applications:
  - Machine translation
  - News summarization
  - ChatGPT!



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

Bottleneck Problem

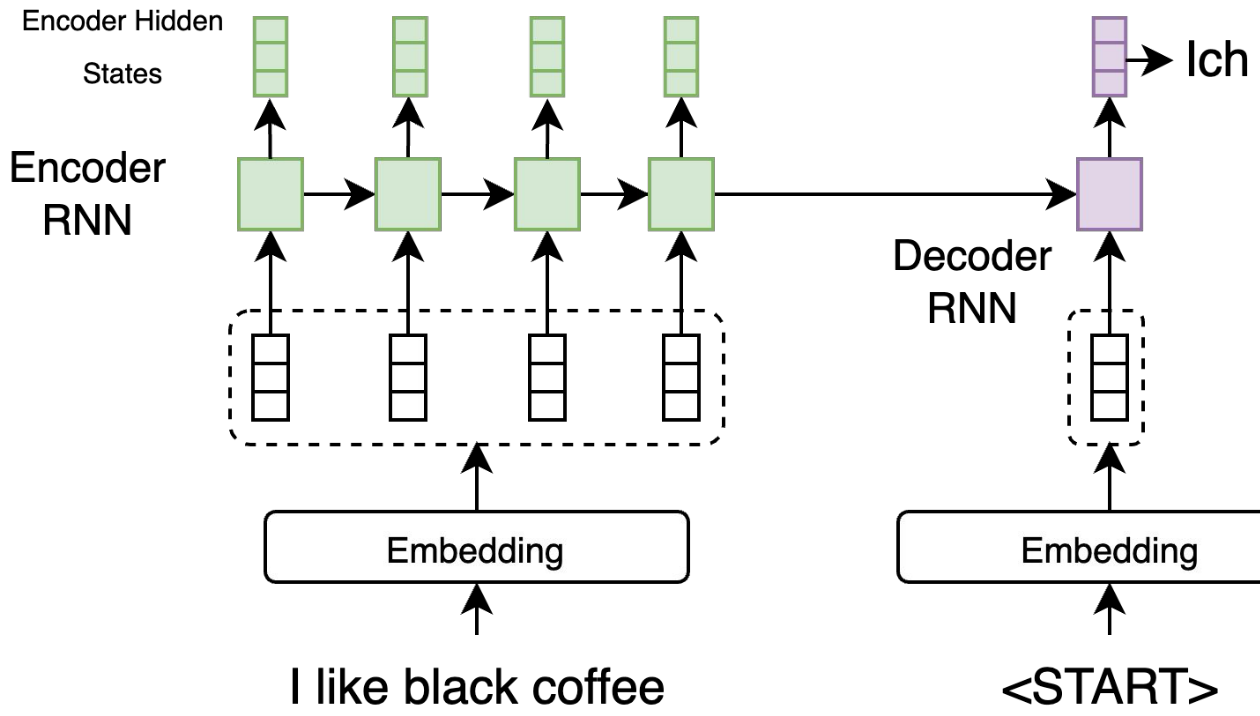
- All the information about the source sequence must be stored in a single vector
  - How to translate a long paragraph?
  - How to summarize long articles?



