
LSTMs
Pavlos Protopapas

Outline

- Recap: GRU
- GRU++
- Long short-term memory (LSTM)
- LSTM applications

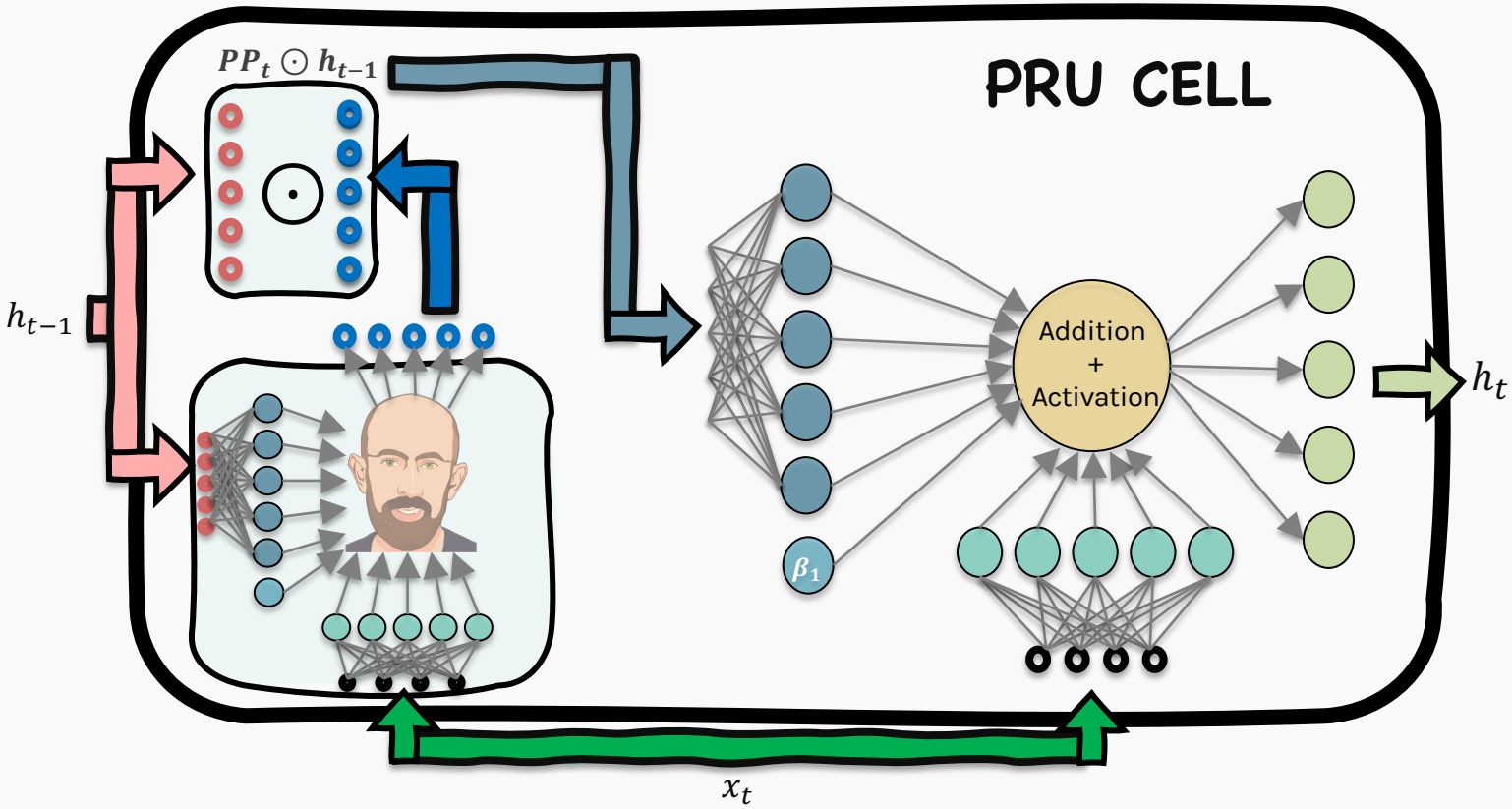
Outline

- **Recap: GRU**
- GRU++
- Long short-term memory (LSTM)
- LSTM applications

Recap: Pavlos Recurrent Unit (PRU)

