HDDL - LESSON 4

RECURRENT NEURAL NETWORKS
LSTMs AND GRUs

JOSEBA DALMAU

Sequential Data

The boss said the employee is late.

The employee said the boss is late.

Same words, different meanings

⇒ The order of the words in the sentence matters!

Sequential Data

- Text
- Audio : speech, music
- Video
- Time series
- Biological data : DNA , RNA

Tasks for Sequential Data

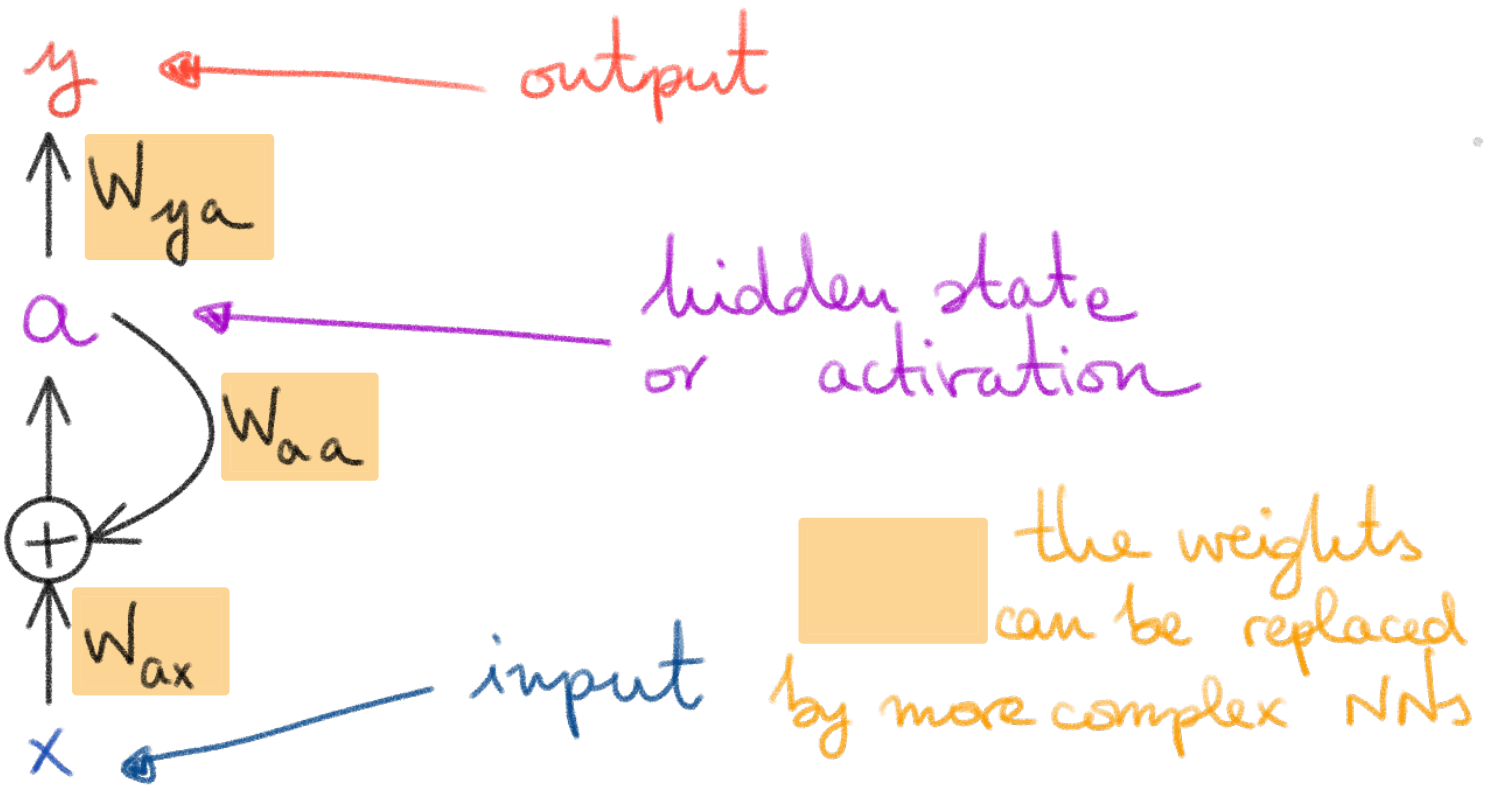
- Prediction → next token prediction in text, audio
  → forecasting for time series
- Classification → sentiment analysis
  → DNA sequence classification
- Generation : text, audio, music, video...
- Sequence-to-sequence mapping (seq2seq)
  → machine translation
  → text summarization
  → speech-to-text conversion

