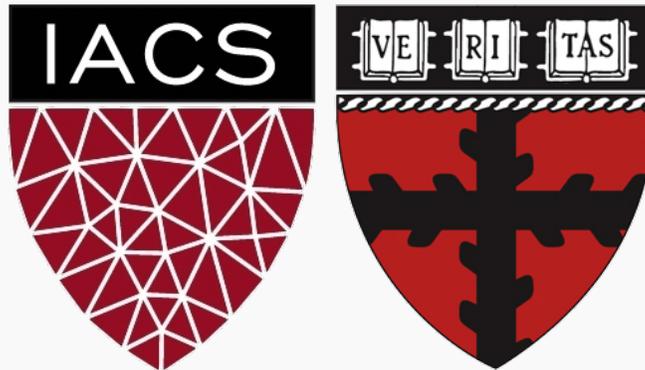


LSTM Networks

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, and Chris Tanner



Outline

Recap

GRU++

Long short-term memory LSTM

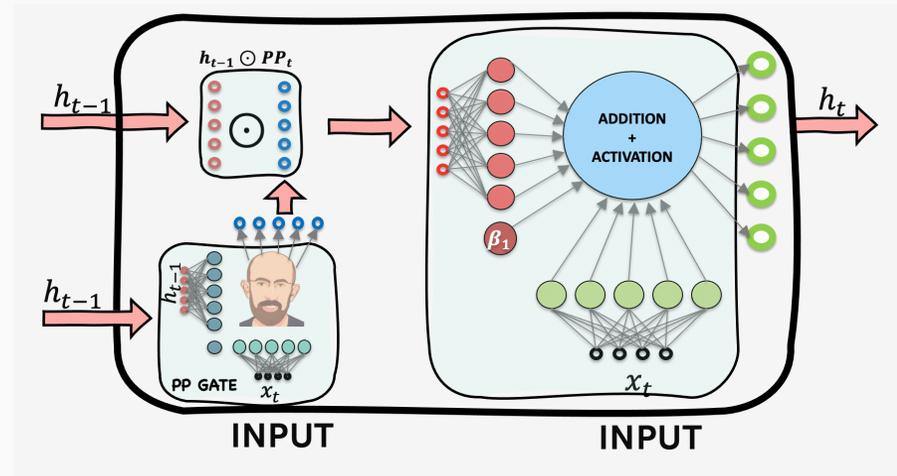
LSTM applications

$$\mathbf{h}_t = \tanh(\mathbf{V}\mathbf{X}_t + \mathbf{U}\mathbf{h}_{t-1} + \beta_1)$$

$$\mathbf{h}_t = \tanh(\mathbf{V}\mathbf{X}_t + \mathbf{U}\mathbf{h}_{t-1} + \beta_1)$$

$$\mathbf{h}_t = \tanh(\mathbf{V}\mathbf{X}_t + \mathbf{U}[\mathbf{PP}_t \odot \mathbf{h}_{t-1}] + \beta_1)$$

$$\mathbf{PP}_t = \sigma(\mathbf{V}_{pp}\mathbf{X}_t + \mathbf{U}_{pp}\mathbf{h}_{t-1} + \beta_{pp})$$

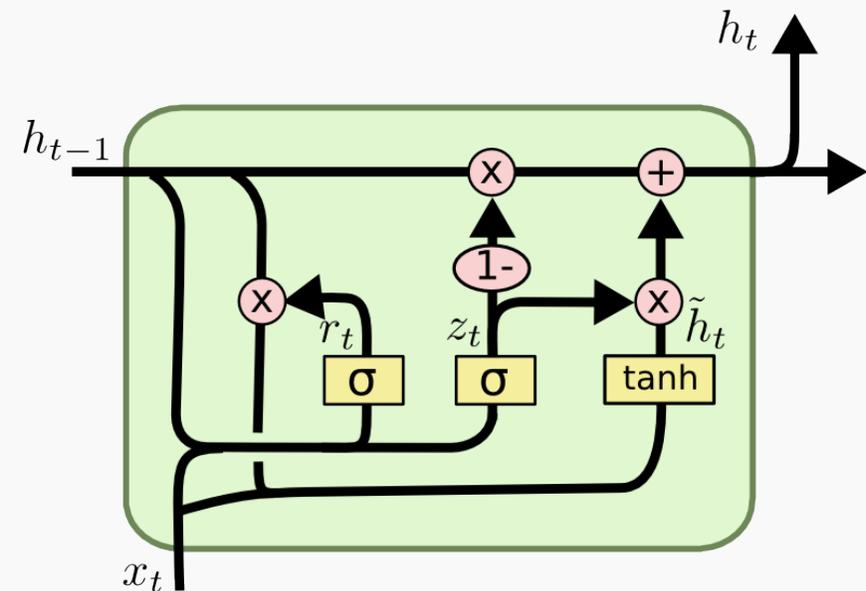


Gated Recurrent Unit (GRU)

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{V}\mathbf{X}_t + \mathbf{U}[\mathbf{R}_t \odot \mathbf{h}_{t-1}] + \beta_1)$$

$$\mathbf{h}_t = \mathbf{Z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{Z}_t) \odot \tilde{\mathbf{h}}_t$$

Reset Gate (equivalent to PP gate)



$$\mathbf{R}_t = \sigma(\mathbf{V}_R \mathbf{X}_t + \mathbf{U}_R \mathbf{h}_{t-1} + \beta_R)$$

$$\mathbf{Z}_t = \sigma(\mathbf{V}_Z \mathbf{X}_t + \mathbf{U}_Z \mathbf{h}_{t-1} + \beta_Z)$$