$$h_t = \tanh(VX_t + U[PP_t \odot h_{t-1}] + \beta_1)$$

$$PP_t = \sigma(V_{pp}X_t + U_{pp}\ h_{t-1} + \beta_{pp})$$



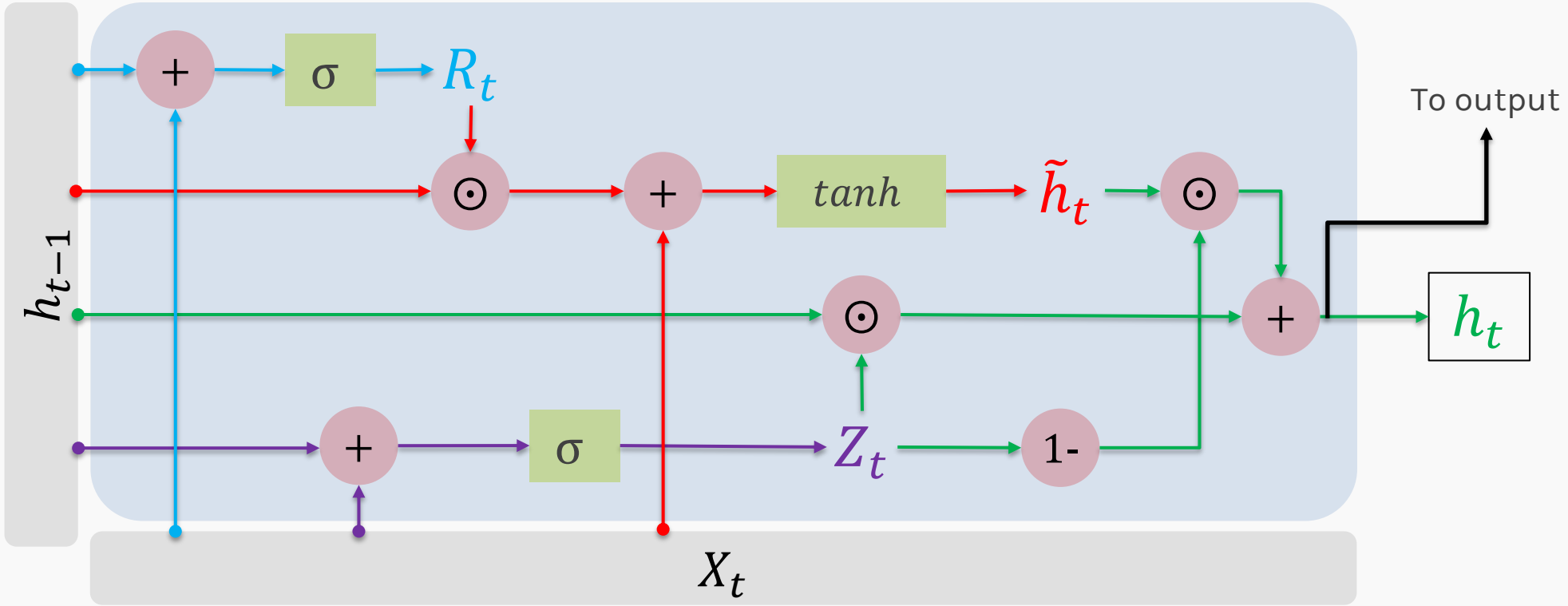
Recap: Gated Recurrent Unit (GRU)

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

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

$$R_t = \sigma(V_R X_t + U_R h_{t-1} + \beta_R)$$

$$Z_t = \sigma(V_Z X_t + U_Z h_{t-1} + \beta_Z)$$



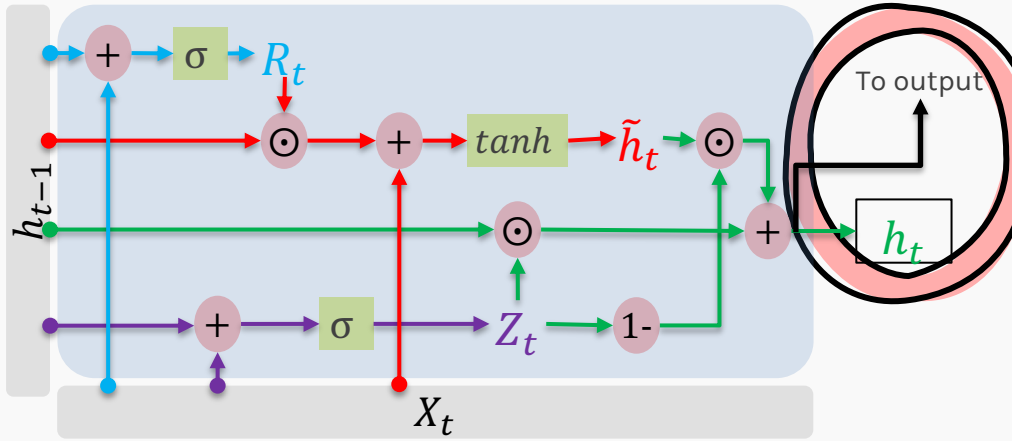
Recap: Gated Recurrent Unit (GRU)