Naive Solutions

- Use common feed-forward networks
  - → separate parameters for each input
  - → have to learn language rules for all positions
  - → fixed size input/output
- Use a 1-dimensional convolution
  - → parameters are shared accross inputs
  - → output is a fcn of a small nb of neighbors of the input
  - → fixed size input/output

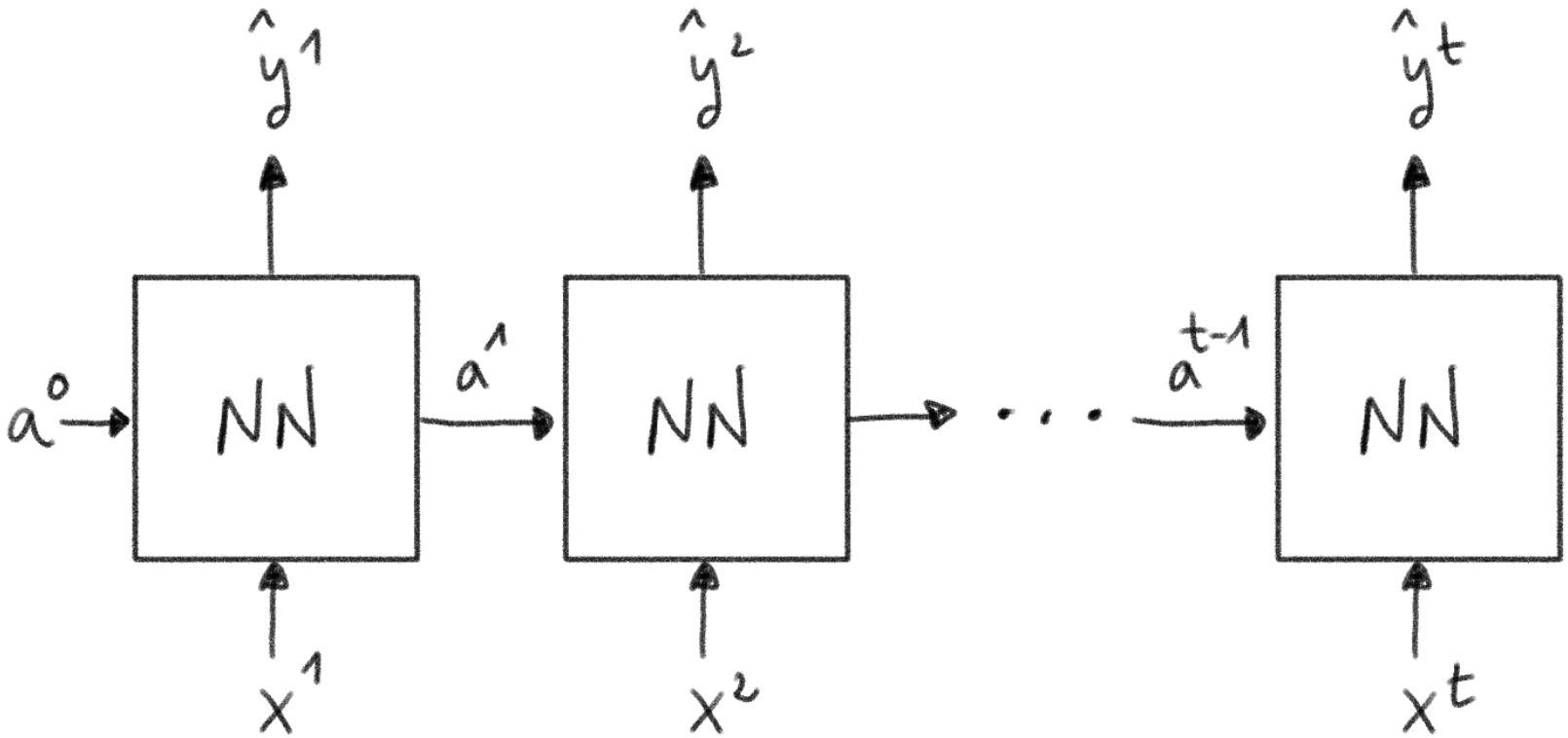
RNNs: Basic Idea

Process input sequentially, keep activations for next input:



RNNs: "Unfolding"

Process input sequentially, Keep activations for next input:



Simplest Formulas

$$a^t = g_1\left(W_{aa} a^{t-1} + W_{ax} x^t + b_a \right)$$

activation functions — weight matrices — bias vectors

$$\hat{y}^t = g_2\left(W_{ya} a^t + b_y \right)$$

The coefficients in the RED COLORED matrices and vectors are the same for all t!

PROs AND CONs

Can process input of any length

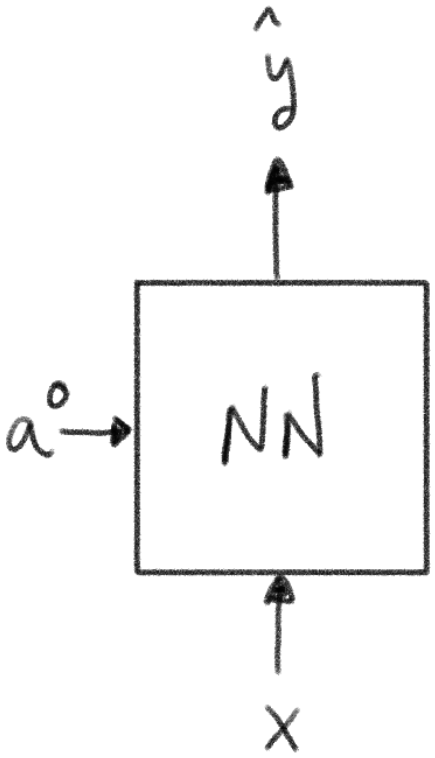
Model size indep. of input length

Slow computation (why?)

