



SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Established in collaboration with MIT

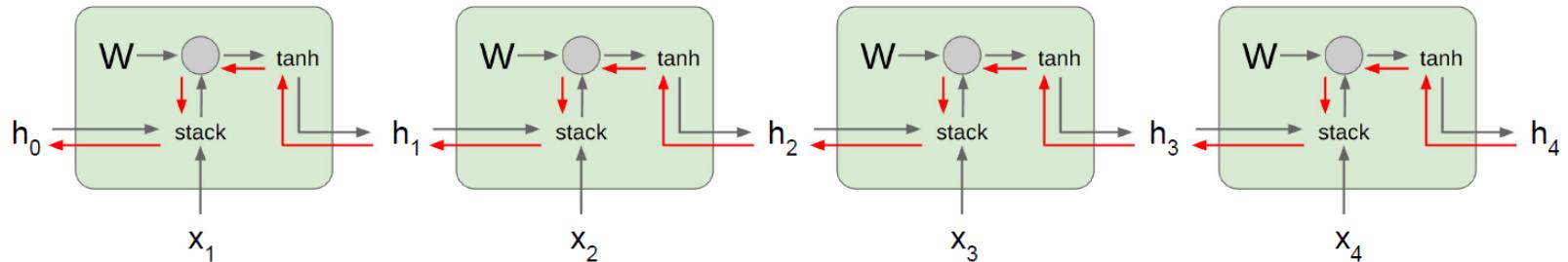
Recurrent Neural Networks (Part II)

SOUJANYA PORIA

50.038 Computational data science

Problems with Vanilla RNNs

- Vanishing and Exploding Gradients



- Unable to remember inputs from long ago

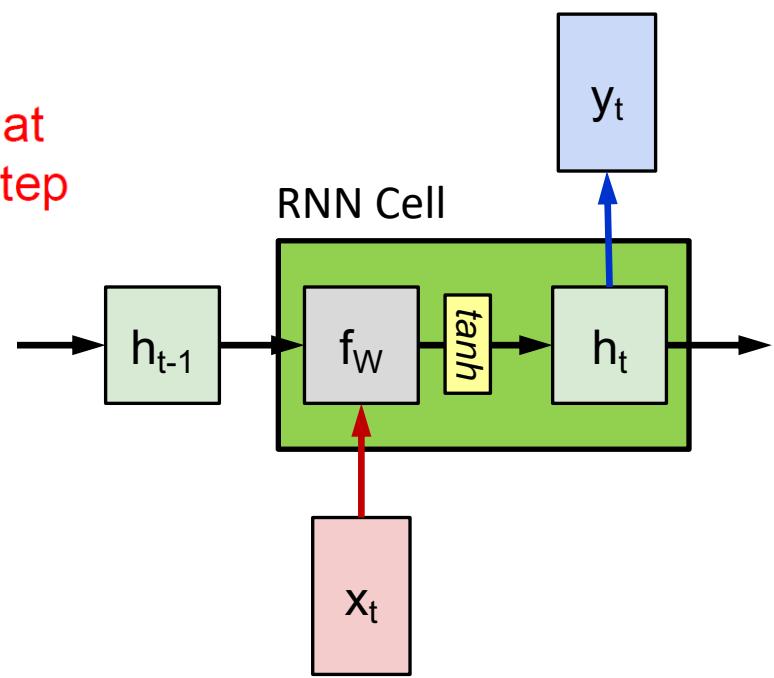
I live in France I speak fluent French.



Recap: A Vanilla RNN Cell

$$h_t = f_W(h_{t-1}, x_t)$$

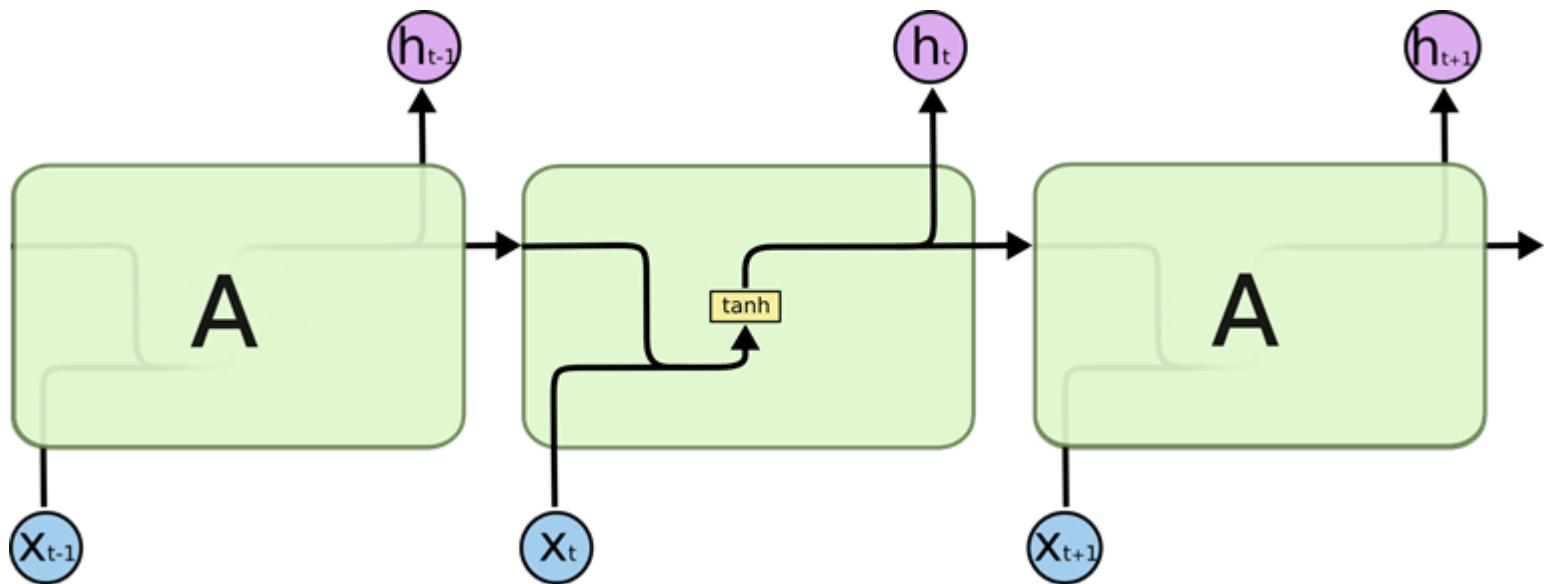
new state / old state input vector at
some function | some time step
with parameters W