Update Gate



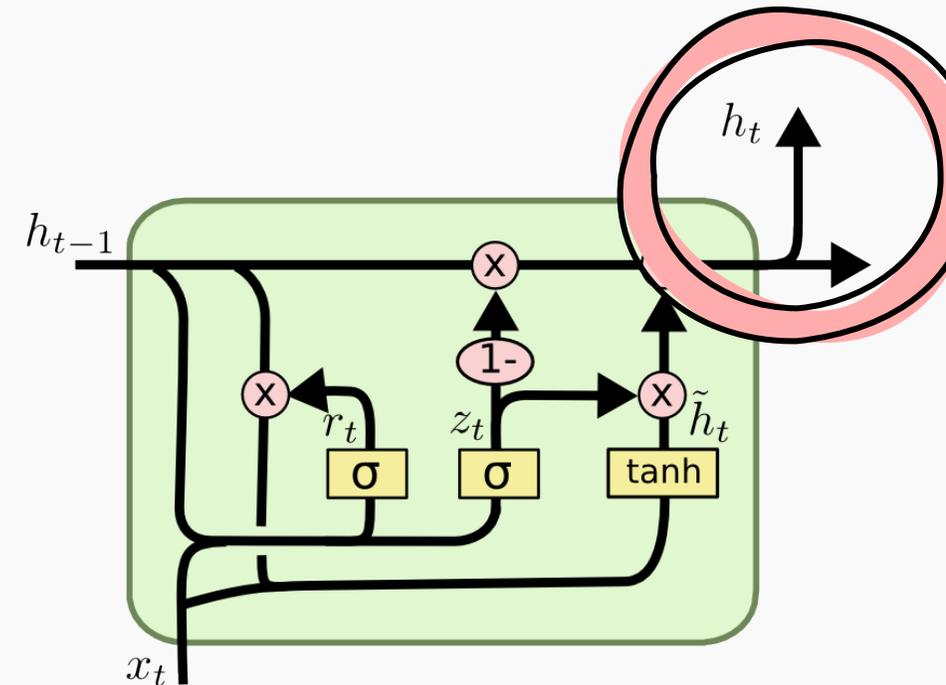
GRU

GRU STRENGTHS?

- Current input can affect how much of the past information to consider
- The update gate solves the vanishing gradient problem
- Hidden state more robust to outlier inputs because of the update gate

GRU ISSUES?

- The same hidden state is used for memory and output
- With only two gates, performance suffers on longer sequences



$$\tilde{h}_t = \tanh(X_t V + (R_t \odot h_{t-1}) U + \beta_1)$$

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

How can we
improve GRU?



$$\tilde{h}_t = \tanh(X_t V + (R_t \odot h_{t-1}) U + \beta_1)$$

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

I have an idea!



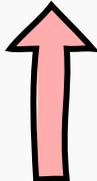
$$\tilde{h}_t = \tanh(X_t V + h_{t-1} U + \beta_1)$$

$$\tilde{h}_t = R_t \odot \tilde{h}_t$$

Instead of resetting the previous hidden state, we can reset the candidate directly



$$\tilde{h}_t = \tanh(X_t V + h_{t-1} U + \beta_1)$$

$$\tilde{h}_t = i_t \odot \tilde{h}$$


Let's just call it the *input* gate instead of *reset* gate



GRU++

$$h_t = \underbrace{f_t}_{\text{circled}} \odot h_{t-1} + \boxed{i_t \odot \tilde{h}_t}$$

We change the notation of Z_t to f_t

So now we can directly use this to find the current hidden state



$$h_t = f_t \odot h_{t-1} + i_t \odot \tilde{h}_t$$



Can become unbounded!

But isn't it possible
for h_t to become
unbounded over time?



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

Yes, it's possible.
Before we fix that, let's call the
memory part as c_t instead of h_t
(why not)



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = \tanh(c_t)$$

And use a nice activation function such as **hyperbolic tan** on it



GRU++

Note: We have two memories!
 h_t is bounded
 c_t is not bounded

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = \tanh(c_t)$$

And use a nice activation function such as **hyperbolic tan** on it



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = \tanh(c_t)$$

How about we add another gate at the end?



$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = \tanh(c_t)$$



How about we add another gate at the end?



Come on now!!!
You are killing me

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \tanh(c_t)$$

Why not!
This could make
our network more
versatile



GRU++ → LSTM

$$\tilde{c}_t = \tanh(X_t V + h_{t-1} U + \beta_1)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = o_t \odot \tanh(c_t)$$

Now, putting it
all together...

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$



Where...

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$

We now have
three gates
instead of two



LSTM

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$



But this looks like the vanilla LSTM!

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$

LSTM

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

EXACTLY, my young padawan!



But this looks like the vanilla LSTM!