Bad with long range dependencies

Cannot consider future input

RNN Types: one-to-one

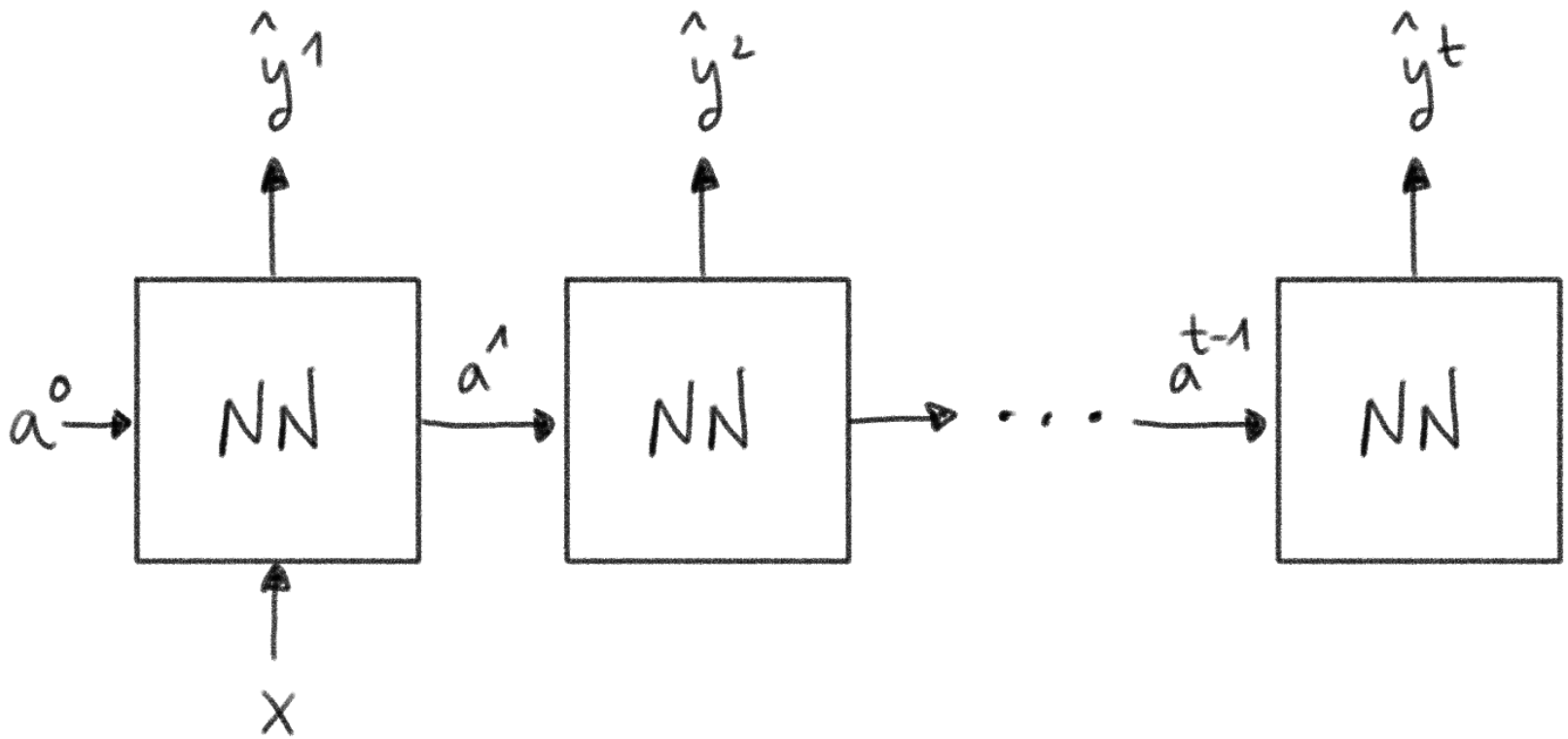