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



Outline

- Recap: GRU
- **GRU++**
- Long short-term memory (LSTM)
- LSTM applications

GRU++

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

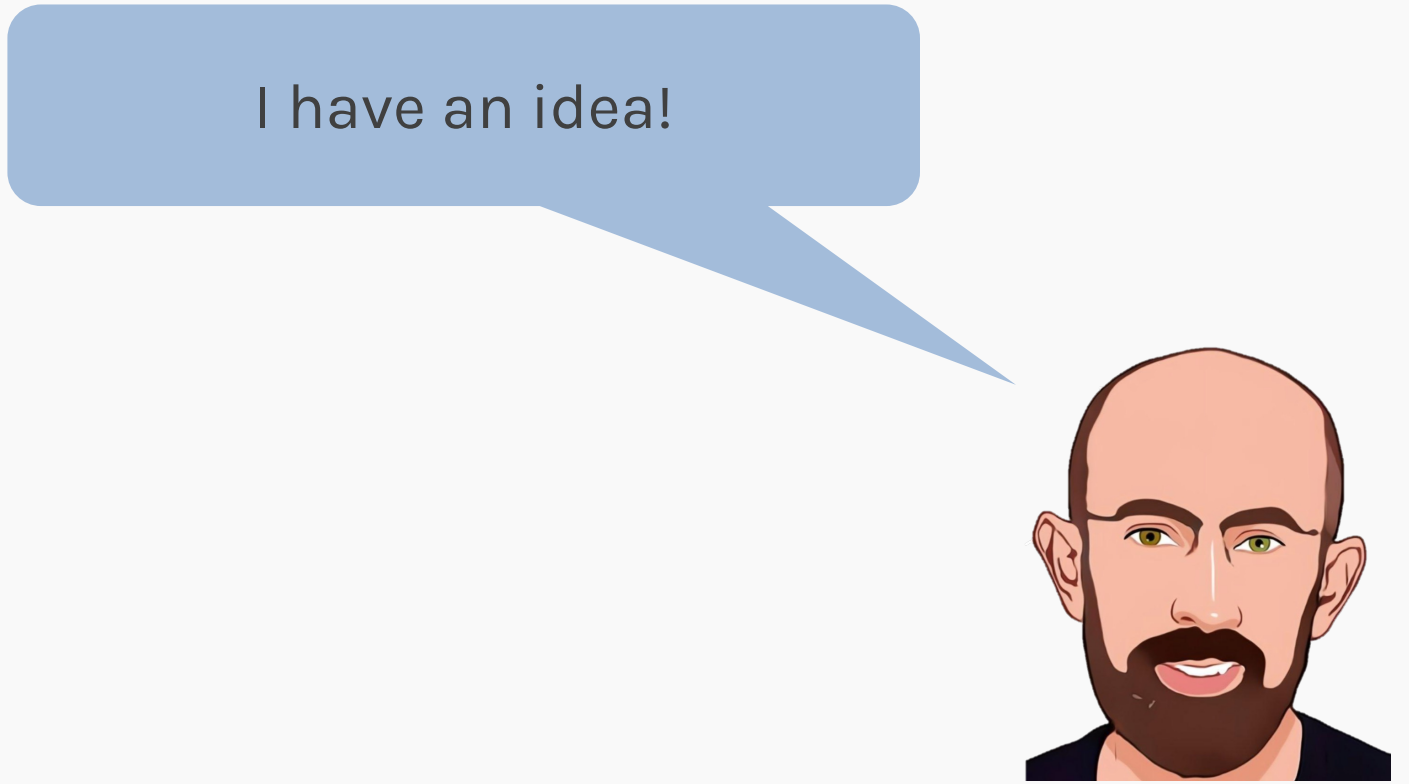
$$h_t = Z_t \odot h_{t-1} + (1 - Z_t) \odot \tilde{h}_t$$

How can we improve GRU?