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$

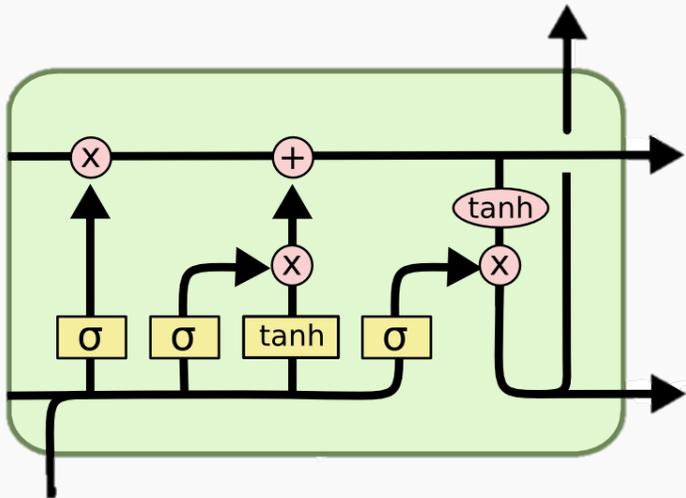


LSTM

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$



$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

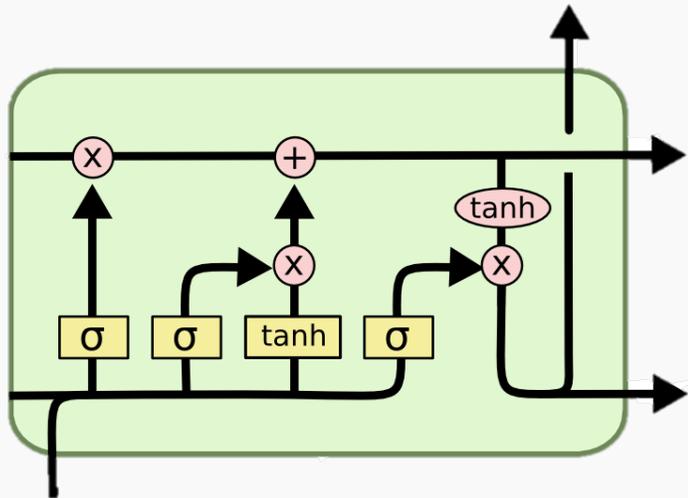
$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$

LSTM

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$



$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$

LSTM

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

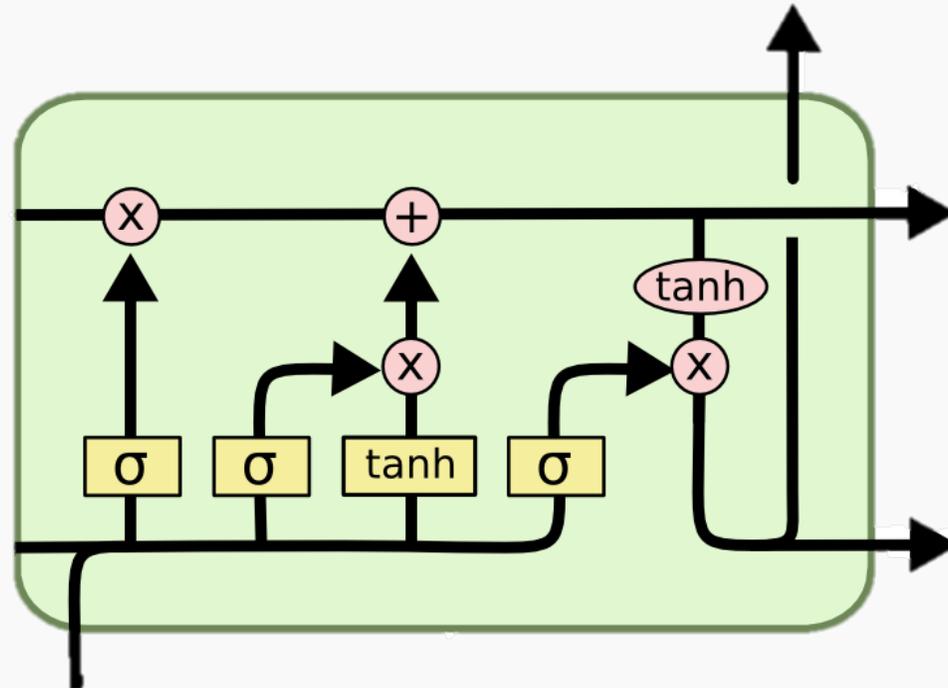
$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$



LSTM

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

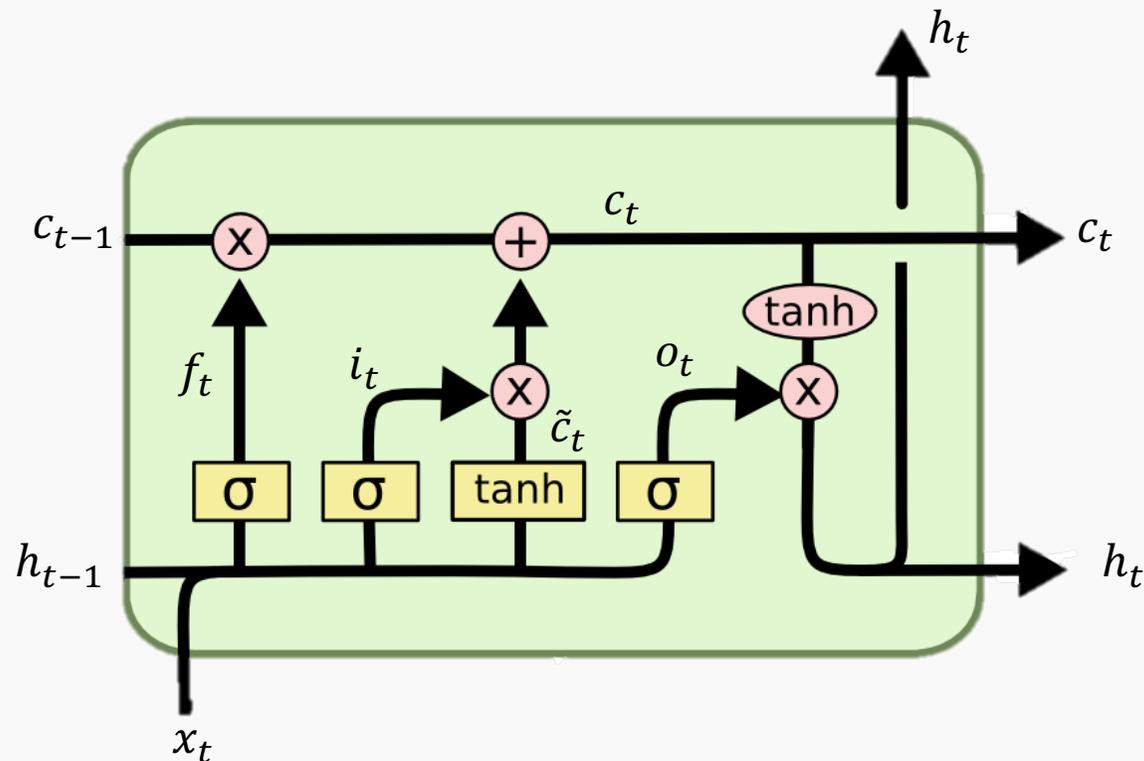
$$\mathbf{c}_t = f_t \odot \mathbf{c}_{t-1} + i_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

