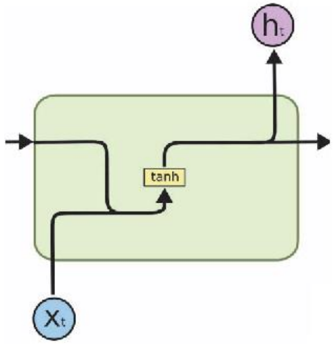
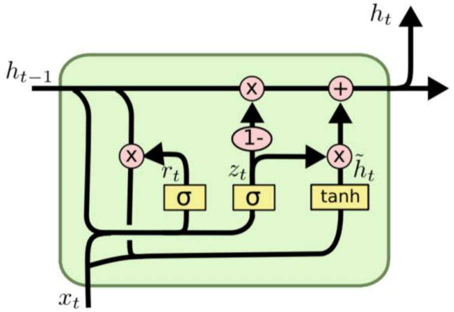
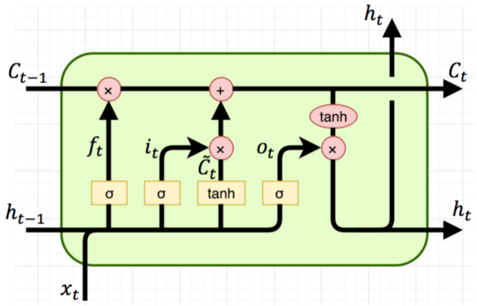
Deep Learning for Natural Language Processing







Developed by Titus Tang 2020

Workshop Itinerary

Time	Event
10:00 – 11:00	Lecture-Demo 1 – Text Processing
11:00 – 12:00	Workshop 1 – Text Processing
12:00 – 13:00	Lunch Break
13:00 – 14:00	Lecture 2 – Recurrent Neural Networks
14:00 – 15:00	Workshop 2 – Recurrent Neural Networks
15:00 – 15:30	Question session

Workshop Content – Part 1

- Learn to recognise text as data.
- Learn how to process text data using various software packages.
- Hands-on processing of text data for use in training a neural network.

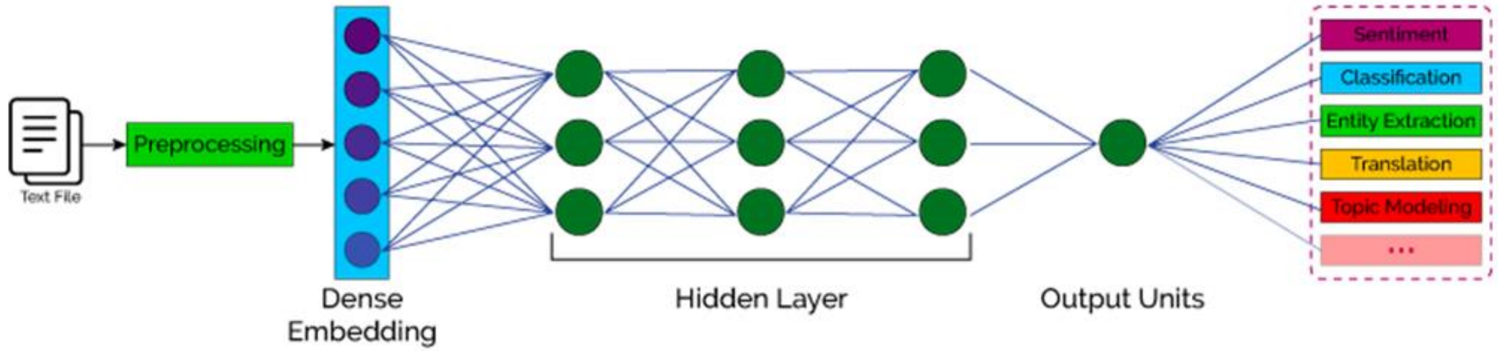
Workshop Content – Part 2

- Learn about recurrent neural networks and how they work.
- Build a recurrent neural network using TensorFlow.
- Train and test the network using the dataset that you have prepared.
- Explore the behaviour and capabilities of the trained model.

Assumed Knowledge

- Basic understanding of fundamental deep learning concepts.
- Basic Python programming skills.

How do neural networks process language (time series) data?