GRU++

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

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



GRU++

$$\tilde{h}_t = \tanh(VX_t + Uh_{t-1} + \beta_1)$$

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

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

GRU++

$$\tilde{h}_t = \tanh(VX_t + Uh_{t-1} + \beta_1)$$

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

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

GRU++

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

We change the notation of $\mathrm{Z_t}$ to $\mathrm{f_t}$

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

GRU++

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

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

GRU++

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

Yes, it's possible.

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

GRU++

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

Yes, it's possible.

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

GRU++

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

$$h_t = \tanh(c_t)$$

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

GRU++

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

$$h_t = \tanh(c_t)$$

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

GRU++

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

$$h_t = \tanh(c_t)$$

How about we add another gate at the end?

GRU++

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

$$h_t = \tanh(c_t)$$

How about we add another gate at the end?

Come on now!!!
You are killing me

GRU++

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

$$h_t = \mathrm{o_t} \odot \tanh(c_t)$$

Why not!
This could make our network more versatile.

GRU++ → LSTM

$$\tilde{c}_t = \tanh(VX_t + Uh_{t-1} + \beta_1)$$

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

$$h_t = \mathrm{o_t} \odot \tanh(c_t)$$

$$\mathrm{f_t} = \sigma(V_f X_t + U_f h_{t-1} + \beta_f)$$

$$\mathrm{i_t} = \sigma(V_i X_t + U_i h_{t-1} + \beta_i)$$

$$\mathrm{o_t} = \sigma(V_o X_t + U_o h_{t-1} + \beta_o)$$

Now, putting it all together...