$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

Recap: A Vanilla RNN Unit



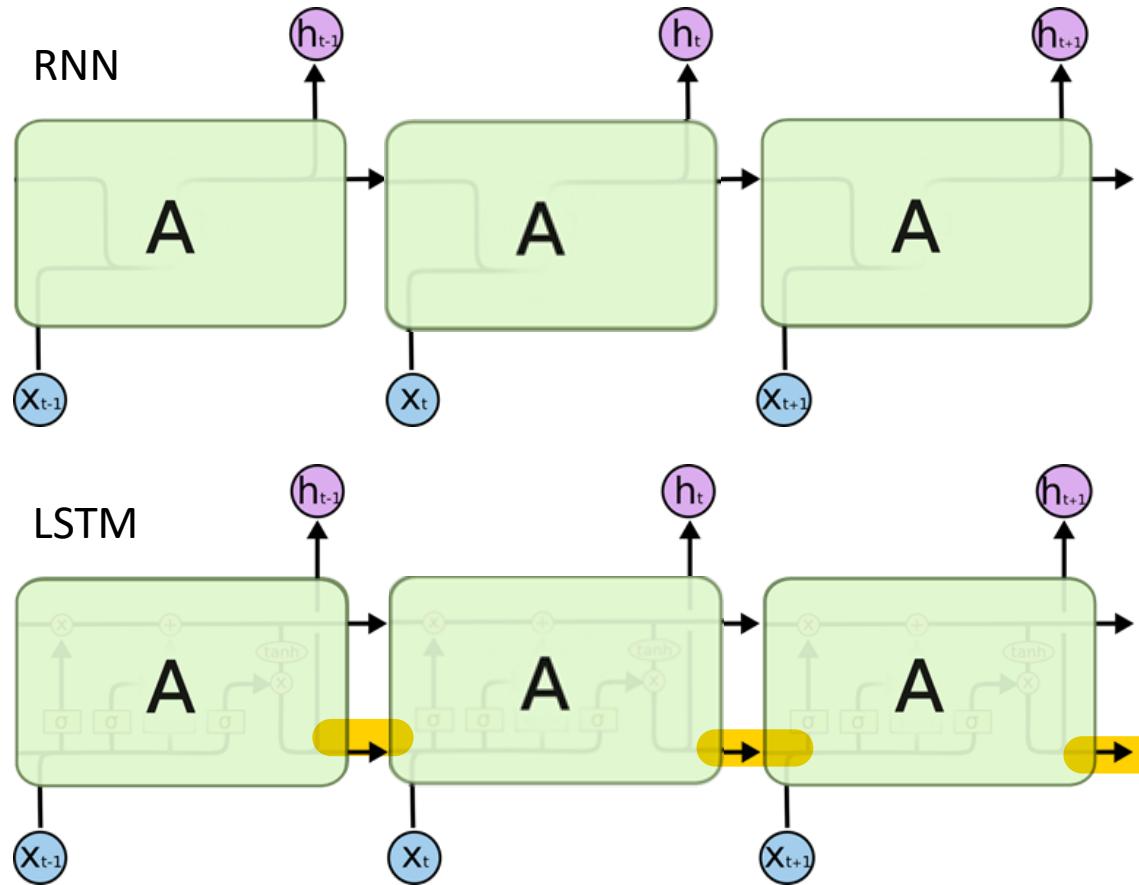
Long Short-Term Memory

- Long Short-Term Memory (LSTM) were designed to overcome problems of vanilla RNNs
 - i.e., vanishing/exploding gradients, long-term dependencies
- A LSTM cell/unit comprises the following:
 - Cell State
 - Forget Gate
 - Input Gate
 - Output Gate
- Contrast this to the vanilla RNN cell



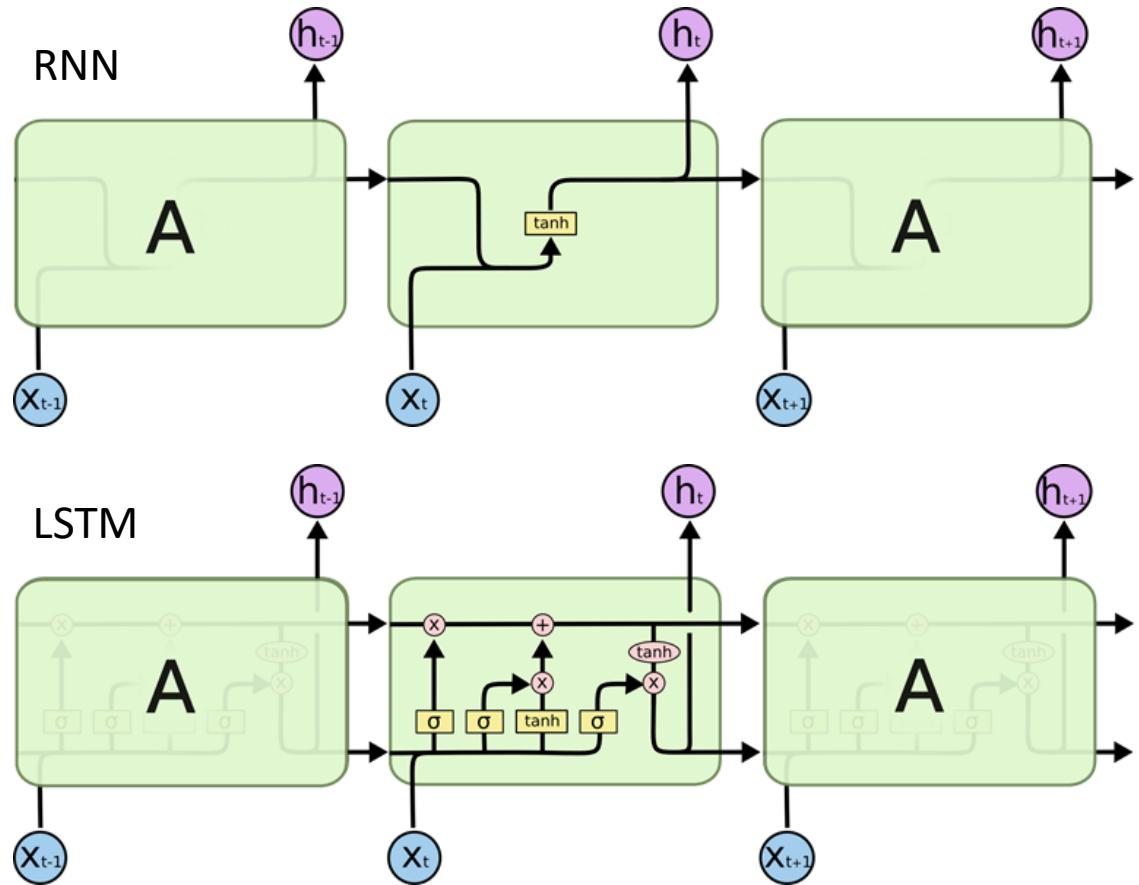
Exercise 4: Vanilla RNN vs LSTM

- Based on an overview of both architectures, what similarities do you observe?