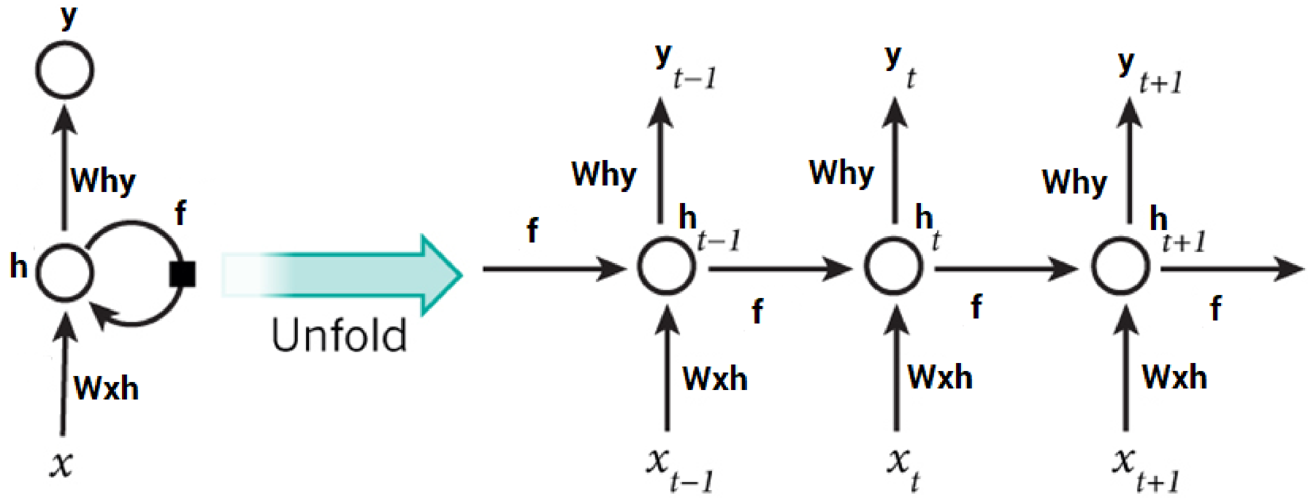


Converts words into numbers

- ✓ Clean data and define a scope of words: a dictionary.
- ✓ Machines only work with numbers: convert all words into numbers.
- ✓ Feed data into network one at a time.
- ✓ Network must have some concept of memory to remember past words.

In the second lecture this will be covered.

Recurrent Neural Networks



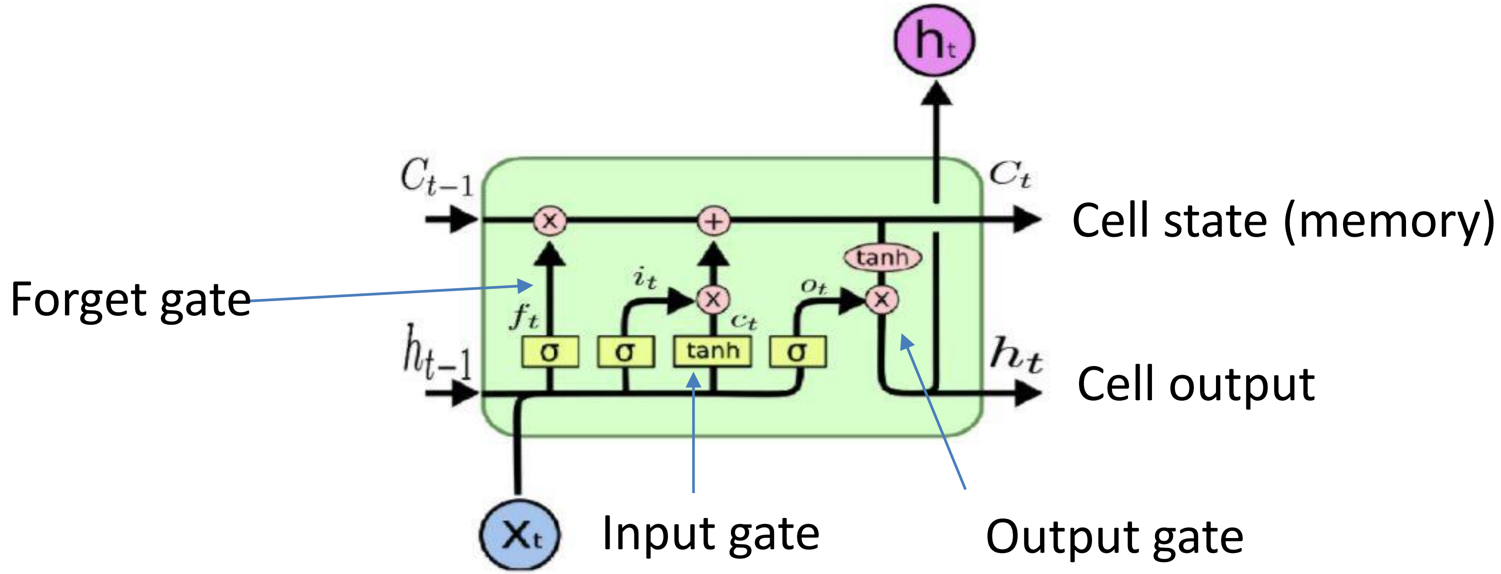
✓ Network design for dealing with time series data.