GRU++ → LSTM

$$\tilde{c}_t = \tanh(VX_t + Uh_{t-1} + \beta_1)$$

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

$$h_t = \mathrm{o_t} \odot \tanh(c_t)$$

$$\mathrm{f_t} = \sigma(V_f X_t + U_f h_{t-1} + \beta_f)$$

$$\mathrm{i_t} = \sigma(V_i X_t + U_i h_{t-1} + \beta_i)$$

$$\mathrm{o_t} = \sigma(V_o X_t + U_o h_{t-1} + \beta_o)$$

We have **three** gates now, instead of two

GRU++ → LSTM

$$\tilde{c}_t = \tanh(VX_t + Uh_{t-1} + \beta_1)$$

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

$$h_t = \mathrm{o_t} \odot \tanh(c_t)$$

$$\mathrm{f_t} = \sigma(V_f X_t + U_f h_{t-1} + \beta_f)$$

$$\mathrm{i_t} = \sigma(V_i X_t + U_i h_{t-1} + \beta_i)$$

$$\mathrm{o_t} = \sigma(V_o X_t + U_o h_{t-1} + \beta_o)$$

GRU++ → LSTM

$$\tilde{c}_t = \tanh(VX_t + Uh_{t-1} + \beta_1)$$

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

$$h_t = \mathrm{o_t} \odot \tanh(c_t)$$

$$\mathrm{f_t} = \sigma(V_f X_t + U_f h_{t-1} + \beta_f)$$

$$\mathrm{i_t} = \sigma(V_i X_t + U_i h_{t-1} + \beta_i)$$

$$\mathrm{o_t} = \sigma(V_o X_t + U_o h_{t-1} + \beta_o)$$

Outline

- Recap: GRU
- GRU++
- **Long short-term memory (LSTM)**
- LSTM applications

LSTM: Trainable Weights

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

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

$$h_t = \mathrm{o_t} \odot \tanh(c_t)$$

$$\mathrm{f_t} = \sigma(V_f X_t + U_f h_{t-1} + \beta_f)$$

$$\mathrm{i_t} = \sigma(V_i X_t + U_i h_{t-1} + \beta_i)$$

$$\mathrm{o_t} = \sigma(V_o X_t + U_o h_{t-1} + \beta_o)$$

GRU++ → LSTM

$$\tilde{c}_t = \tanh(VX_t + Uh_{t-1} + \beta_1)$$

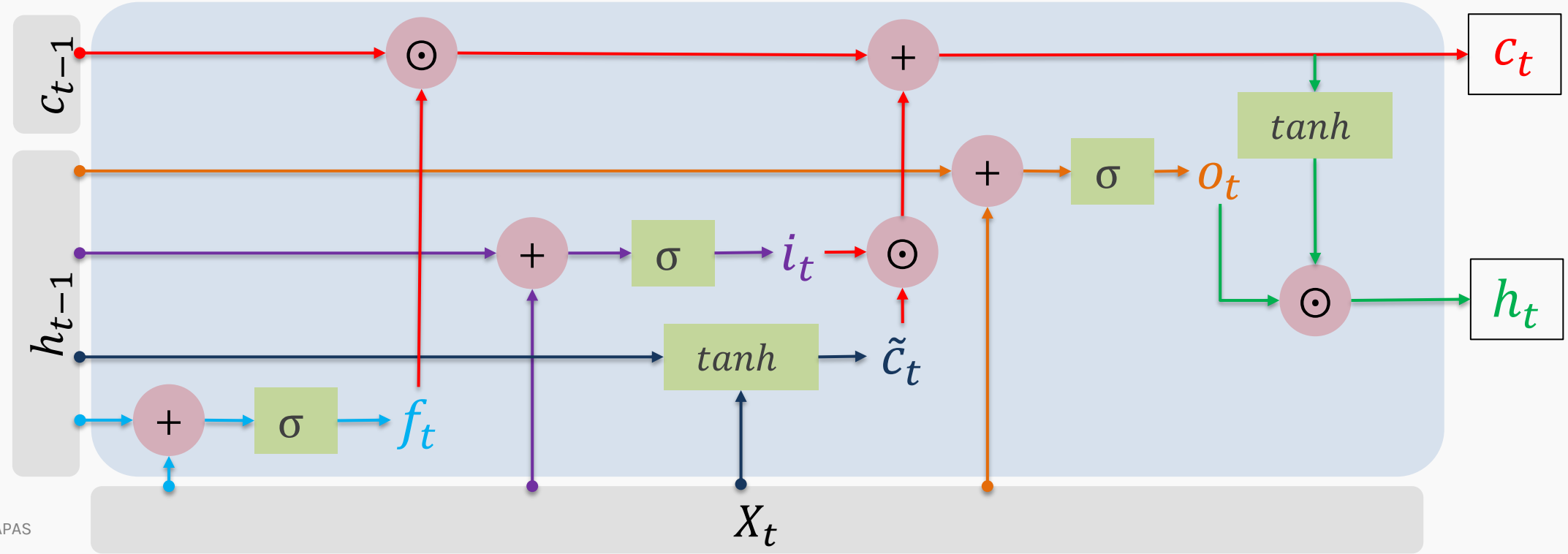
$$c_t = \mathrm{f_t} \odot c_{t-1} + i_t \odot \tilde{c}_t$$