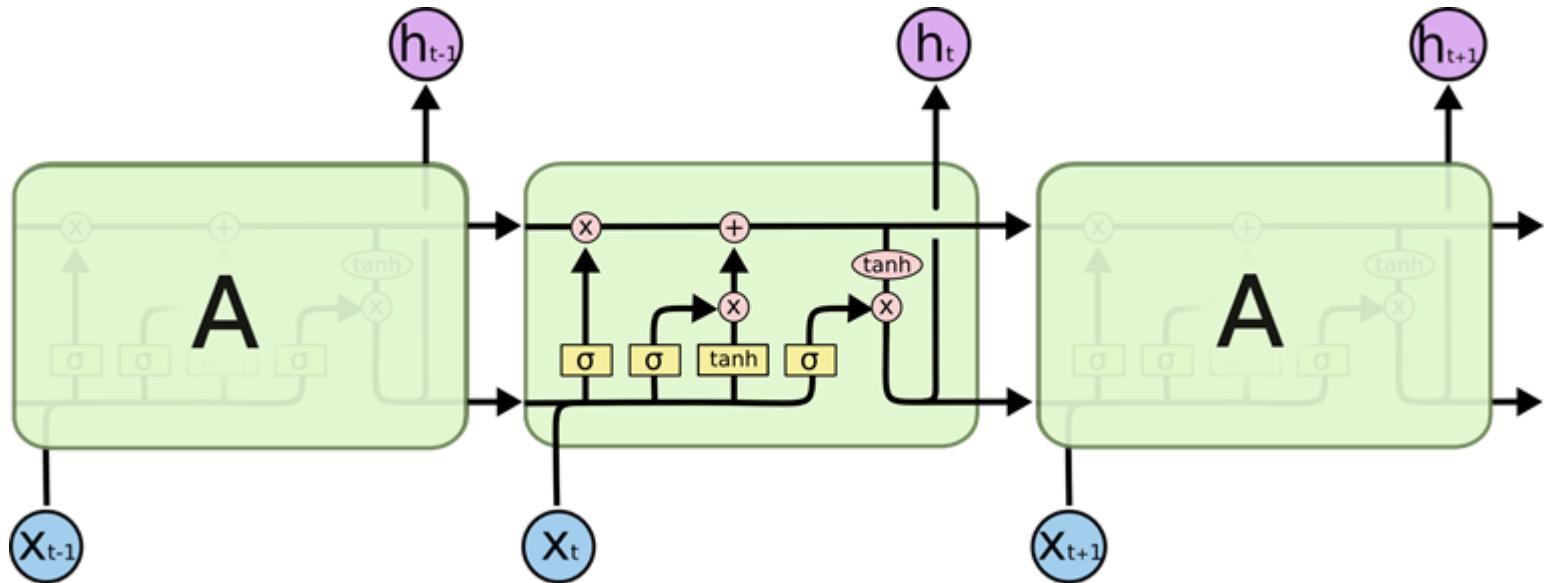
Vanilla RNN vs LSTM

- Differences in terms of internal cell operations
 - Vanilla RNN only considers the previous state h_{t-1} and current input x_t
 - LSTM has the three gates and cell state