✓ Each node provides a feedback to itself over time.

Recurrent Neural Networks

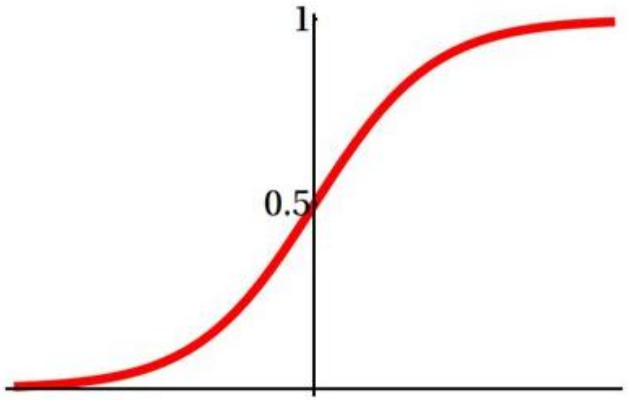
- ✓ This model is too simple.
- ✓ Needs a more sophisticated model that has:
  - ❖ Control over what is learned.
  - ❖ Ability to forget.
  - ❖ Control over what to output.
  - ❖ Ability to associate ideas related over both short and longer time frames.

Long Short Term Memory (LSTM)



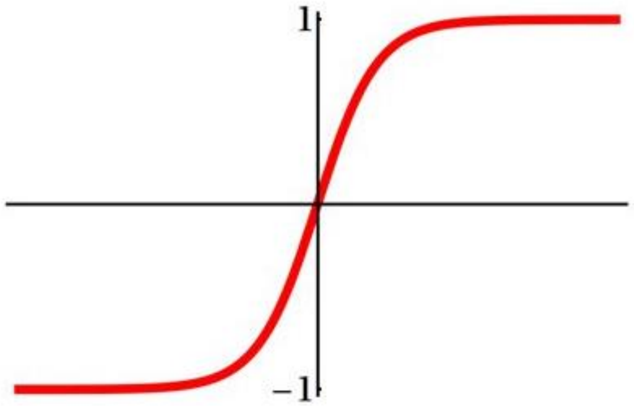
Avoids the issue of forgetting distant but important information.

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



logistic (sigmoid, unipolar)