$T_x = T_y = 1$

↑ sequence lengths

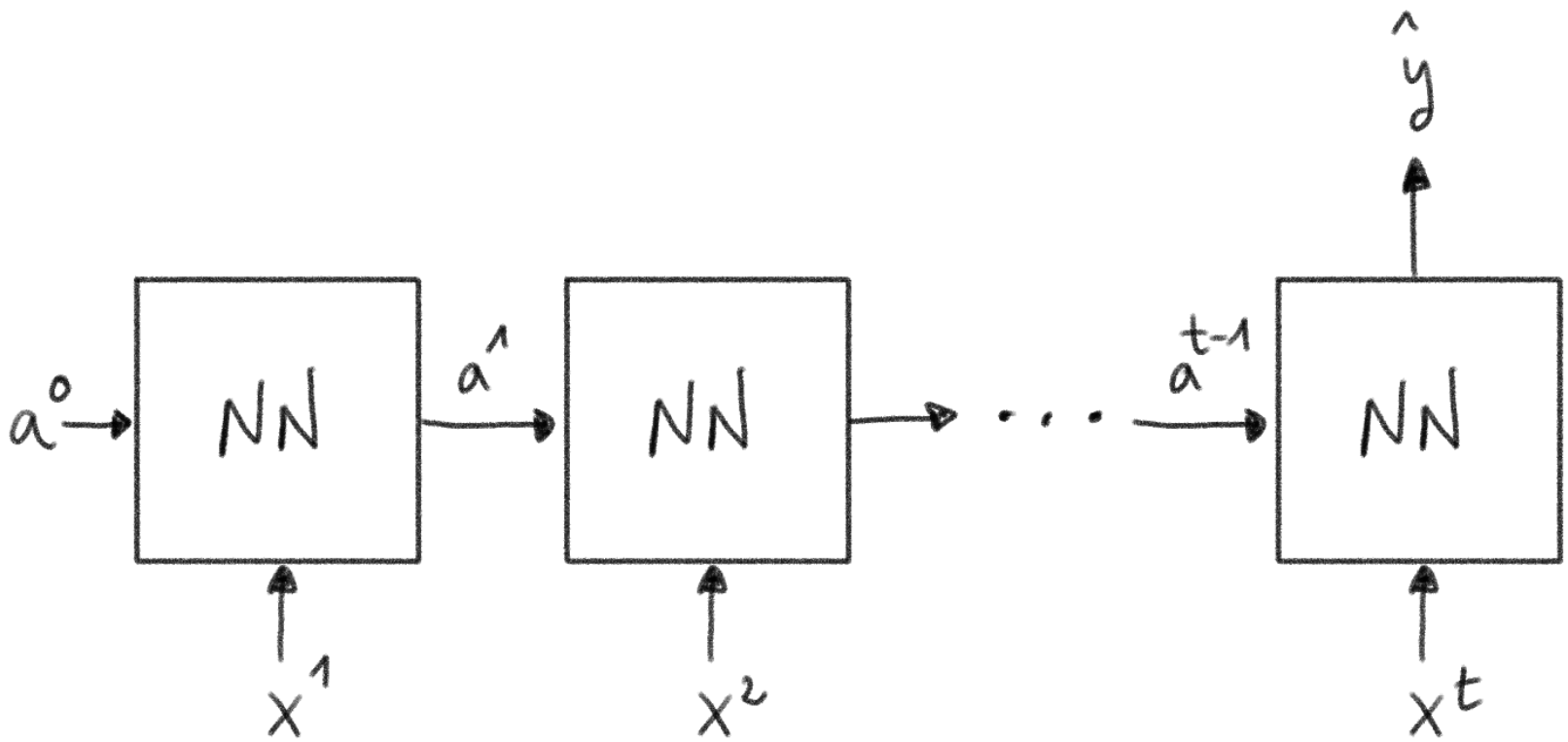
e.g. Fully Connected or Convolutional NNs