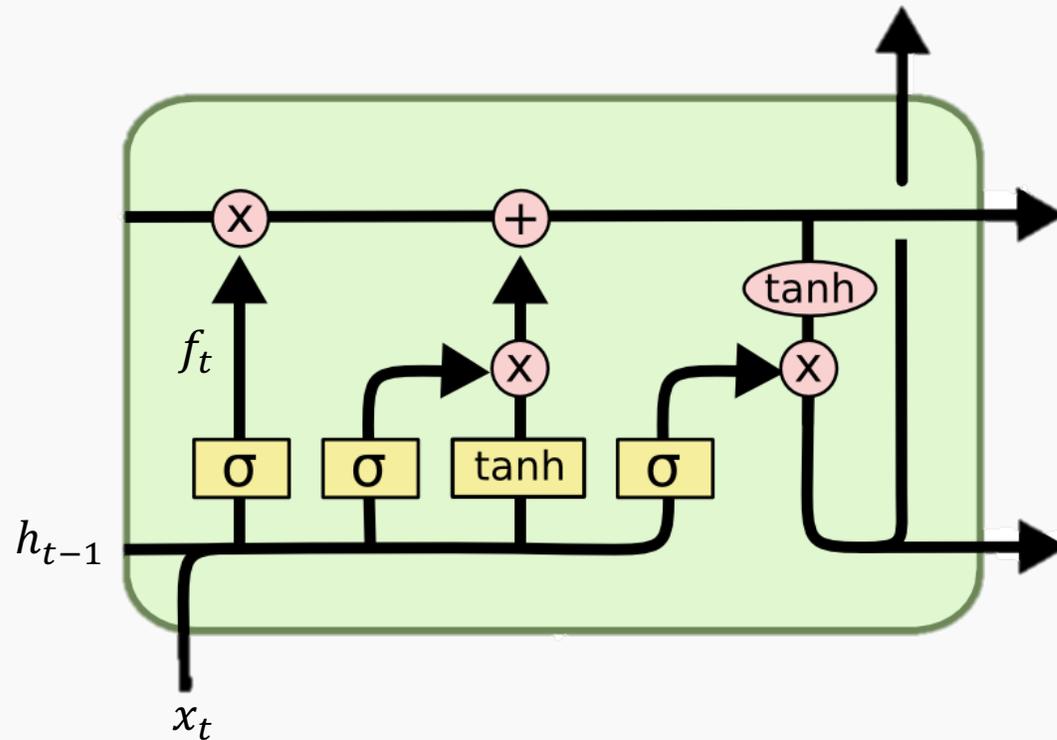
$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$



LSTM

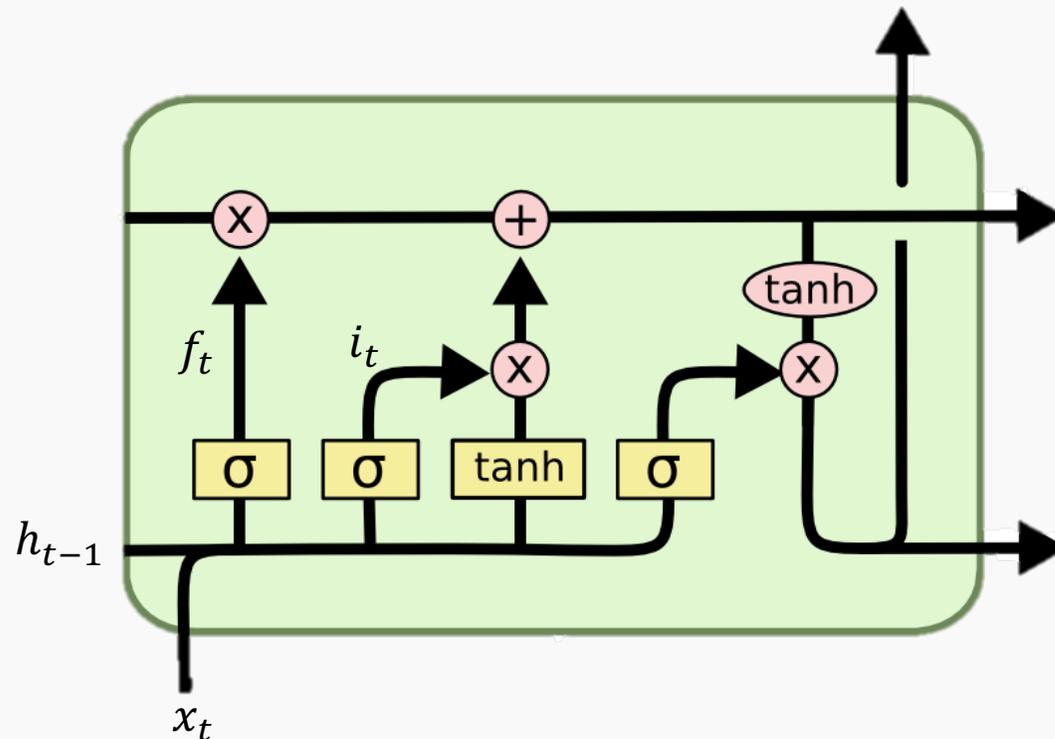
$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$