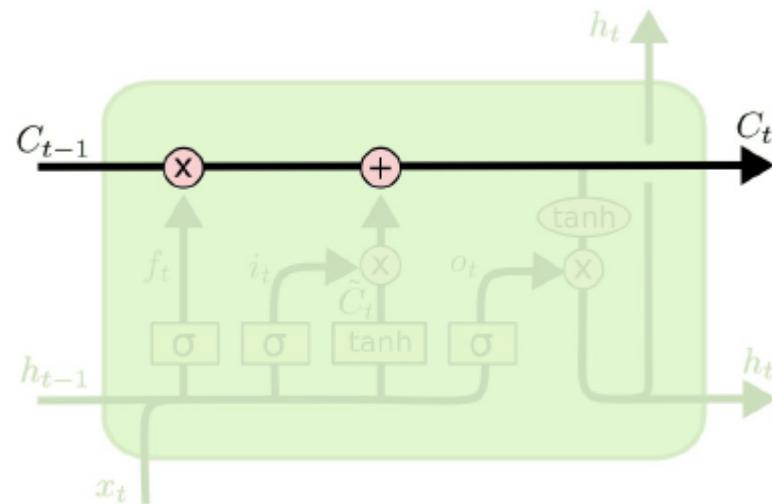
LSTM Components

- Cell state, Forget gate, Input gate, Output gate



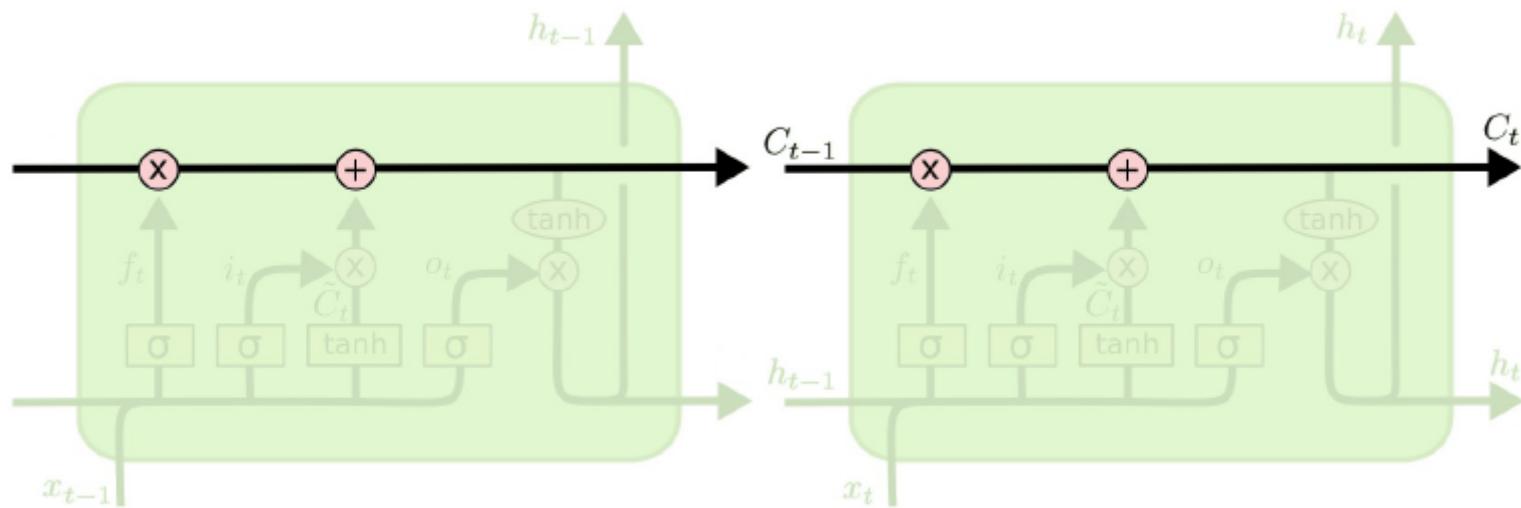
LSTM Components

- Cell State
 - One of the key difference from vanilla RNNs



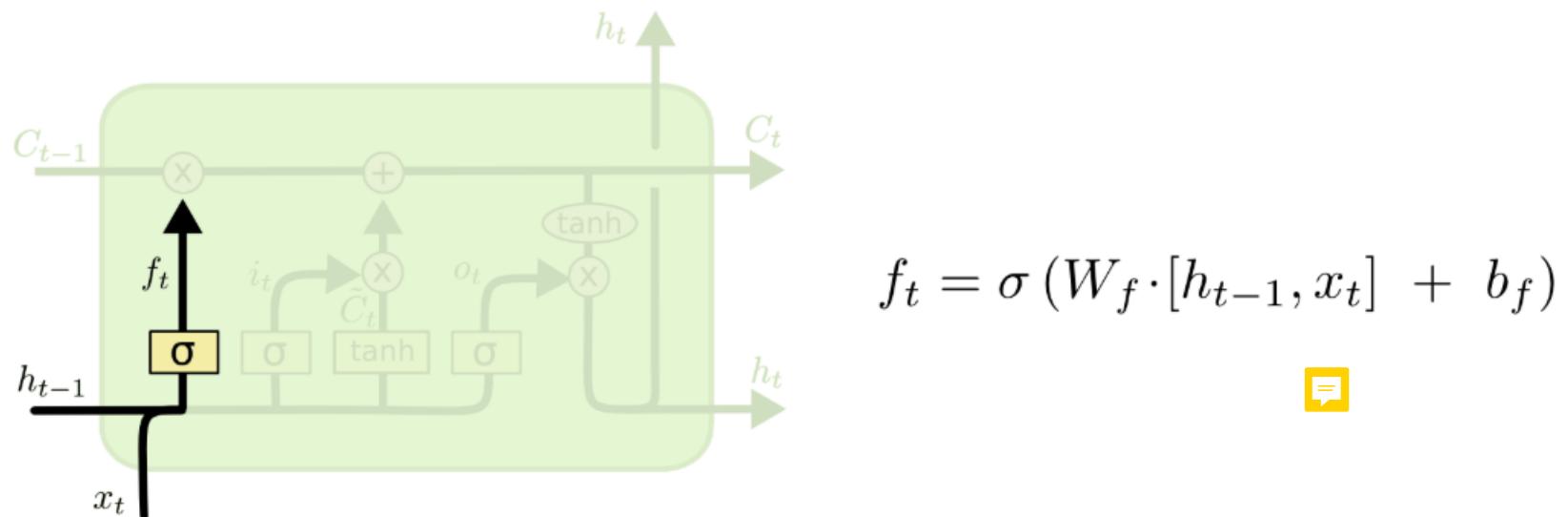
LSTM Components

- Cell State
 - One of the key difference from vanilla RNNs
 - Serves like an information highway through all LSTM cells
 - How is information added to this cell state?



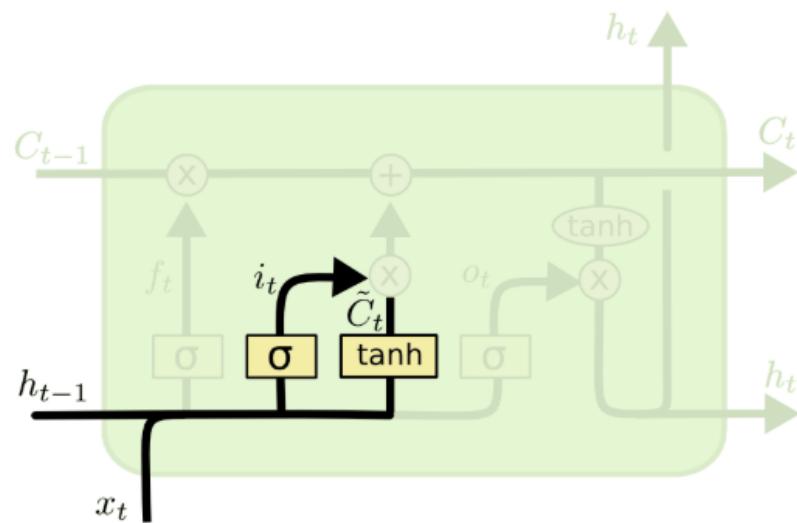
LSTM Components

- Forget Gate
 - Determines which part of the previous cell state to remember/forget
 - The sigmoid function allows us to completely forget (0) or remember (1)



LSTM Components

- Input Gate
 - Determine what new information to write to cell state
 - Sigmoid to choose what to write, tanh to generate the candidate values

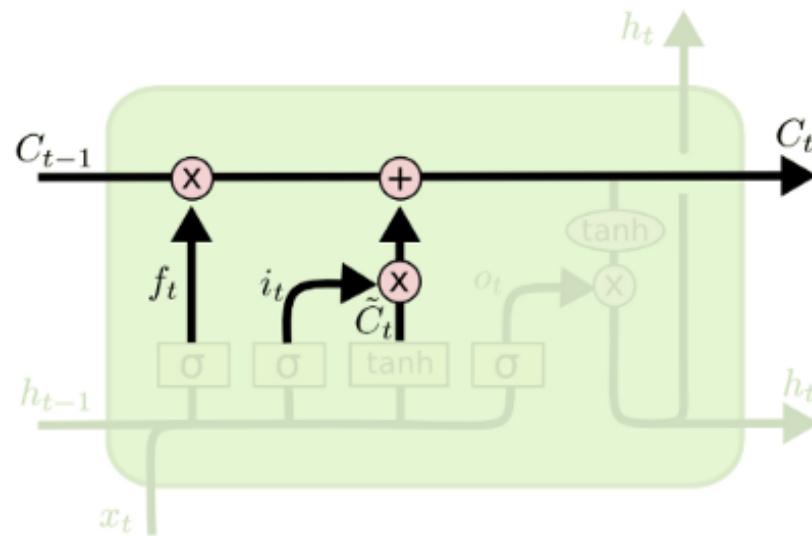


$$i_t = \sigma (W_i \cdot [h_{t-1}, x_t] + b_i)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$



LSTM Components

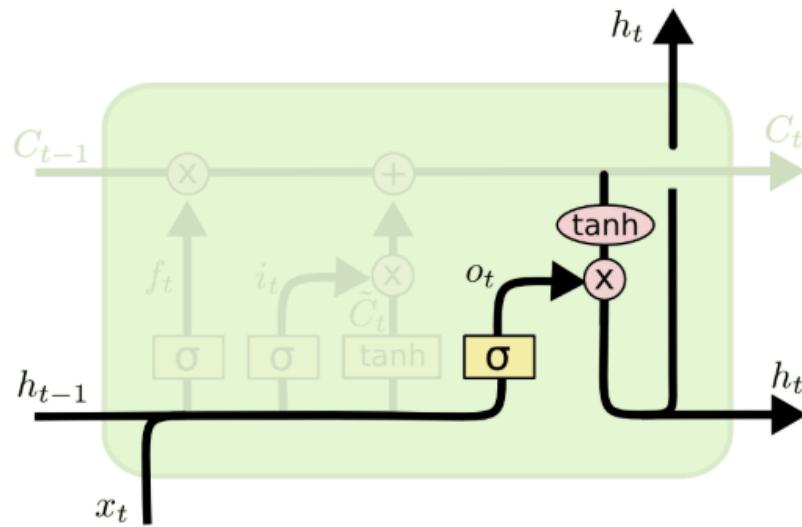
- Cell State – Forget and Input
 - Operations to update the cell state from C_{t-1} to C_t
 - Based on the previous Forget gate and Input gate