RNN Types: one-to-many



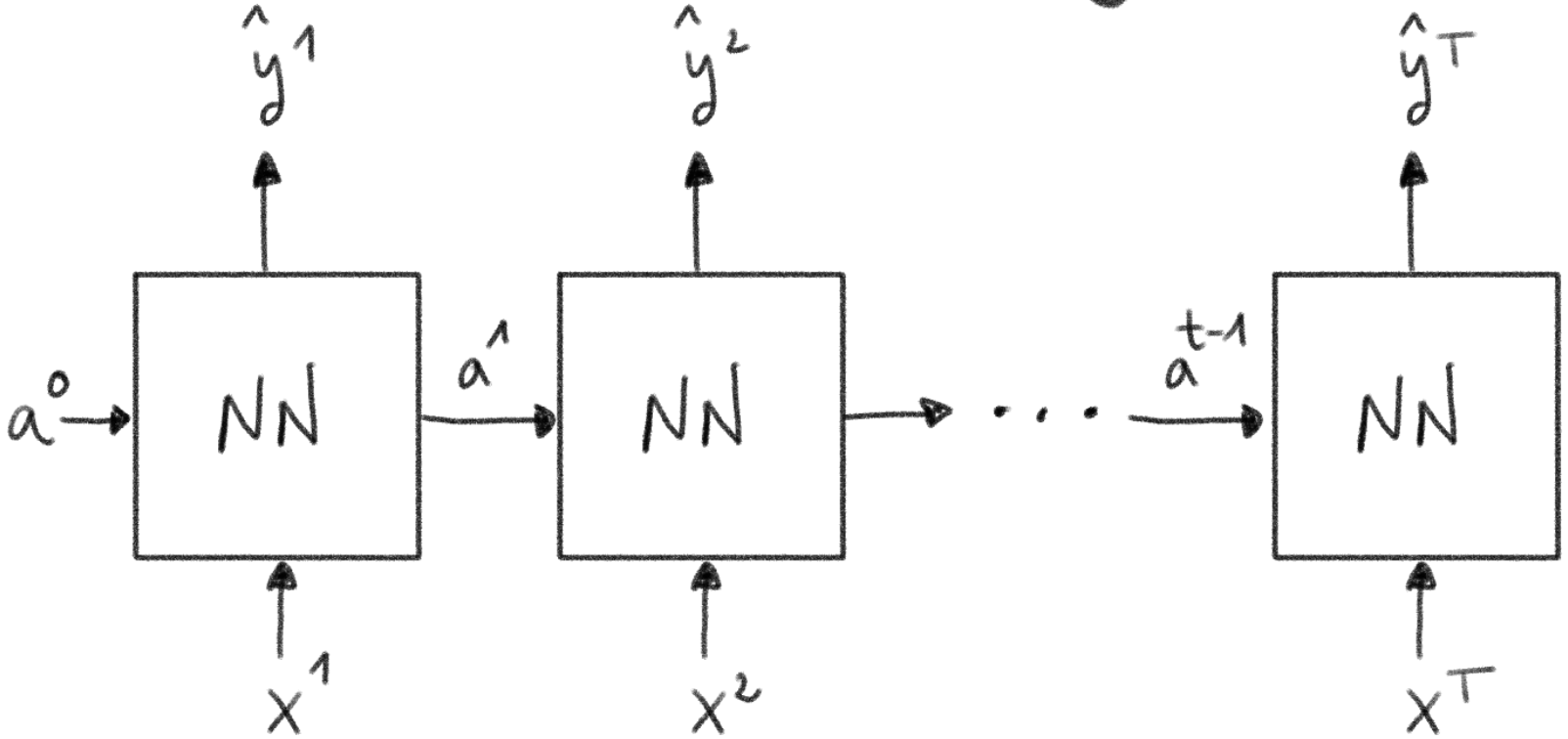
$T_x = 1, T_y > 1$ e.g. image captioning