LSTM

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

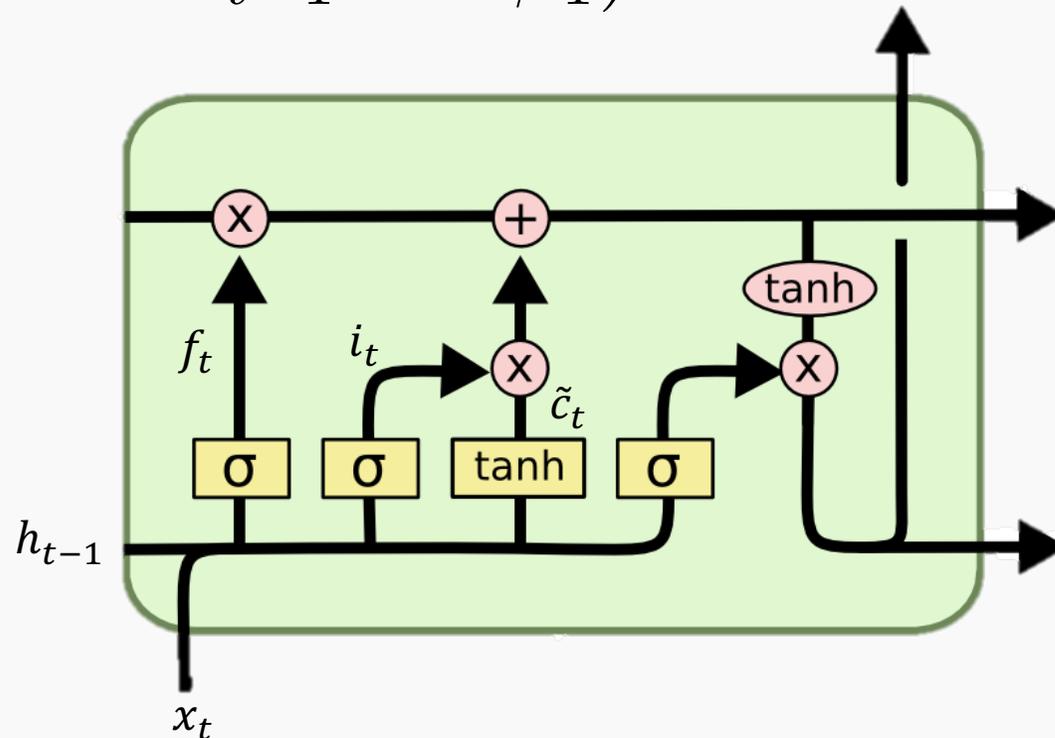


LSTM

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$



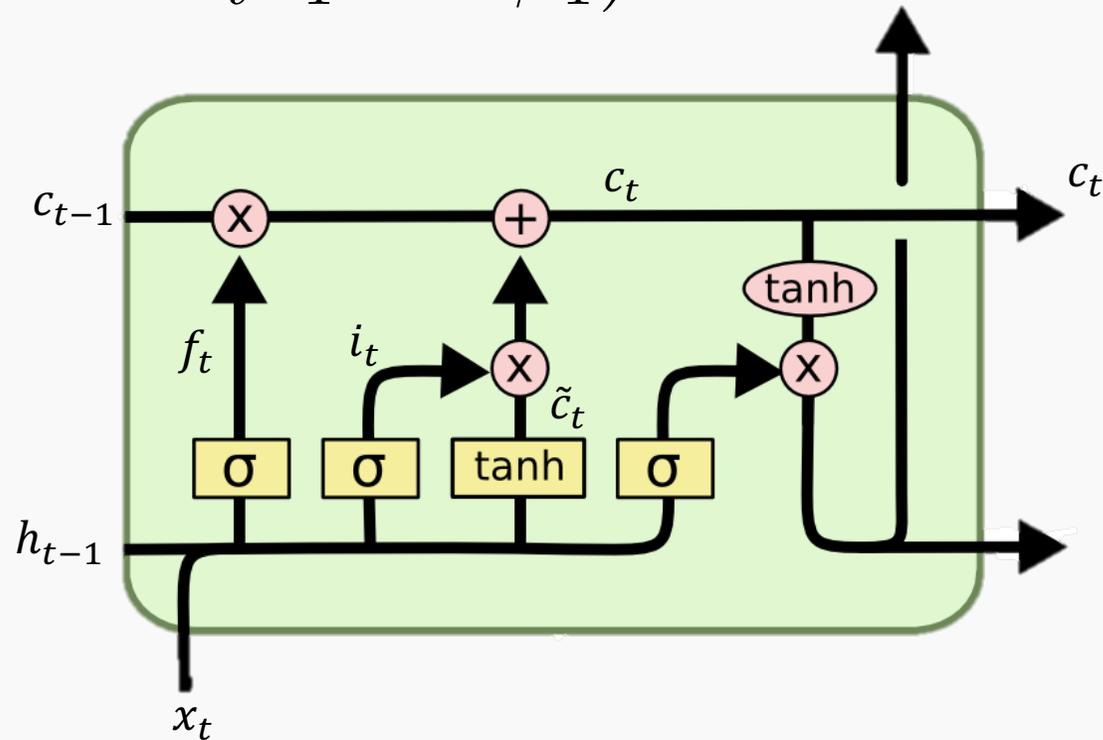
LSTM

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$



LSTM

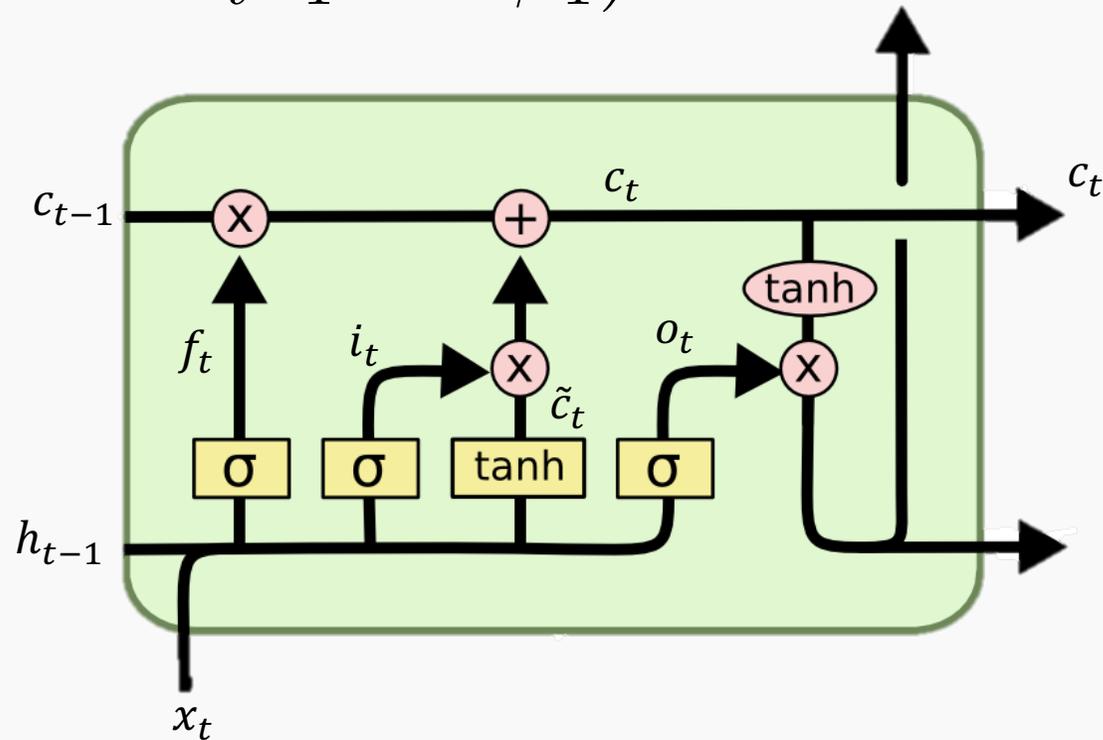
$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$



LSTM