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

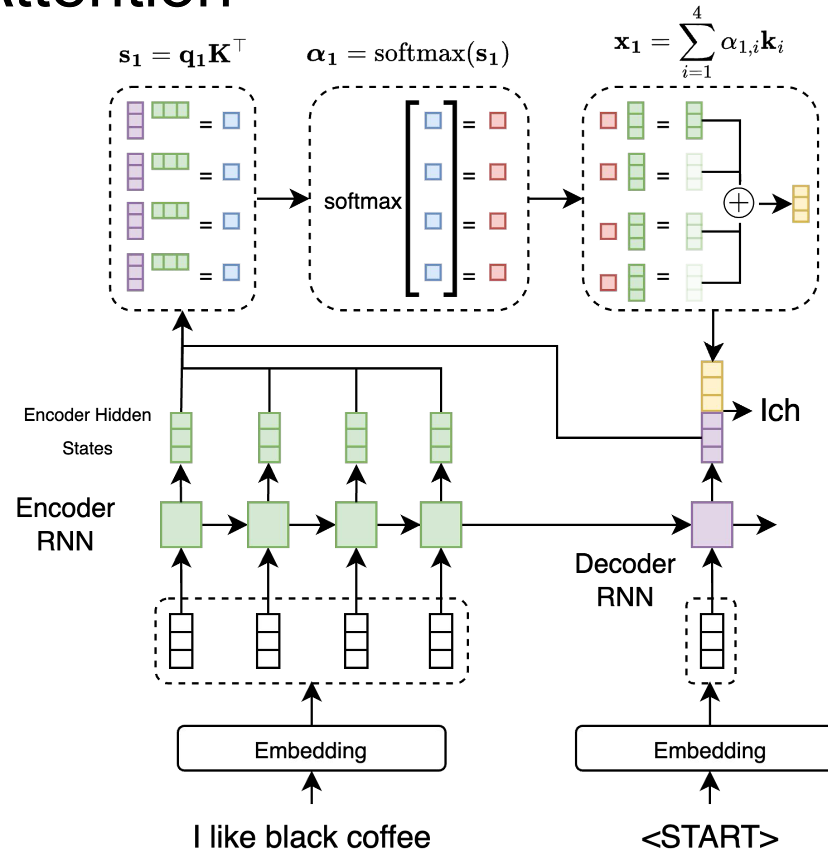
RNN for Machine Translation

Would be nice if we could “look back” at previous hidden states



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

RNN with Attention

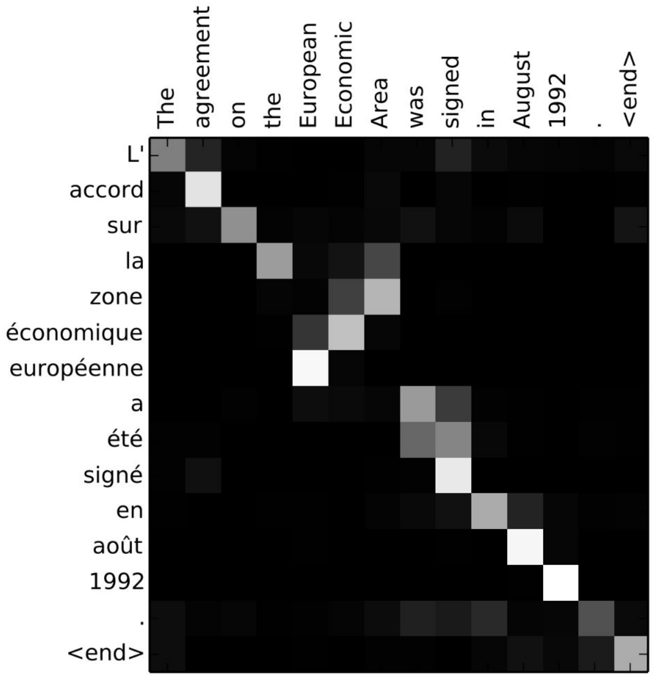


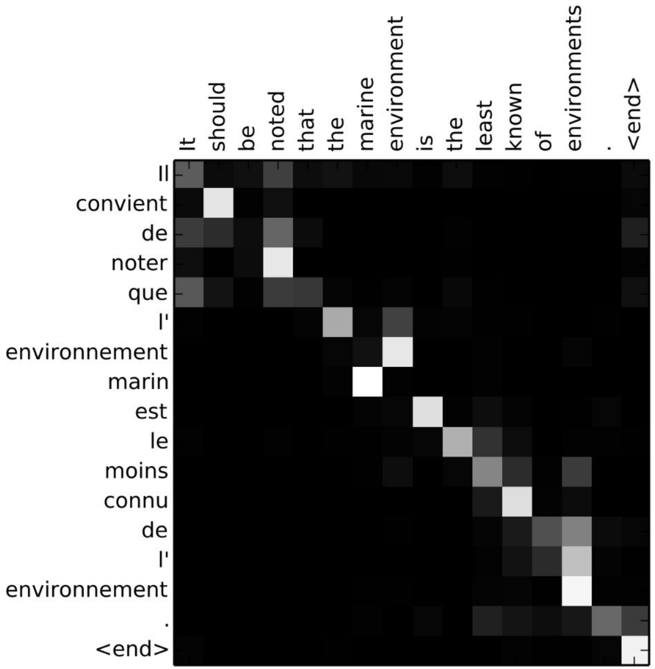
$\mathbf{s_1} = \mathbf{q_1}\mathbf{K}^\top$
$\boldsymbol{\alpha}_1 = \mathrm{softmax}(\mathbf{s_1})$
$\mathbf{x_1} = \sum_{i=1}^{4} \alpha_{1,i}\mathbf{k}_i$
softmax
Encoder Hidden States
Encoder RNN
Decoder RNN
Ich
Embedding
Embedding
I like black coffee
<START>