$$\tanh(\Sigma) = \frac{e^{\Sigma} - e^{-\Sigma}}{e^{\Sigma} + e^{-\Sigma}}$$

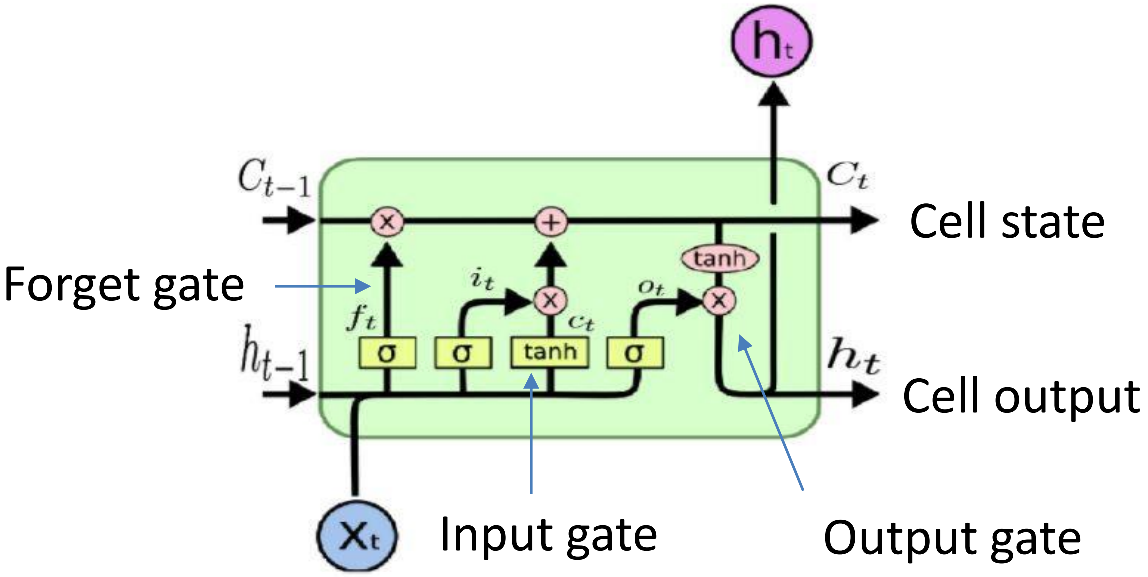


tanh (bipolar)

- ✓ Useful as a switch.
- ✓ Useful to select a portion of something.

- ✓ Squeeze all values down to -1 ←→ 1.
- ✓ Avoids numerical explosion.

Long Short Term Memory (LSTM)



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

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

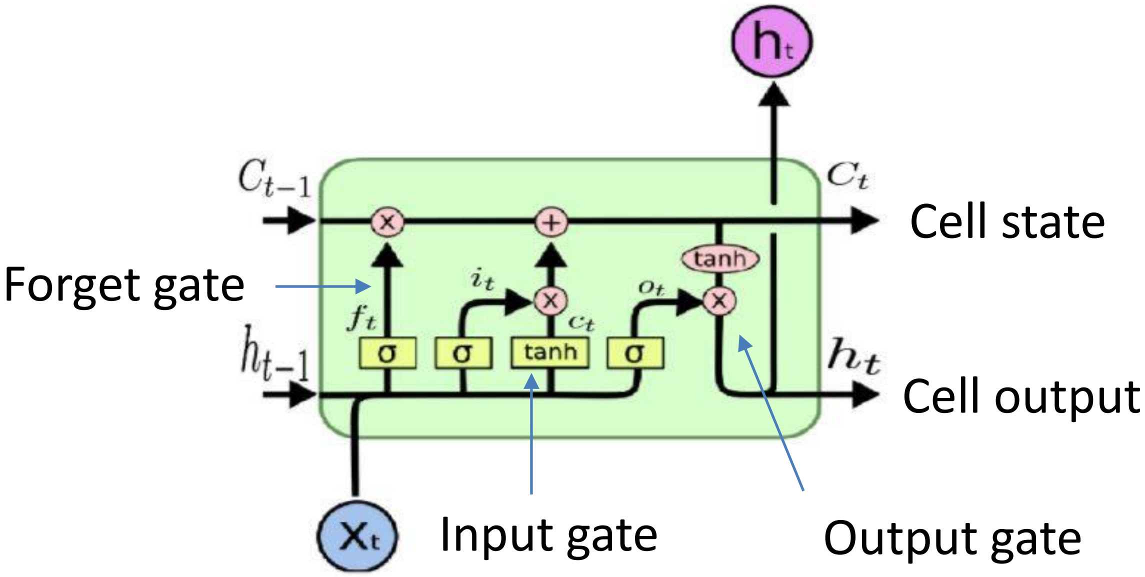
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

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

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

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

Long Short Term Memory (LSTM)