$$\mathbf{f}_t = \sigma(\mathbf{V}_f \mathbf{X}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \beta_f)$$

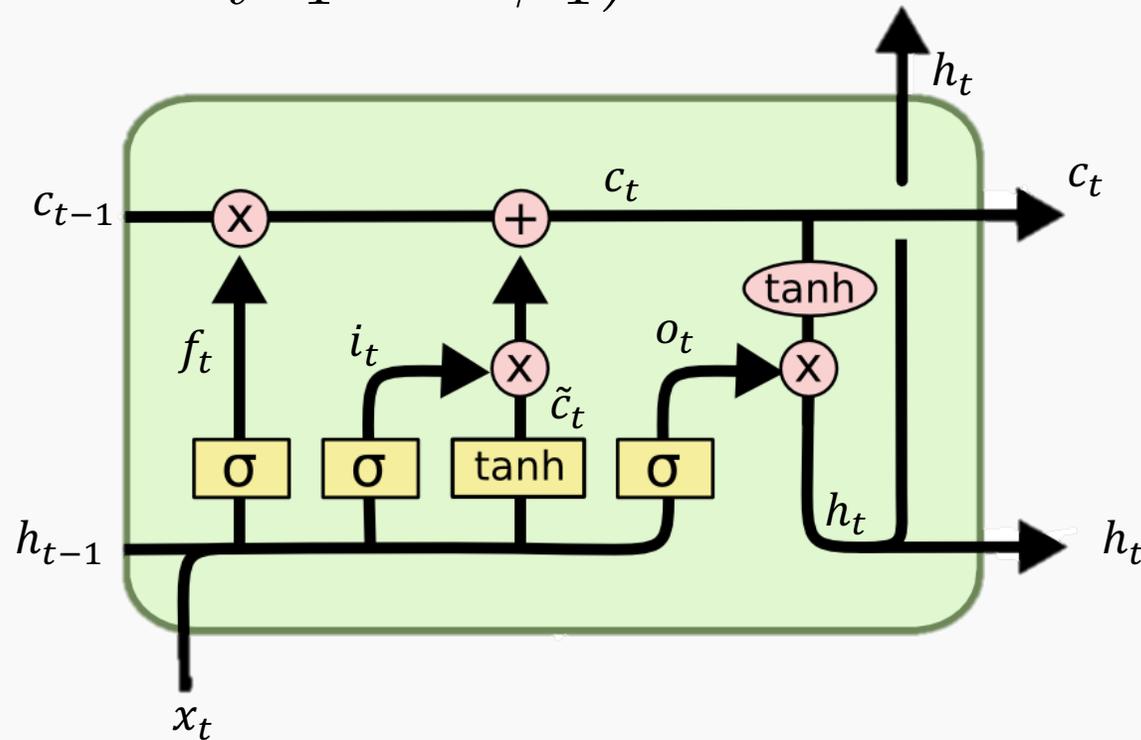
$$\mathbf{i}_t = \sigma(\mathbf{V}_i \mathbf{X}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \beta_i)$$

$$\tilde{\mathbf{c}}_t = \tanh(\mathbf{X}_t \mathbf{V} + \mathbf{h}_{t-1} \mathbf{U} + \beta_1)$$

$$\mathbf{c}_t = \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \tilde{\mathbf{c}}_t$$