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

Visualizing Attention

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



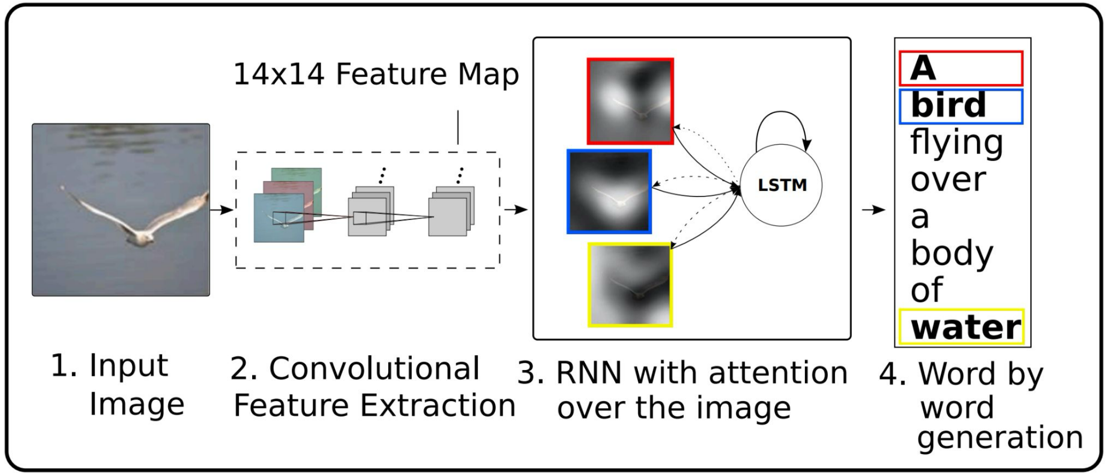


Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate."

Attention Application- Image Captioning!

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

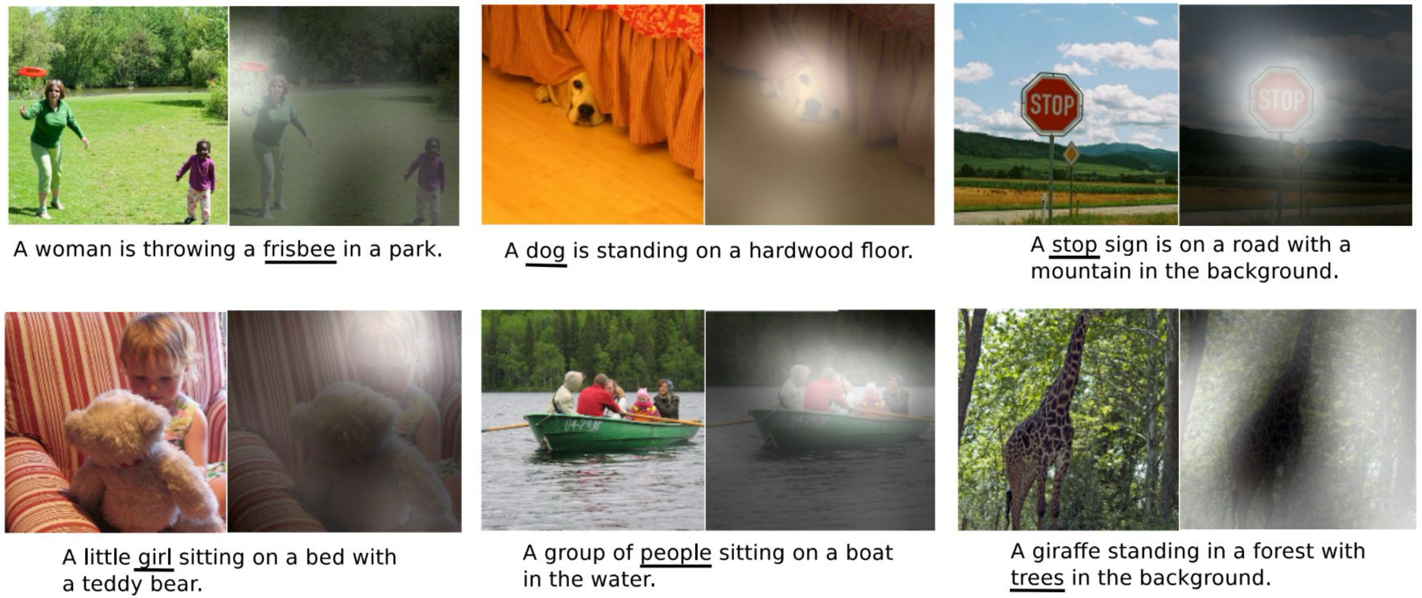
Figure 1. Our model learns a words/image alignment. The visualized attentional maps (3) are explained in section 3.1 & 5.4


