Introduction to Generative AI
(GAI 602)

WEEK 3

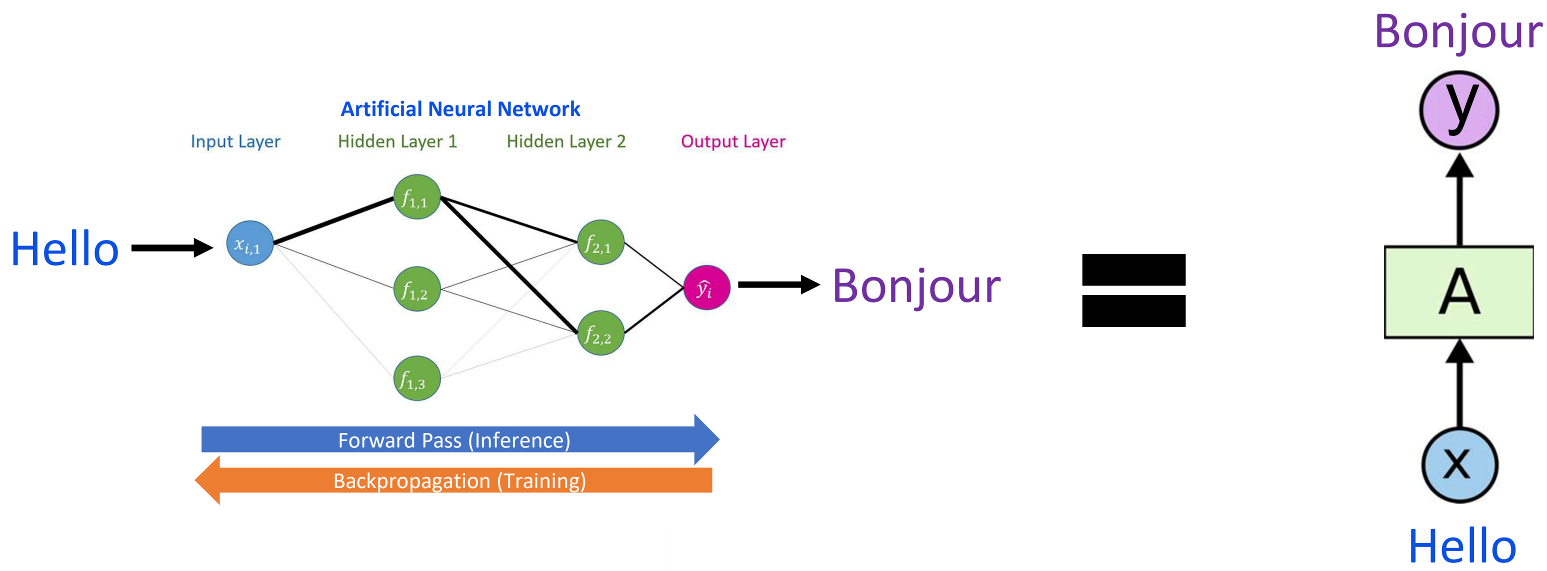
Sequence to Sequence Models

Use-cases for Seq-2-Seq models

- Named entity recognition
- Natural language generation
- Music composition
- Handwriting recognition
- Language identification
- Paraphrase detection
- Speech recognition

Sequence to Sequence Models

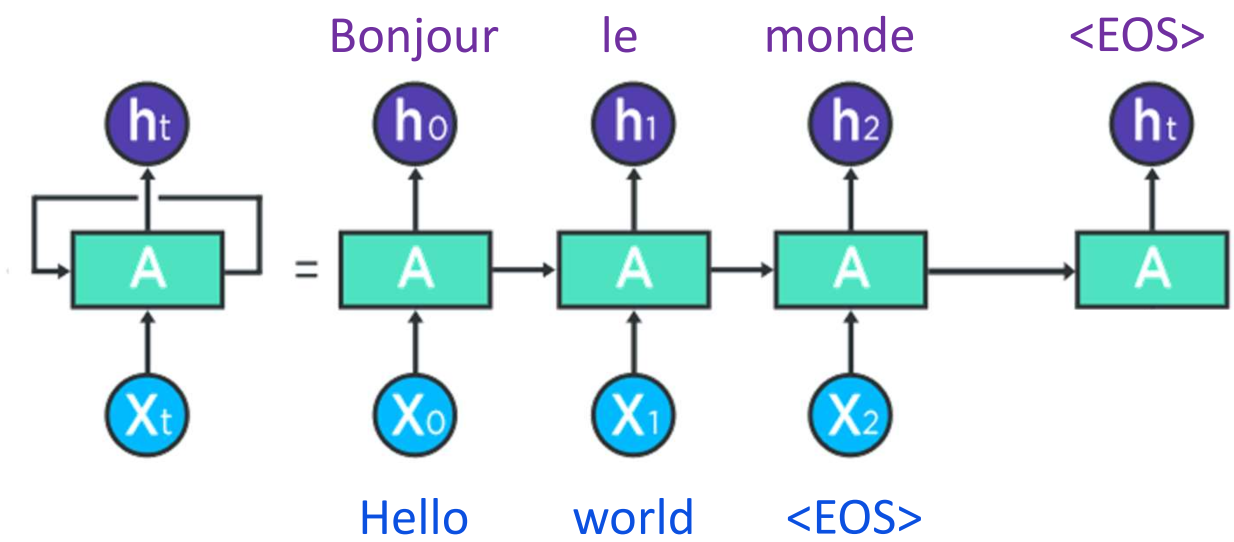
Let's say we want to translate **"Hello world"** into French.