$$h_t = \mathrm{o_t} \odot \tanh(c_t)$$

$$\mathrm{f_t} = \sigma(V_f X_t + U_f h_{t-1} + \beta_f)$$

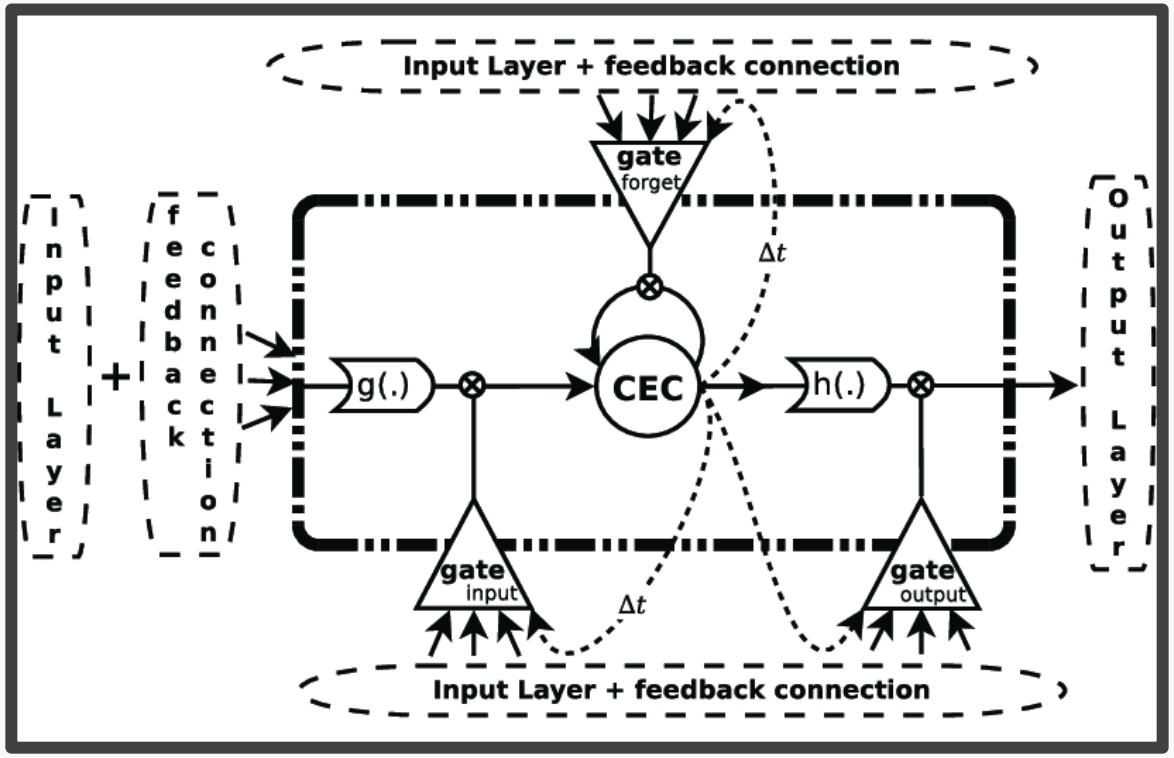
$$\mathrm{i_t} = \sigma(V_i X_t + U_i h_{t-1} + \beta_i)$$

$$\mathrm{o_t} = \sigma(V_o X_t + U_o h_{t-1} + \beta_o)$$



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.