RNN Types: many-to-one



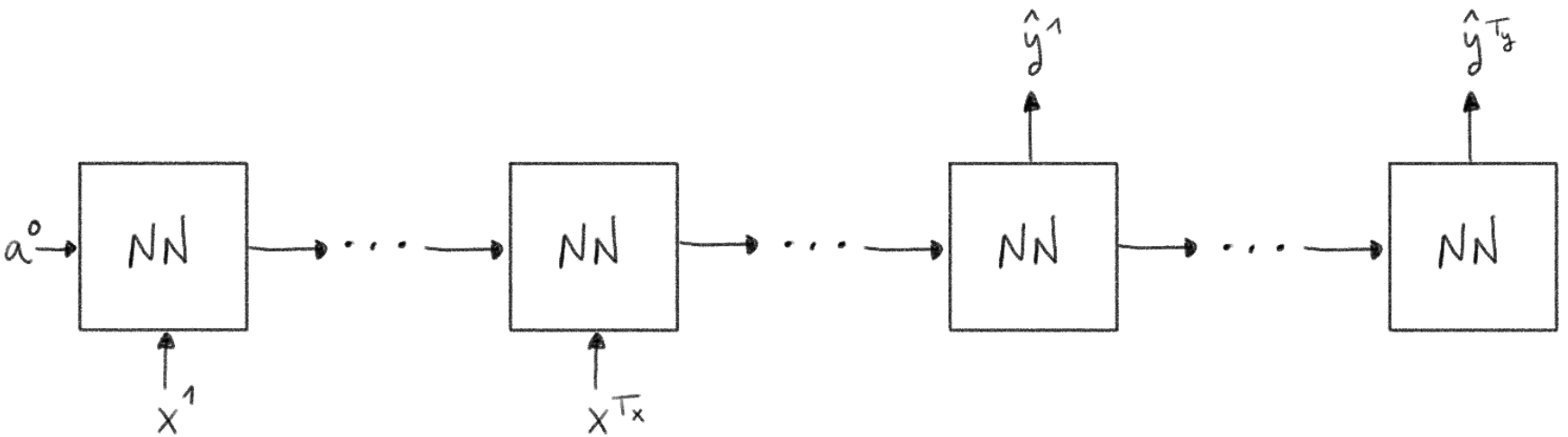
$T_x > 1, T_y = 1$ e.g. sentiment analysis