Recurring Neural Network (RNN)

We need a way to remember the **context** to help generate the right word.

RNNs support the concept of **memory**, supporting data sequences



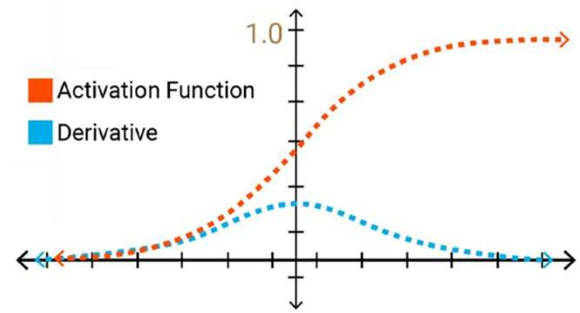
Vanishing Gradient Problem

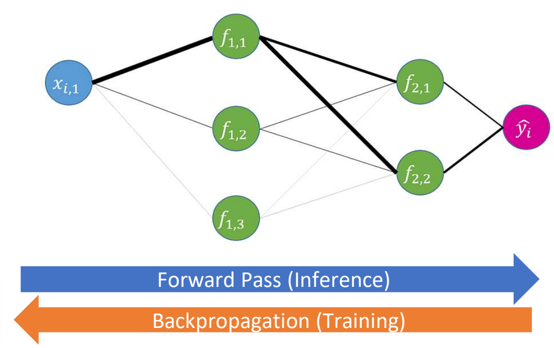
- **Deep neural networks use backpropagation.**
- **Back propagation uses the chain rule.**
- **The chain rule multiplies derivatives.**
- **Often these derivatives between 0 and 1.**

  As the chain gets longer, products get smaller until they disappear

or…

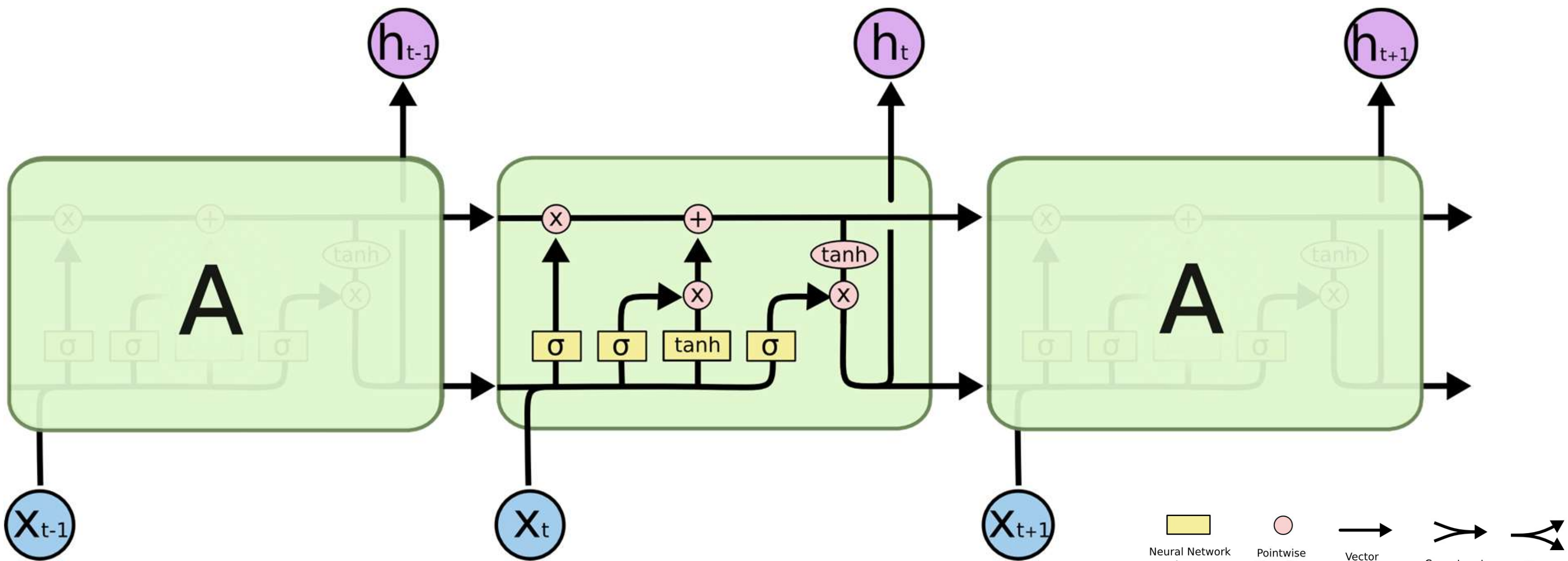
  As the chain gets longer, products get larger until they explode