Portion to forget   Old info   New info

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

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

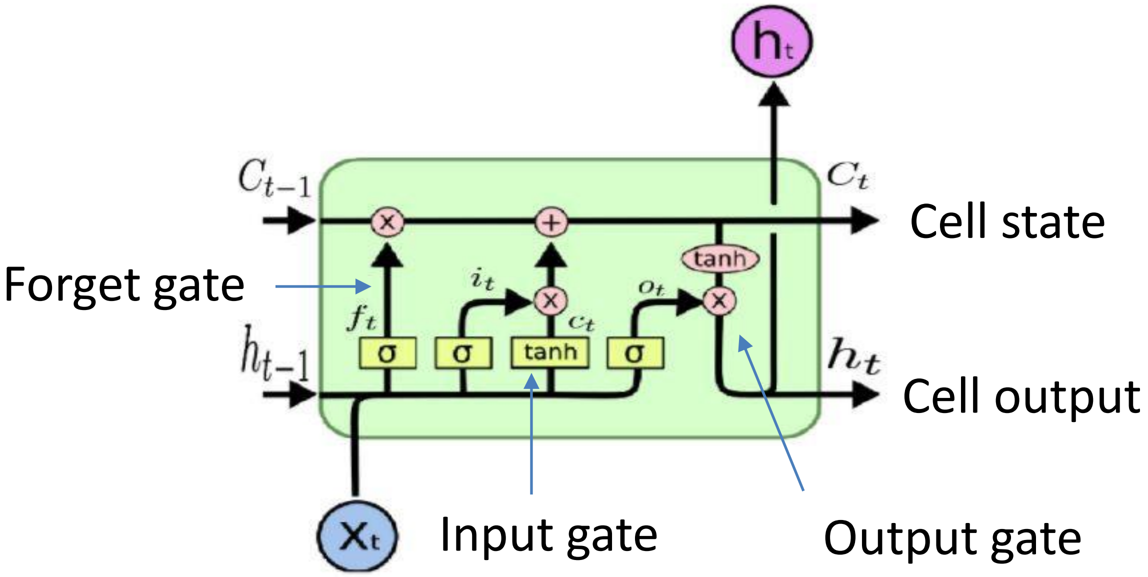
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \;+\; b_C)$$

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

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

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

Long Short Term Memory (LSTM)



Portion to forget   Old info   New info

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

Portion to learn

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

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

What to learn

Stuff in memory   New stuff learned

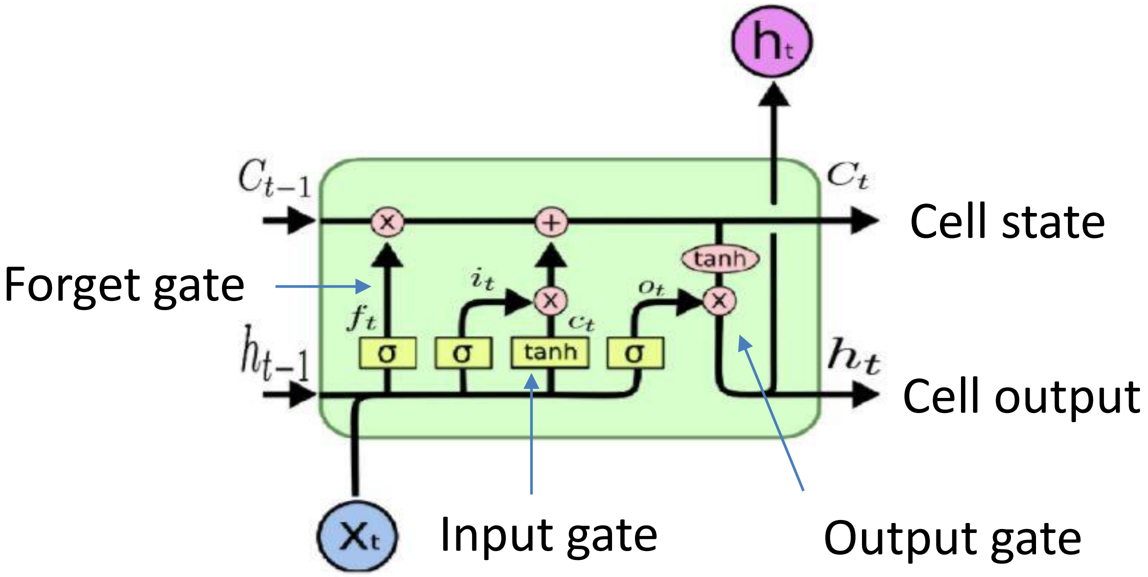
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Stuff remaining