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

LSTM Components

- Output Gate
 - Determines which part of the new cell state to output
 - The output h_t can then be passed to the next LSTM cell and/or used to generate a prediction at timestep $t+1$



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

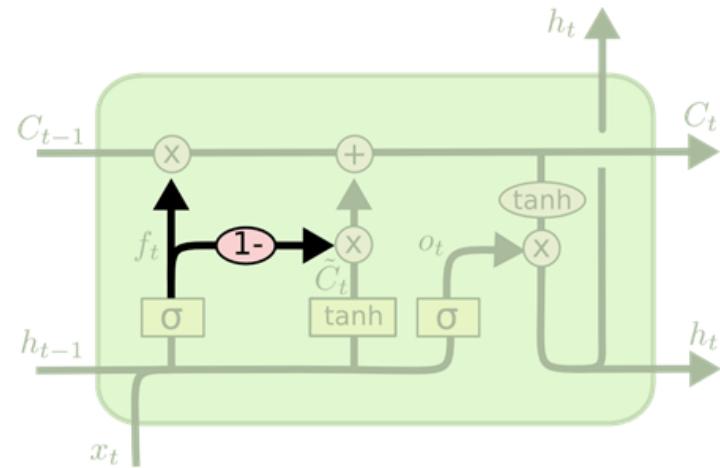
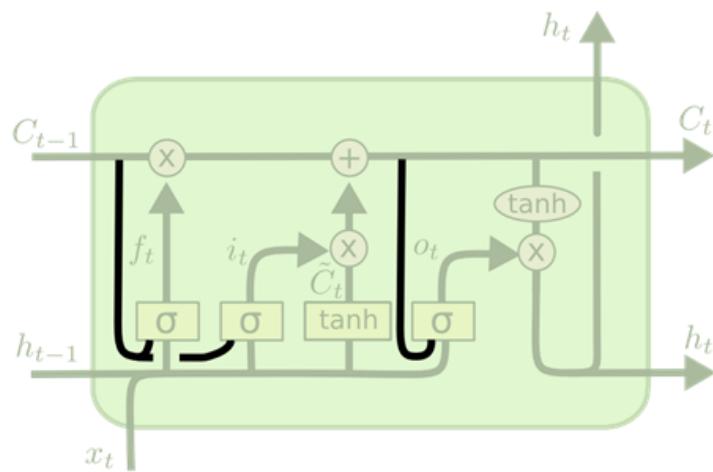
$$h_t = o_t * \tanh (C_t)$$

Other LSTM Variants

- LSTM with Peepholes

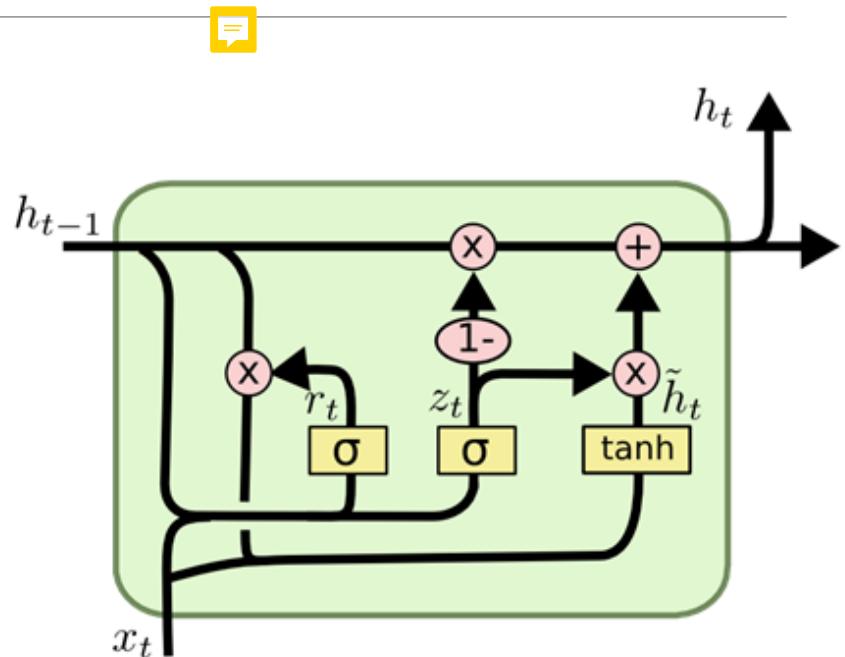
- Allow the three gates to “peep” into the cell state

- LSTM with combined Forget/Input gates



Other Variants

- Gated Recurrent Units
 - A simplified version of LSTM
 - Has no independent cell state
 - Has a Reset gate (r_t)
 - Determines how to combine the new input with the previous memory
 - Has a Update gate (z_t)
 - Determines how much of the previous memory to keep around
 - Combines the Forget and Input gates of LSTM



Chung, J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning*, December 2014.

LSTM in Keras

- Initialize a sequential model (layers added one after another)

```
model = Sequential()
```

- For a task like text classification, add an embedding layer to convert words to a dense vector (e.g., of dimension 256)

```
model.add(Embedding(max_features, output_dim=256))
```

- Add in a LSTM layer that outputs a dense vector of dimension 128

```
model.add(LSTM(128))
```

- Add regularization, if necessary

```
model.add(Dropout(0.5))
```

LSTM in Keras

- Add in a final fully-connected layer to predict labels (change number of nodes and activation function, as required)

```
model.add(Dense(1, activation='sigmoid'))
```

- Define parameters for training the defined model

```
model.compile(loss='binary_crossentropy',
               optimizer='rmsprop',
               metrics=['accuracy'])
```

LSTM in Keras

- Training the model

```
model.fit(x_train, y_train, batch_size=16, epochs=10)
```

- Testing the model

```
score = model.evaluate(x_test, y_test, batch_size=16)
```