Long short-term memory (LSTM) model

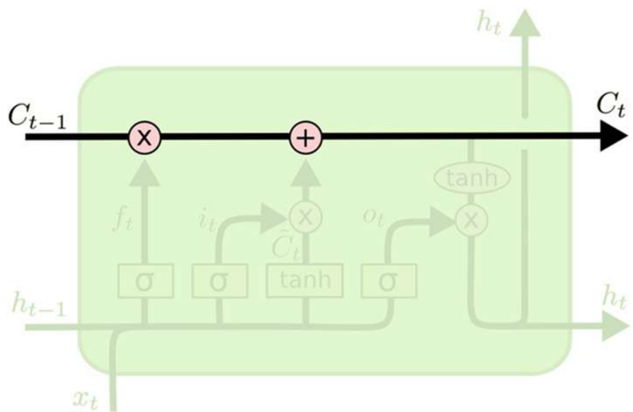
- LSTM uses **memory cells** to store information at each time step
- Uses **gates** to control the flow of information through the network
- How it works:
  - **Forget gate**: limit information passed from one cell to the next
  - **Input gate**: protect the current step from irrelevant inputs
  - **Output gate**: prevent the current step from passing irrelevant outputs to later steps



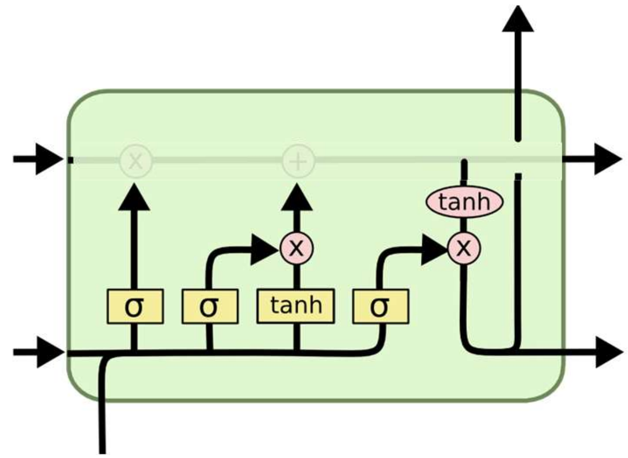
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

LSTM: Key Concepts

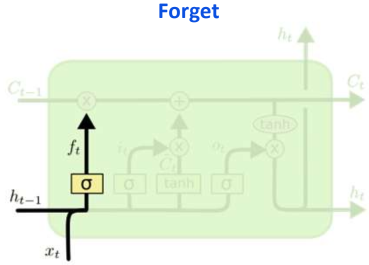
A **Backbone** to carry state forward and gradients backward



Gating to modulate information flow

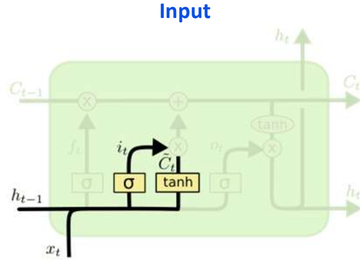


Forget



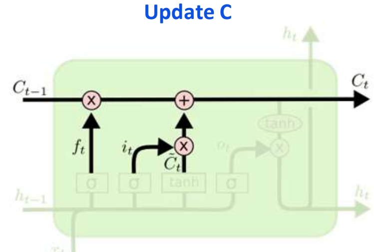
Use previous state, **C**, previous output, **h**, and current input, **x**, to determine how much to suppress previous state

Input



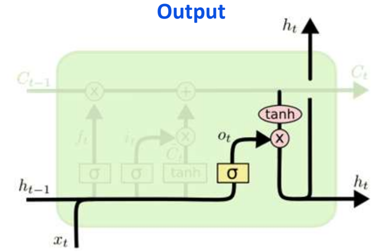
Input gate **i** determines which values of **C** to update. Separate tanh layer produces new state to add to **C**

Update C



Forget gate does attenuates **C**, and then adds output of previous input step.