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

Visualize Attention Weights

- Learns to focus on relevant regions of the image

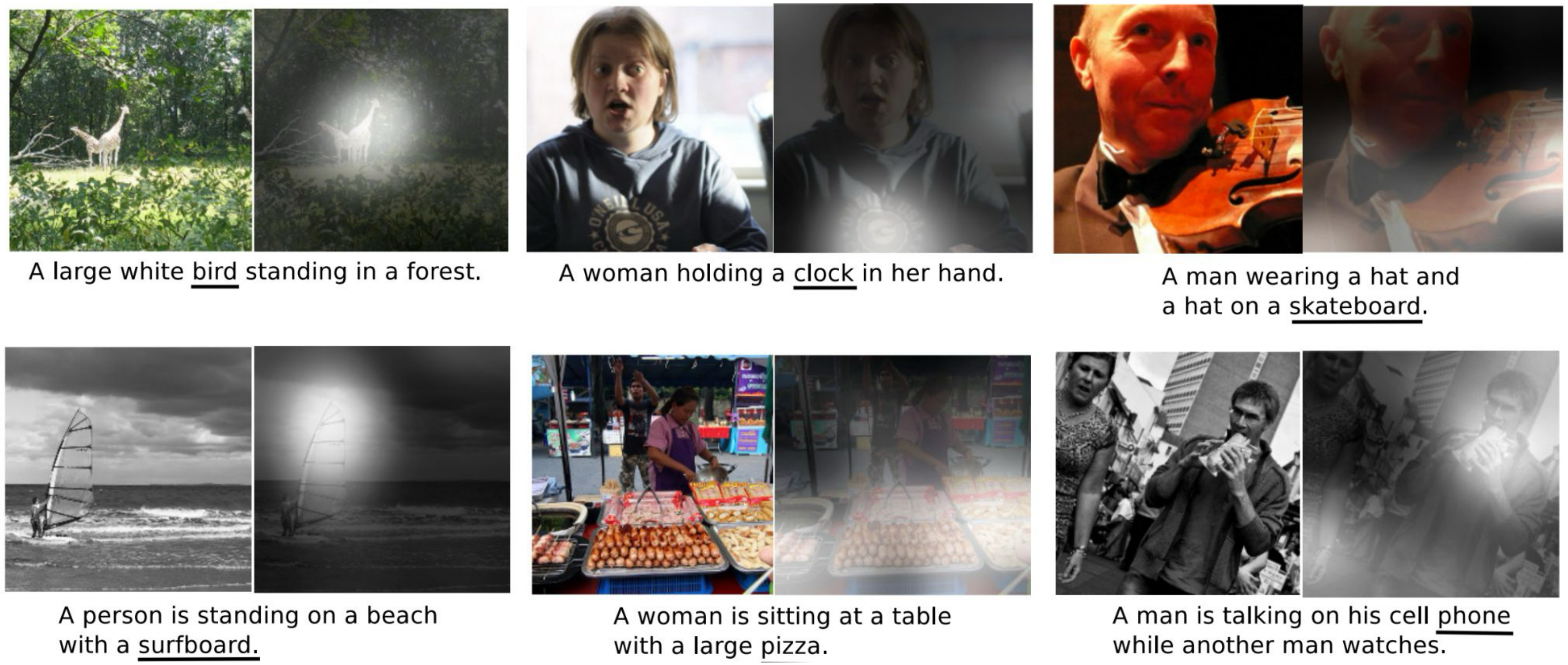
Figure 3. Examples of attending to the correct object (*white* indicates the attended regions, *underlines* indicated the corresponding word)


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.

Error Analysis!

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



A large white bird standing in a forest.

A woman holding a clock in her hand.

A man wearing a hat and a hat on a skateboard.

A person is standing on a beach with a surfboard.

A woman is sitting at a table with a large pizza.

A man is talking on his cell phone while another man watches.

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!