Application: Video Question Answering

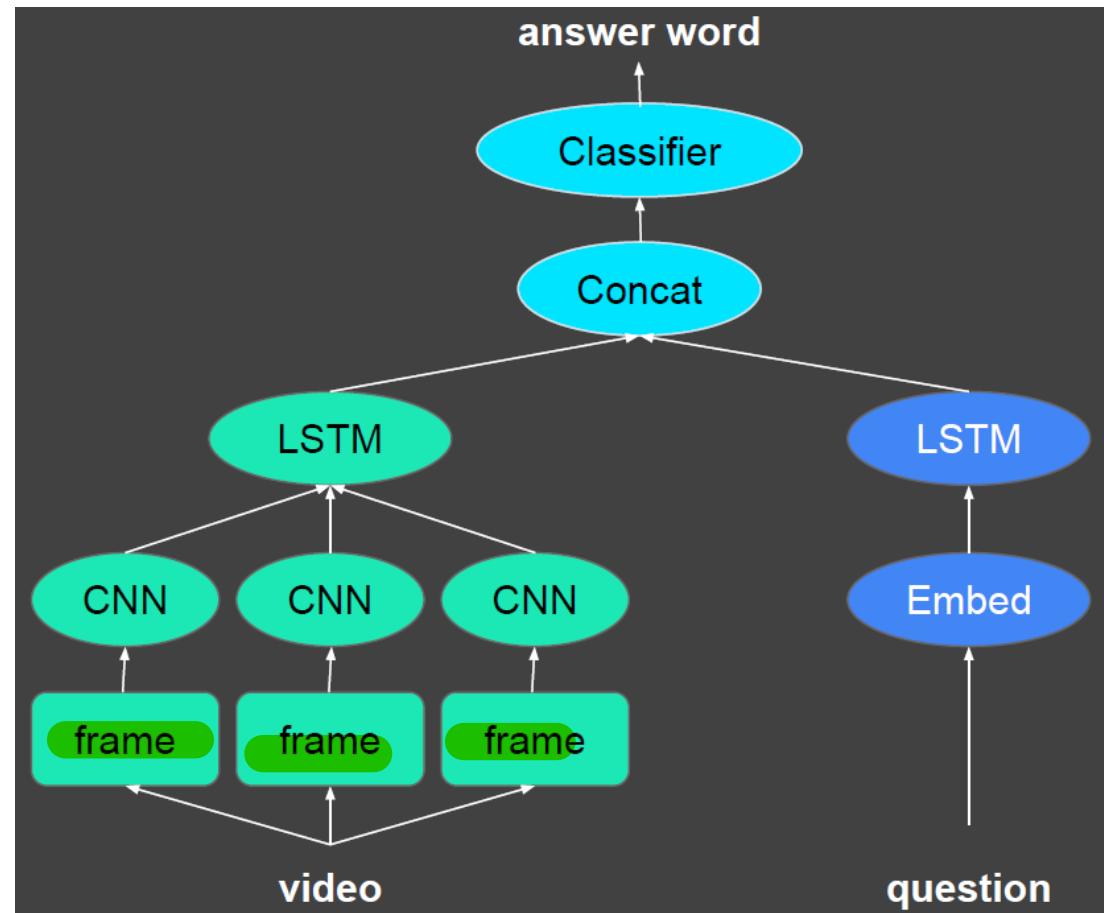


- > "**What is the man doing?**"
- > **packing**

- > "**What color is his shirt?**"
- > **blue**

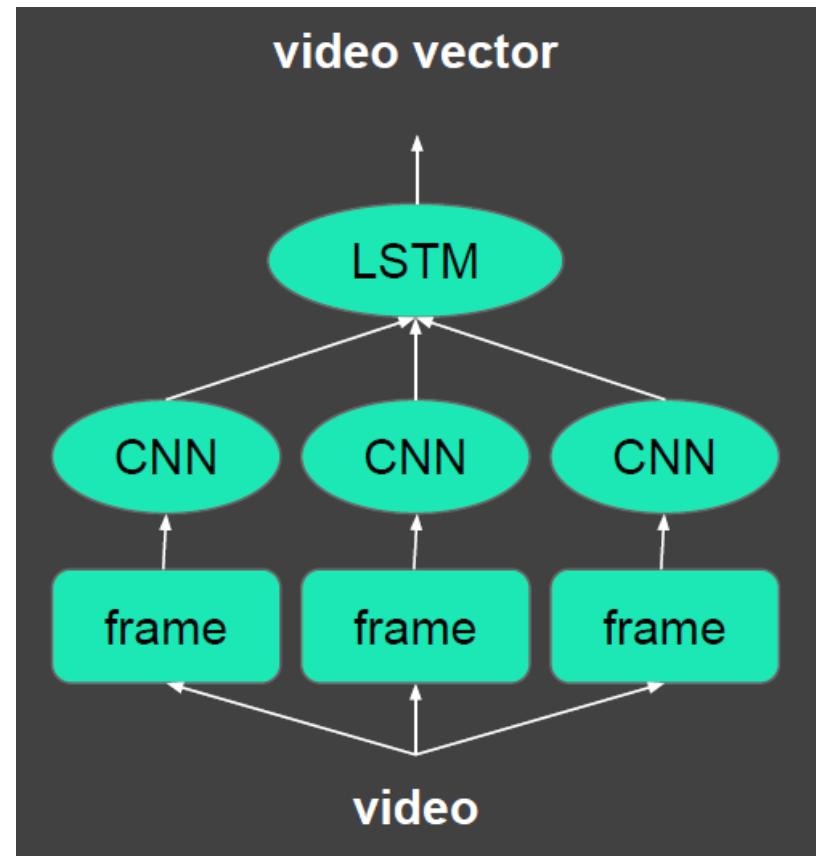
Application: Video Question Answering

- Utilizing MLP, CNN, LSTM and embeddings
 - One component for video
 - One component for question



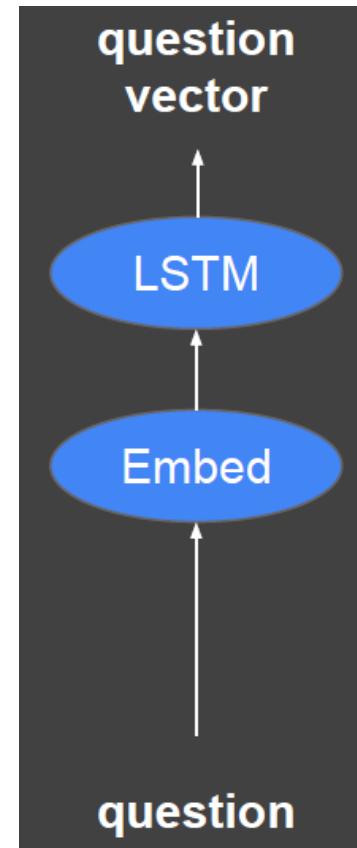
Application: Video Question Answering

- Video component
 - Main idea is to represent video as a dense vector
 - Video split into frames, each frame represented as a dense vector
 - Each “frame” then provided as input to LSTM as a temporal sequence