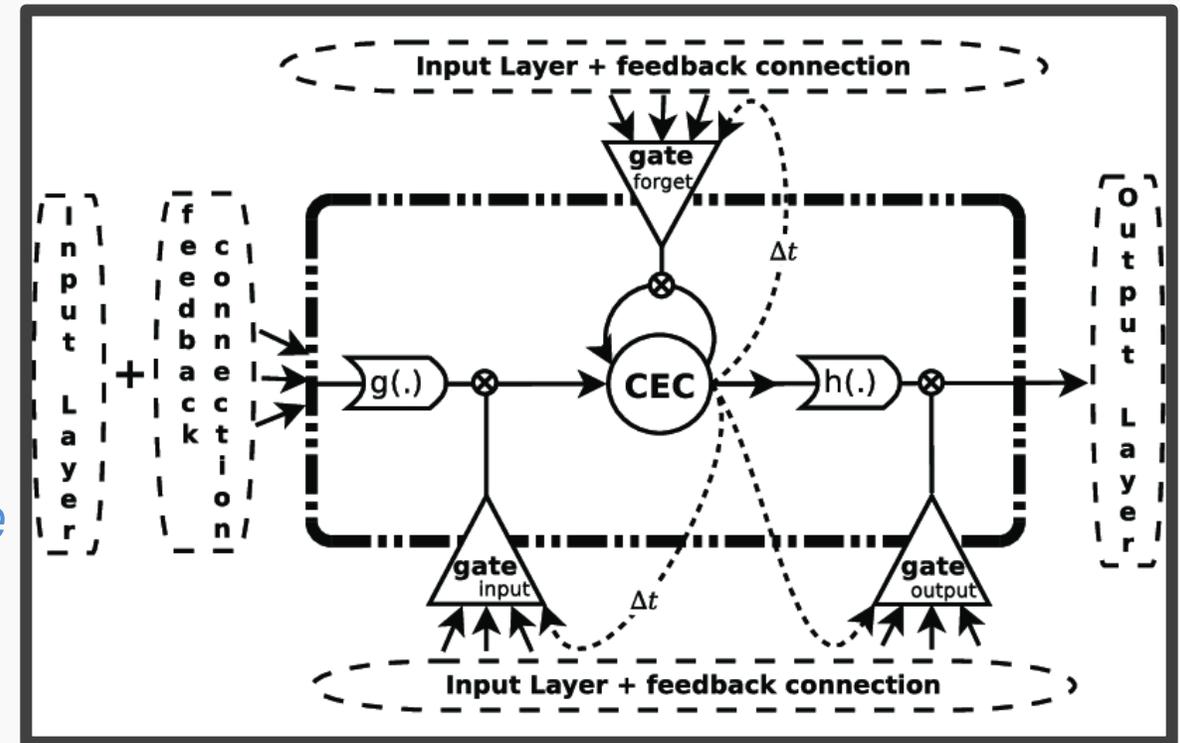
$$\mathbf{o}_t = \sigma(\mathbf{V}_o \mathbf{X}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \beta_o)$$

$$\mathbf{h}_t = \mathbf{o}_t \odot \tanh(\mathbf{c}_t)$$



Long short-term memory (LSTM)

- First introduced in **1995** to counter the **vanishing gradient problem**.
- Training was done using a mixture of Real Time Recurrent Learning and **Backpropagation Through Time**.
- Underwent several major modifications including addition of **forget gate**, **peephole connections** etc.
- In the last two decades, several other **variants** were introduced with mixed results

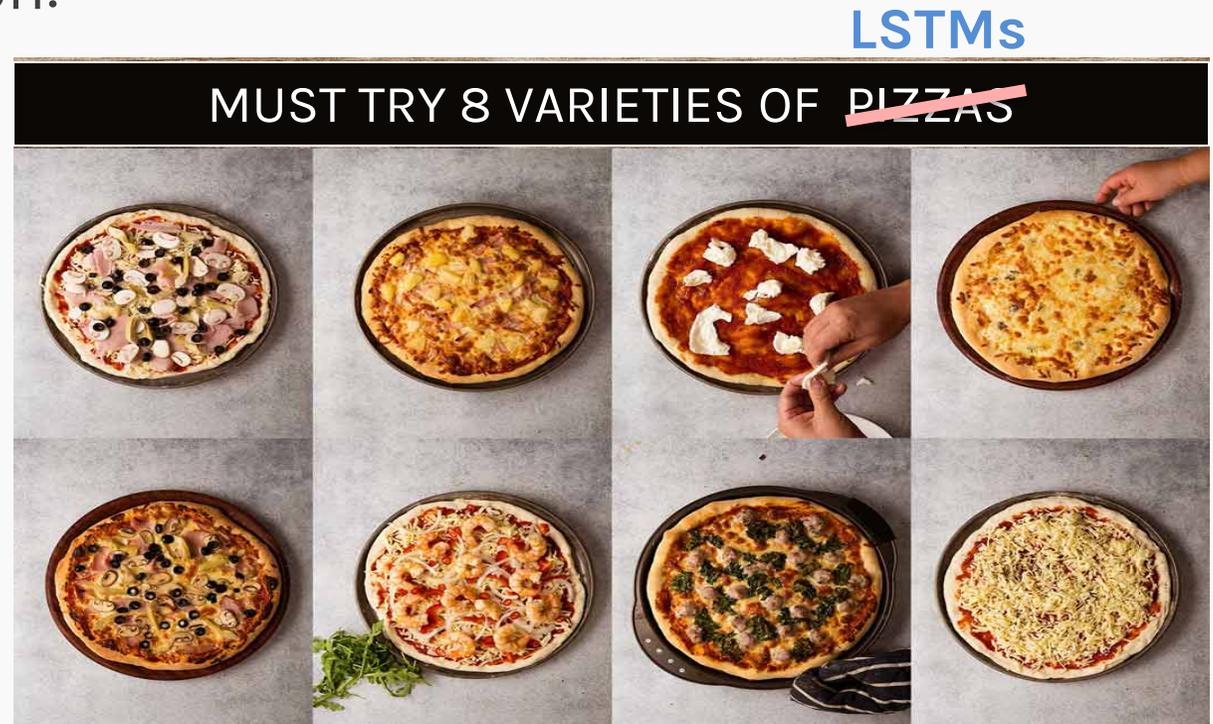


- *First prototype of LSTM cell*

Long short-term memory (LSTM)