Output

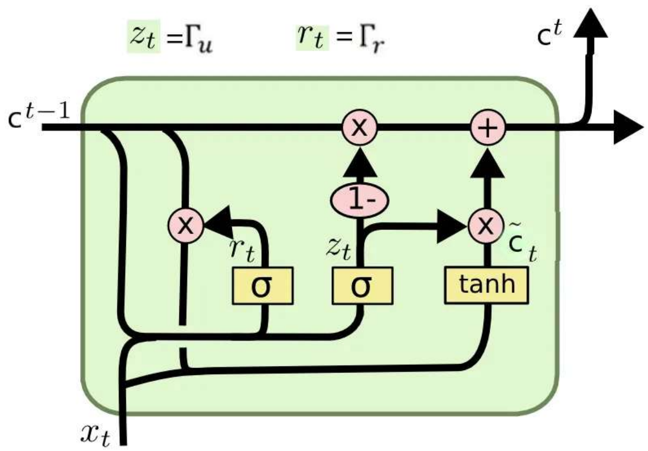


Output gates attenuates to what extent state **C** gets passed using current output **h**.

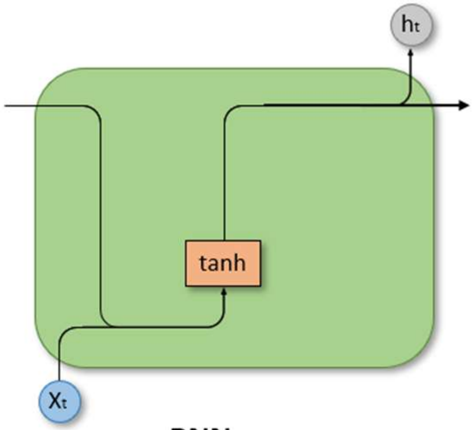
Source: http://colah.github.io/posts/2015-08-Understanding-LSTMs

Gated Recurrent Unit (GRU)

- **Simpler implementation of the LSTM**
  - The GRU model does not have an internal state variable to store memory (a value)
  - Combine **C** and **h** into a single state/output.
  - Combine forget and input gates into update gate, **z**
- **Training time is shorter**
- **Requires less data to capture data properties**

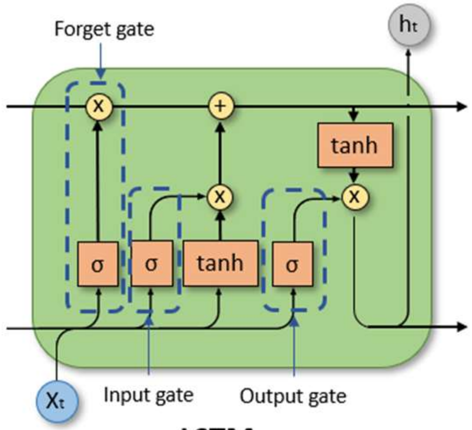


Comparison



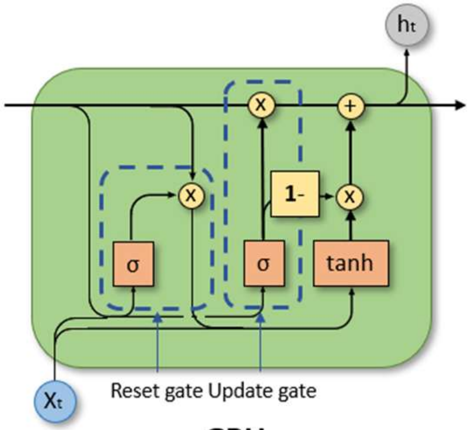
RNN

- RNNs suffer from the **vanishing gradient** problem. As sequences grow longer they struggle to remember information from earlier steps.
- Less effective for tasks that need understanding of long-term dependencies like machine translation or speech recognition.
- To resolve these challenges more advanced models such as LSTM networks were developed.



LSTM

- LSTMs a more complex than RNNs which makes them **slower to train** and demands **more memory**.
- LSTMs handle **longer sequences** better they still face challenges with very long-range dependencies.
- LSTM sequential nature also limits the ability to **process data in parallel** which slows down training.



GRU

- GRUs are **simpler** and **faster** than LSTMs but they still rely on sequential processing which limits parallelization and **slows training** on long sequences.
- Like LSTMs, they can struggle with very long-range dependencies, in some cases.

Source: geeksforgeeks.org/deep-learning/rnn-vs-lstm-vs-gru-vs-transformers


CUNY
School of
Professional Studies