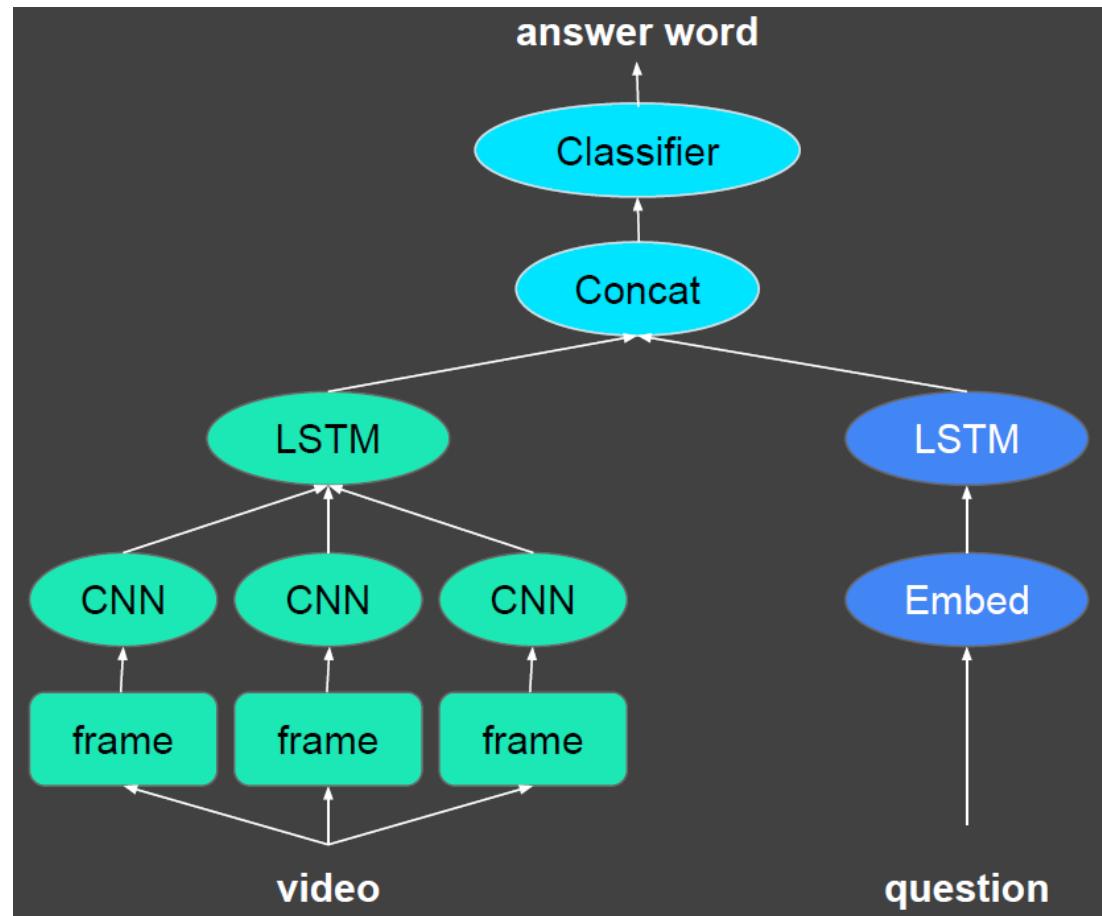
Application: Video Question Answering

- Question component
 - Main idea is to represent question as a dense vector
 - Question is split into words, each represented by a word embedding
 - Each word embedding then provided as input to LSTM as a temporal sequence



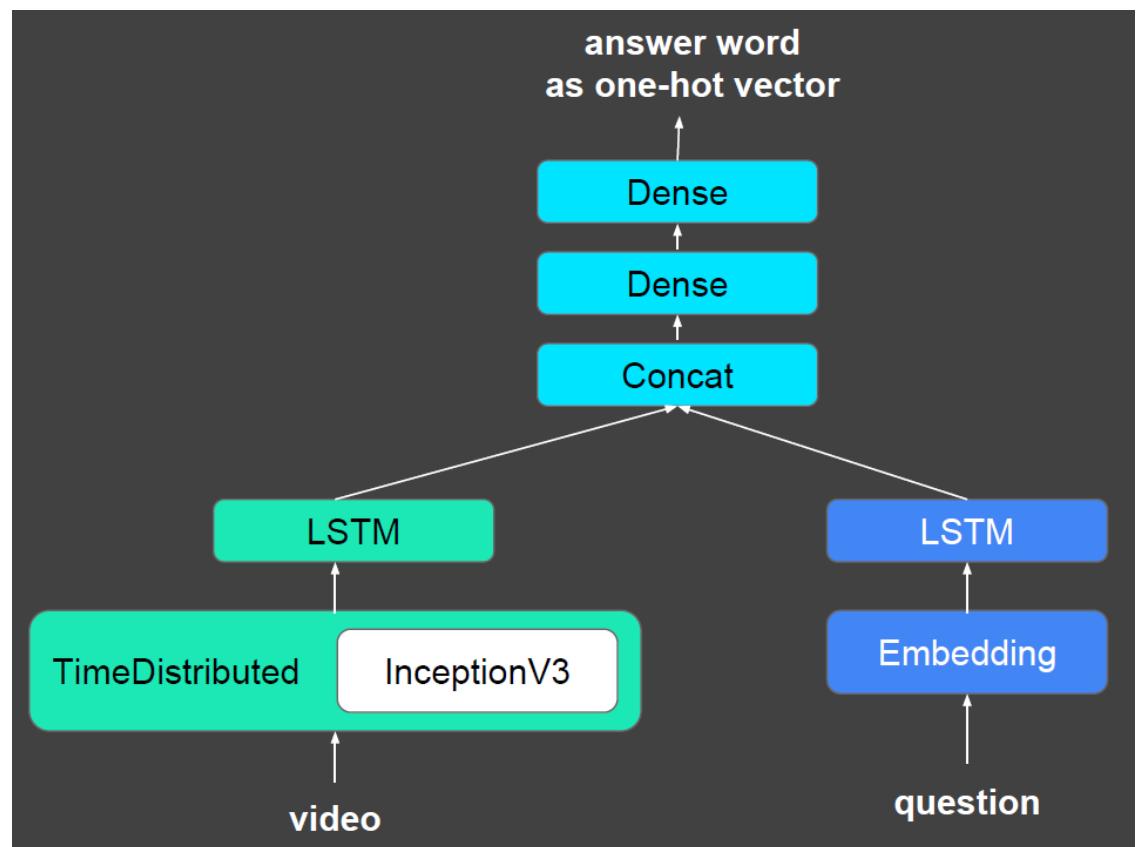
Application: Video Question Answering

- Utilizing MLP, CNN, LSTM and embeddings
 - Dense vectors for video and question components are concatenated
 - Concatenated vector then provided as input to a MLP classifier



Application: Video Question Answering

- Overview of implementation in Keras



Application: Video Question Answering

- Video component
 - Utilizing pre-trained weights to convert video to dense vectors

```
import keras
from keras import layers
from keras.applications import InceptionV3

video = keras.Input(shape=(None, 150, 150, 3), name='video')
cnn = InceptionV3(weights='imagenet',
                    include_top=False,
                    pooling='avg')
cnn.trainable = False
frame_features = layers.TimeDistributed(cnn)(video)
video_vector = layers.LSTM(256)(frame_features)
```

Application: Video Question Answering

- Question component
 - Converting question to a sequence of word (embeddings)

```
question = keras.Input(shape=(None,), dtype='int32', name='question')
embedded_words = layers.Embedding(input_voc_size, 256)(question)
question_vector = layers.LSTM(128)(embedded_words)
```

Application: Video Question Answering

- Answer prediction
 - Concatenating video and question vector
 - Predicting answer using a MLP