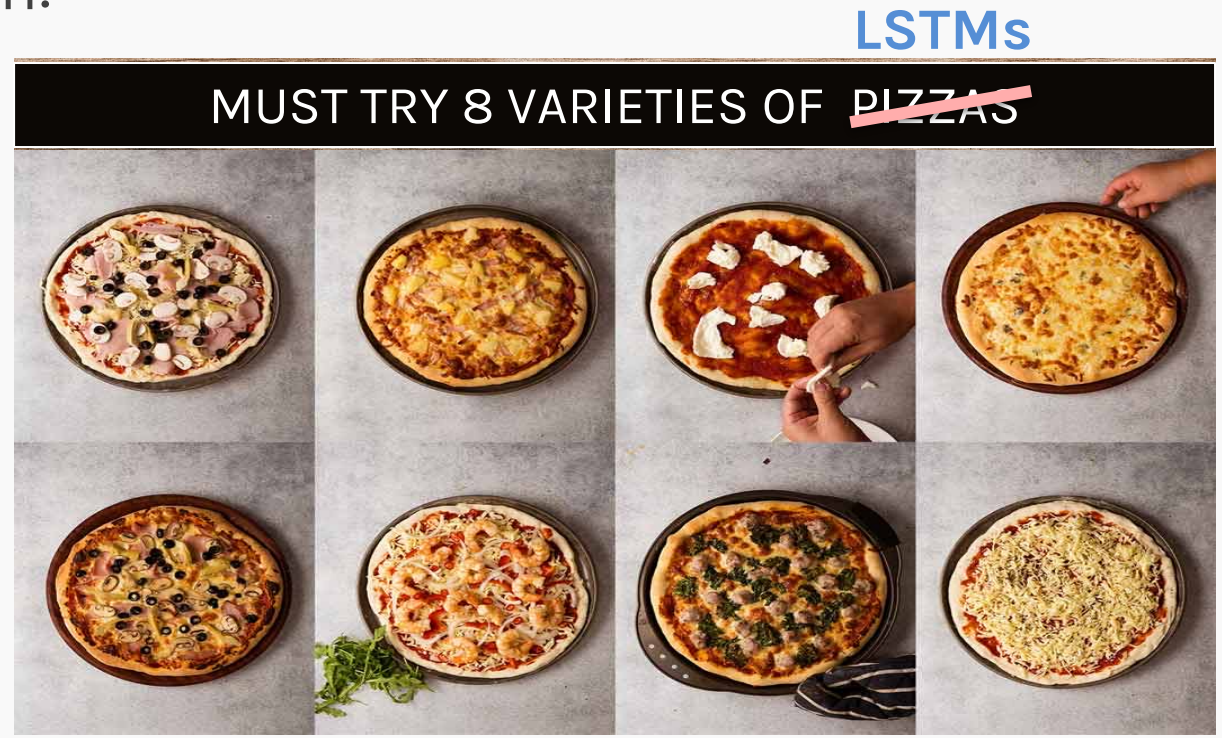


- *First prototype of LSTM cell*

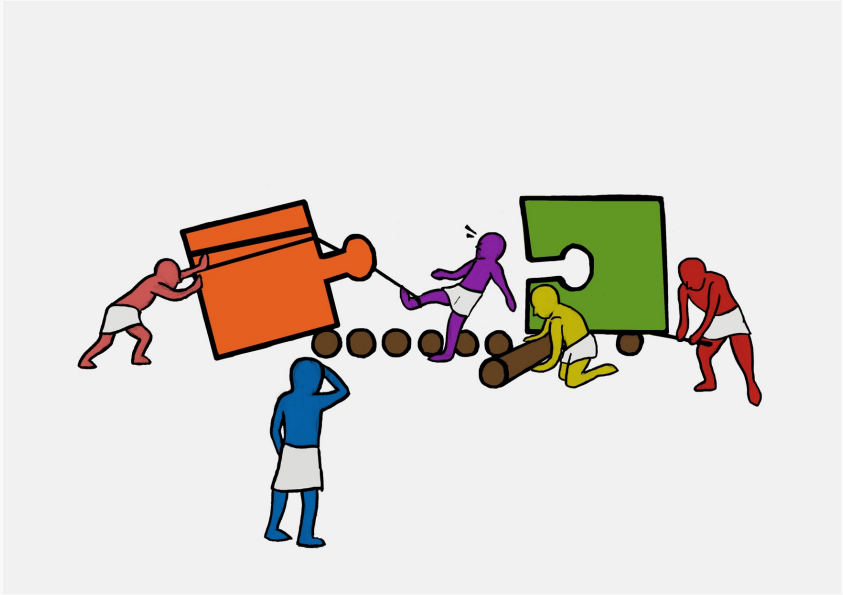
Long short-term memory (LSTM)

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**

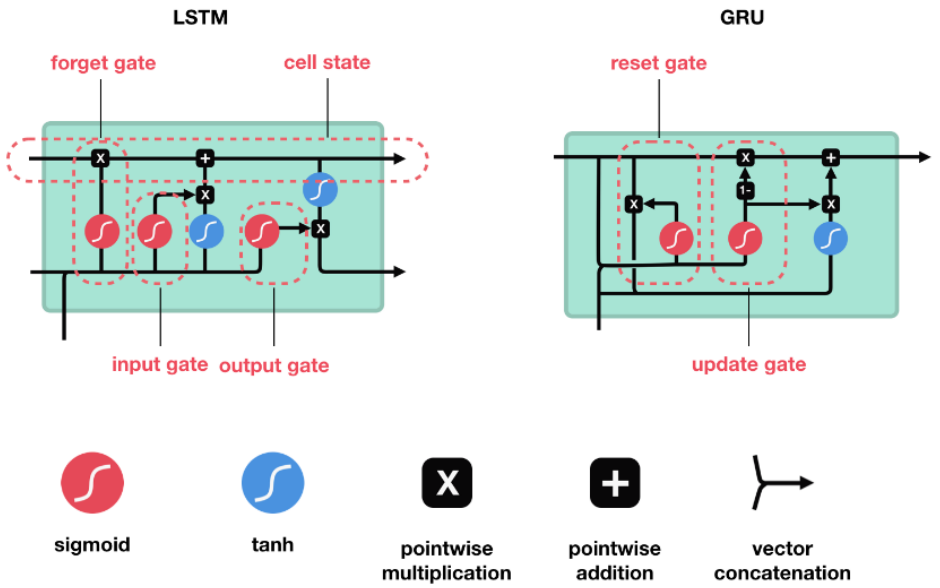


Please refer to the paper ***LSTM: A Search Space Odyssey*** for a thorough analysis of all the variants



🏋 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:



Instructions

- Read the IMDB dataset from the helper code given.
- Take a quick look at your training inputs and labels.
- Pad the values to a fix number `max_words` in-order to have sequences of the same size.
- Build, compile and fit a GRU model