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

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

Long Short Term Memory (LSTM)



Portion to forget Old info New info

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

Portion to learn

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

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

What to learn

Stuff in memory New stuff learned

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

Stuff remaining

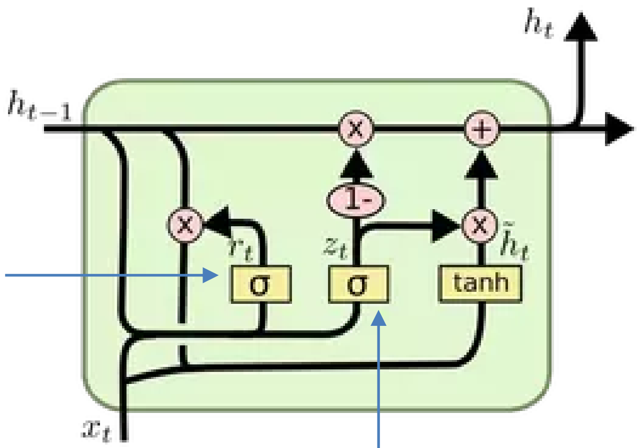
Portion to output

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

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

Output

Gated Recurrent Unit (GRU)



Reset gate

Update gate

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

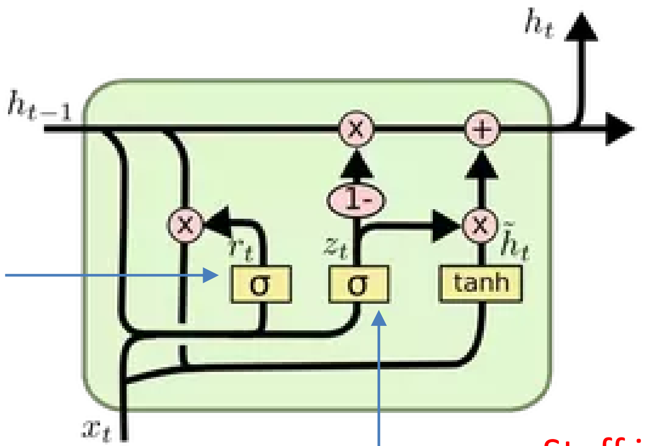
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Gated Recurrent Unit (GRU)

Reset gate



Update gate

$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$ Portion to update

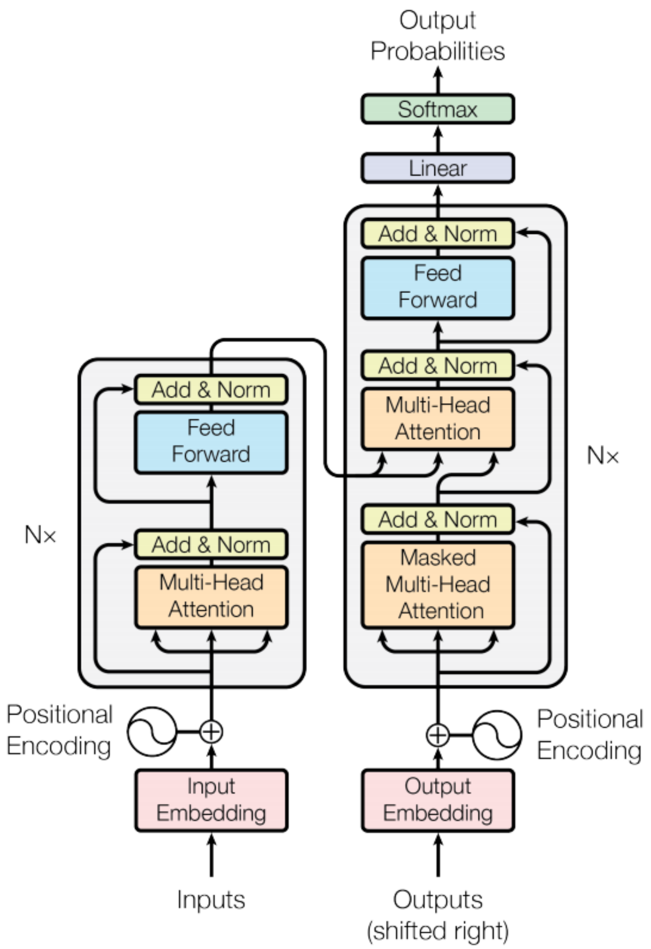
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$ Portion to reset