After its introduction in 1995, here are the eight popular types of LSTM variants other than the vanilla version:

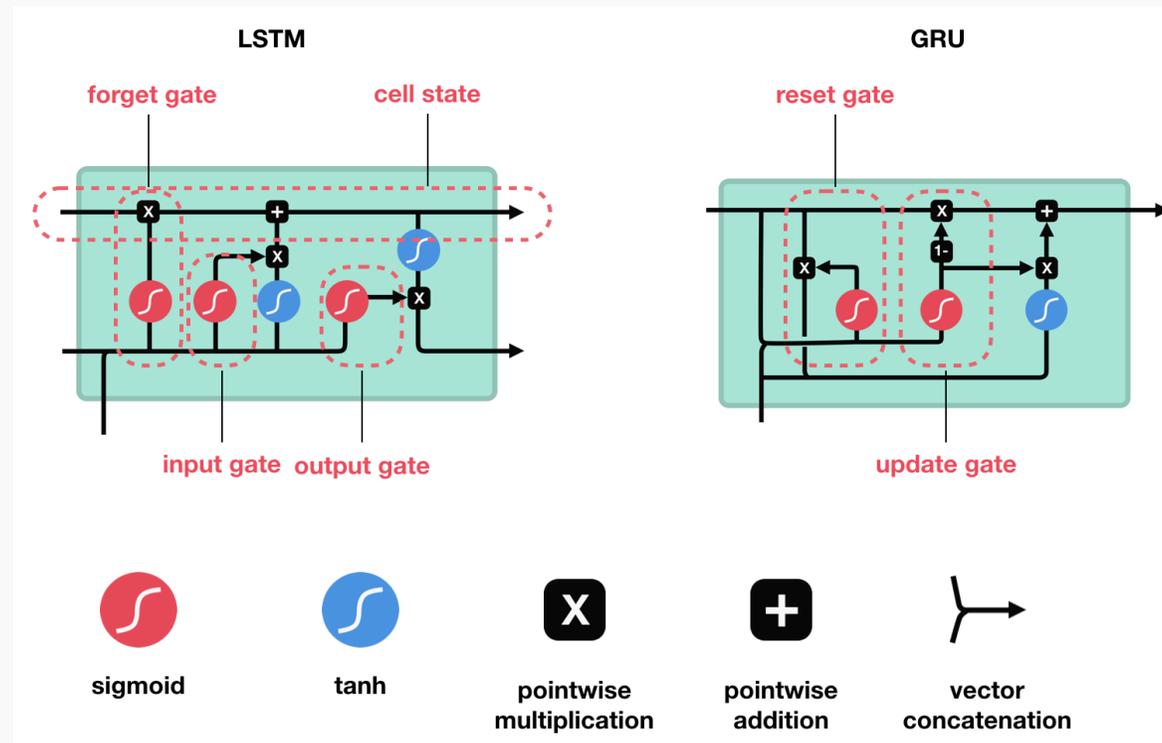
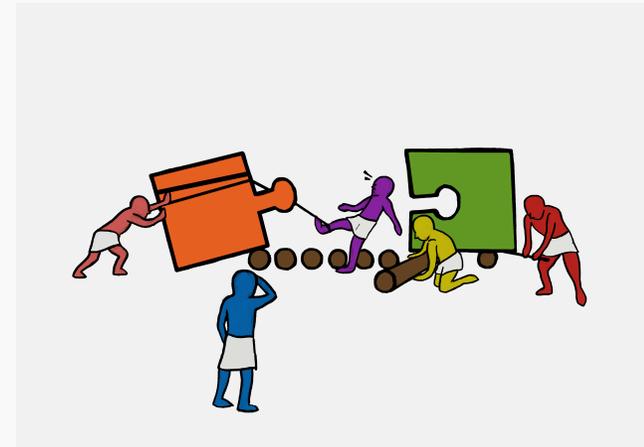
- **NIG** – No input gate
- **NFG** - No forget gate
- **NOG** – No output gate
- **NIAF** – No input activation
- **NOAF** – No output activation
- **CIFG** – Coupled input/forget gate
- **NP** – No peepholes
- **FGR** – Full gate recurrence



Please refer to the paper [*LSTM: A Search Space Odyssey*](#) for a thorough analysis of all the variants

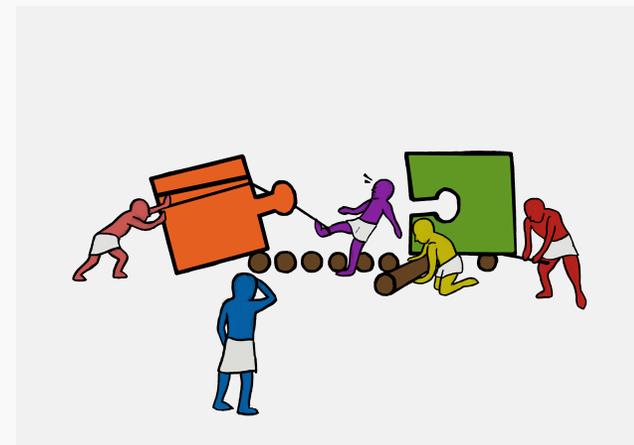
Exercise: LSTM v/s GRU

The goal of this exercise is to compare the performance between two popular gating methods, i.e LSTM and GRUs:



Exercise: LSTM v/s GRU

As in previous exercises you need to add an embedding layer.



EMBEDDING LAYER