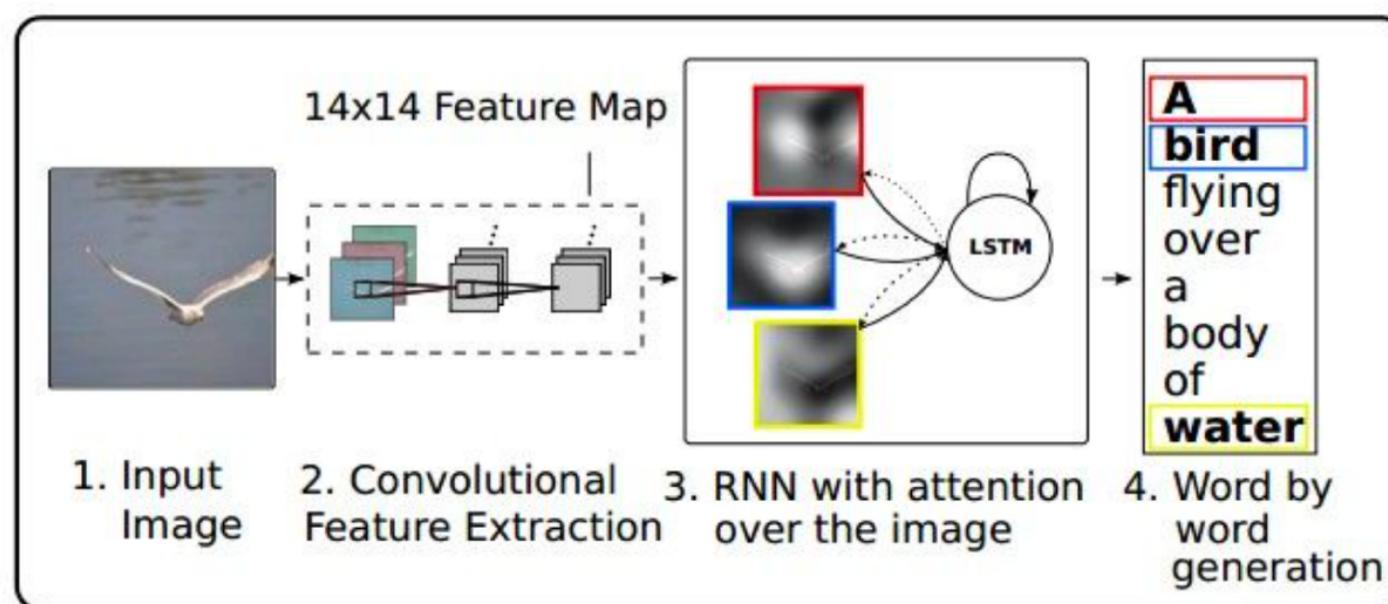
```
x = layers.concatenate([video_vector, question_vector])
x = layers.Dense(128, activation=tf.nn.relu)(x)
predictions = layers.Dense(output_voc_size,
                           activation='softmax',
                           name='predictions')(x)
```

Other RNN Applications

Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

RNN focuses its attention at a different spatial location when generating each word



Other RNN Applications

Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

Soft attention



Hard attention

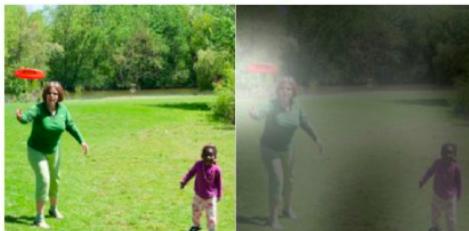


Other RNN Applications

Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

Good results



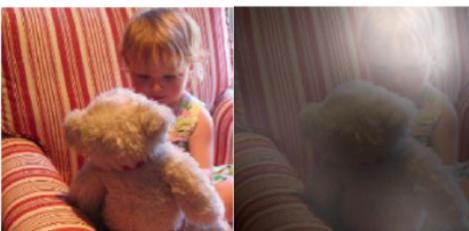
A woman is throwing a frisbee in a park.



A dog is standing on a hardwood floor.



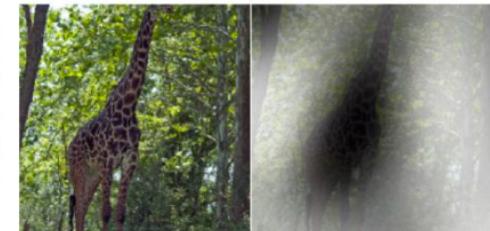
A stop sign is on a road with a mountain in the background.



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.

Other RNN Applications

Applications: Image Captioning with Attention

[Xu et al., ICML 2015]

Failure results



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.

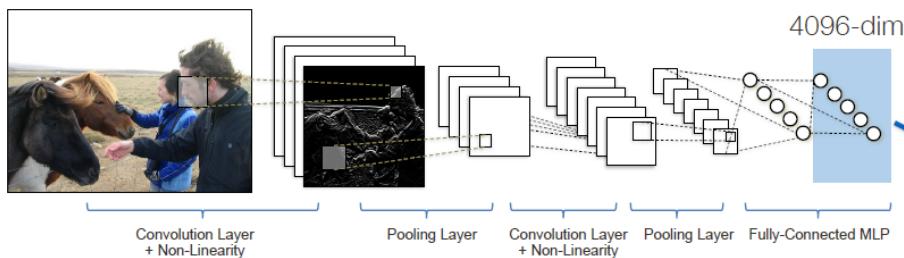


A man is talking on his cell phone while another man watches.

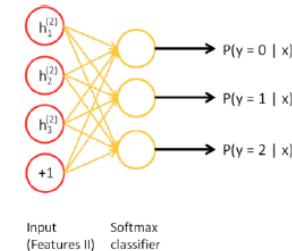
Other RNN Applications

Applications: Typical Visual Question Answering (VQA)

Image Embedding (VGGNet)

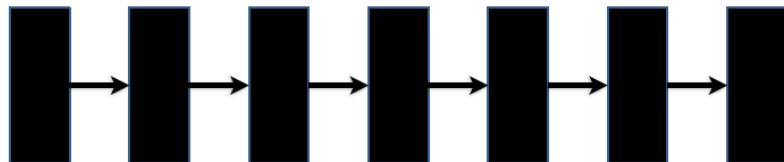


Neural Network
Softmax
over **top K answers**



Question Embedding (LSTM)

"How many horses are in this image?"



Other RNN Applications

Applications: Activity Detection

[Ma et al., 2014]

Activity: A collection of human/object movements with a particular semantic meaning



Action Recognition: Finding if a video segment contains such a movement

Action Detection: Finding a segment (beginning and start) and recognize the action in it

Summary

- Provided an overview of the types of RNNs
- Discussed about vanilla RNNs and its limitation
- Discussed about how LSTM works
- Have an overview of the different variants of LSTMs
- Studied various applications of RNNs/LSTMs

References

- http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture10.pdf
- http://introtodeeplearning.com/materials/2018_6S191_Lecture2.pdf
- <https://www.cs.toronto.edu/~hinton/csc2535/notes/lec10new.pdf>
- <https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21>
- <https://towardsdatascience.com/introduction-to-sequence-models-rnn-bidirectional-rnn-lstm-gru-73927ec9df15>
- <http://colah.github.io/posts/2015-08-Understanding-LSTMs/>
- <https://web.stanford.edu/class/cs20si/lectures/march9guestlecture.pdf>
- <https://www.cs.ubc.ca/~lsigal/532L/Lecture8.pdf>