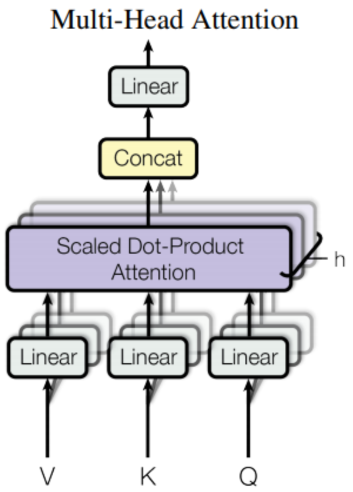
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$ Existing old and new info

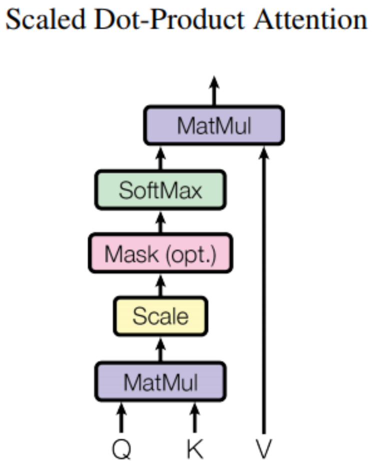
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Stuff in Memory — Old info — New info

BERT : Bidirectional Encoder Representations from Transformers

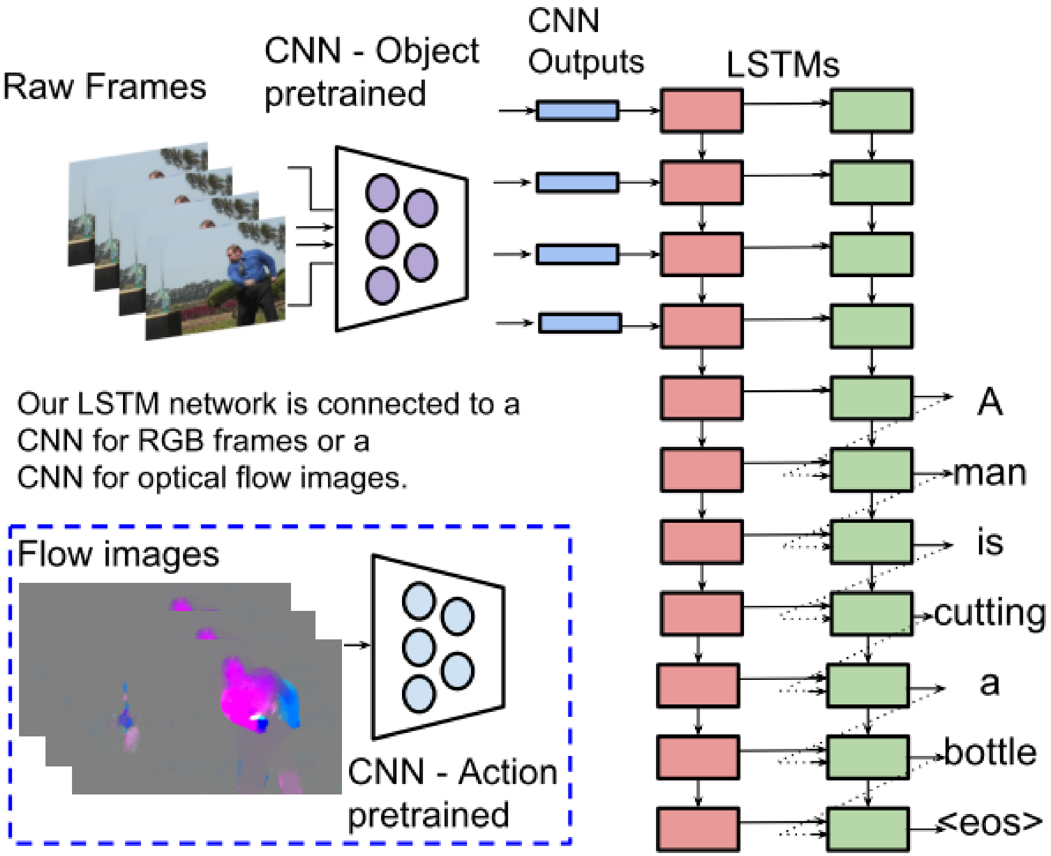






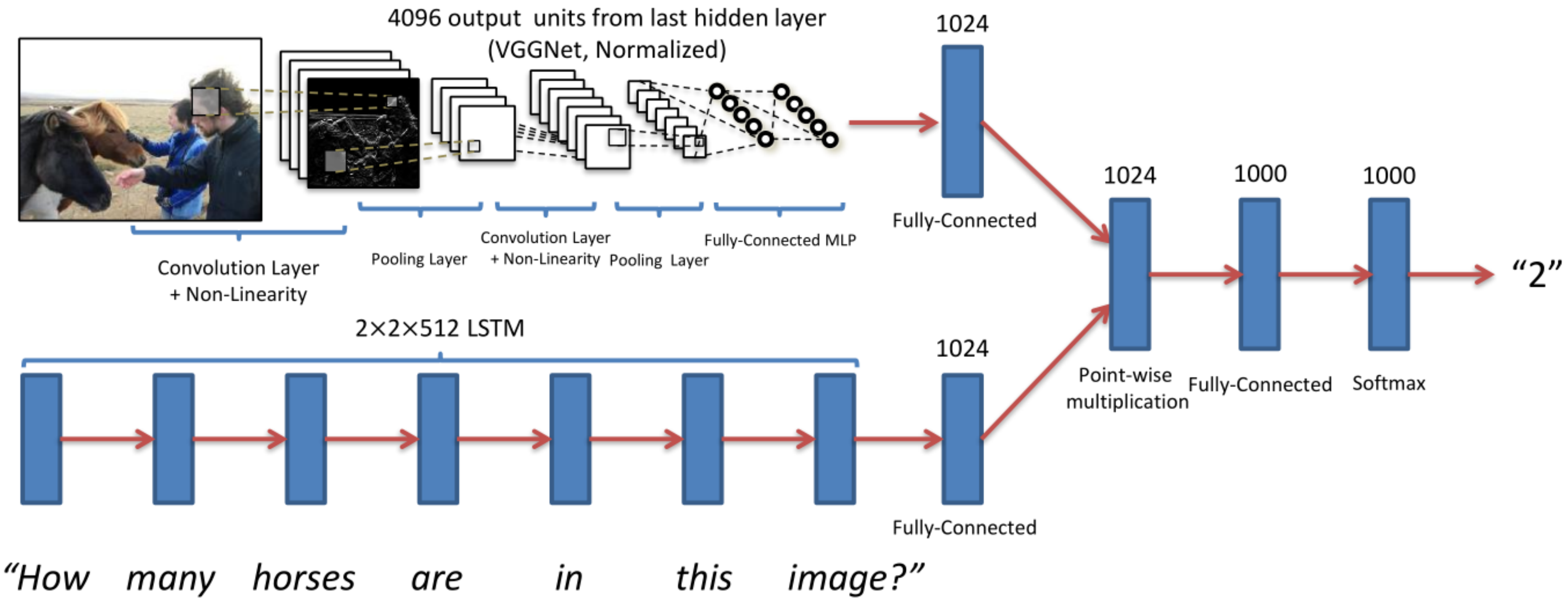
- Attention mechanism learns what features to pay attention to.
  - ✓ V = Values to pay attention to.
  - ✓ K = Keys weighing importance of values depending on context.
  - ✓ Q = Query – the current context.

Combining CNNs with LSTMs



- ✓ Learning how to generate captions for videos.
- ✓ A CNN summarises information in the image.
- ✓ LSTM predicts next word based on image and previous word.

Visual question answering



CNN + LSTM + Fully-connected layers to answer questions based on visual context.