RNN Types: many-to-many (same length)



$T_x = T_y > 1$ e.g. name entity recognition

RNN TYPES: many-to-many (different length)



$T_x \neq T_y > 1$ e.g. machine translation

TRAINING

At each time step t:

$$a^{t} = \tanh\left(W_{aa} a^{t-1} + W_{ax} x^{t} + b_a \right)$$

$$\hat{y}^{t} = \text{softmax}\left(W_{ya} a^{t} + b_y \right)$$

over what?

Training: Loss Function

$$\mathcal{L}_\theta(\hat{y}, y) = \sum_{t=1}^{T} L_\theta(\hat{y}^t, y^t)$$

↑ parameters of the RNN,

i.e., $W_{aa}, W_{ax}, W_{ya}, b_a, b_y$

Example: Negative log-likelihood

$$L_\theta(\hat{y}^t, y^t) = -\log \underbrace{P_{model}(y^t \mid \{x^1, \ldots, x^t\})}_{= \hat{y}^t(y^t)}$$

Training: Backpropagation Through Time (BPTT)

$$\mathcal{L}_\theta(\hat{y}, y) = \sum_{t=1}^{T} L_\theta(\hat{y}^t, y^t)$$

$$\nabla_\theta \mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T} \nabla_\theta L_\theta(\hat{y}^t, y^t)$$

Careful! The dependence of $\hat{y}^t$ on θ is quite complex

→ Runtime of BPTT is $\mathcal{O}(T)$

Memory cost of BPTT is $\mathcal{O}(T)$

EXERCISE : BPTT FOR LINEAR RNN

Assume: $a^t = W_{aa}\, a^{t-1} + W_{ax}\, x^t$

$$\hat{y}^t = W_{ya}\, a^t$$

and compute : $\frac{\partial}{\partial a^t} L(\hat{y}^T, y^T)$ for the MSE loss, for $t \leq T$: i.e.

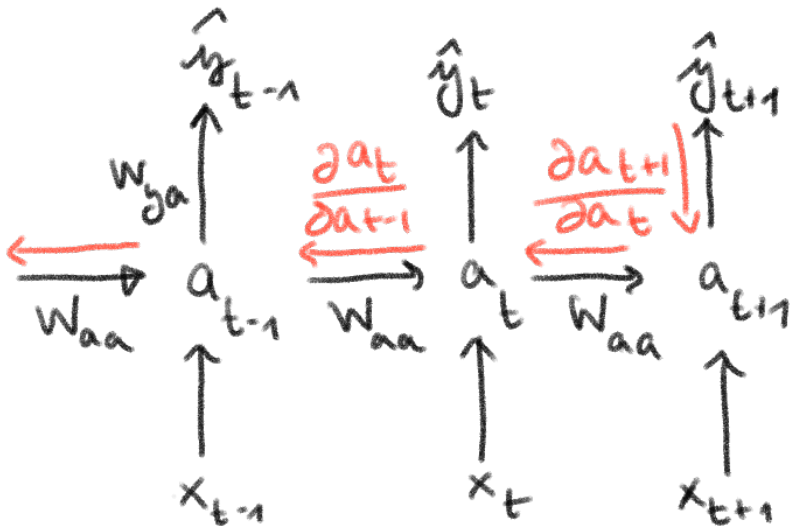
$$L(\hat{y}^T, y^T) = \frac{1}{2} |\hat{y}^T - y^T|^2$$

PROBLEMS WITH BPPT

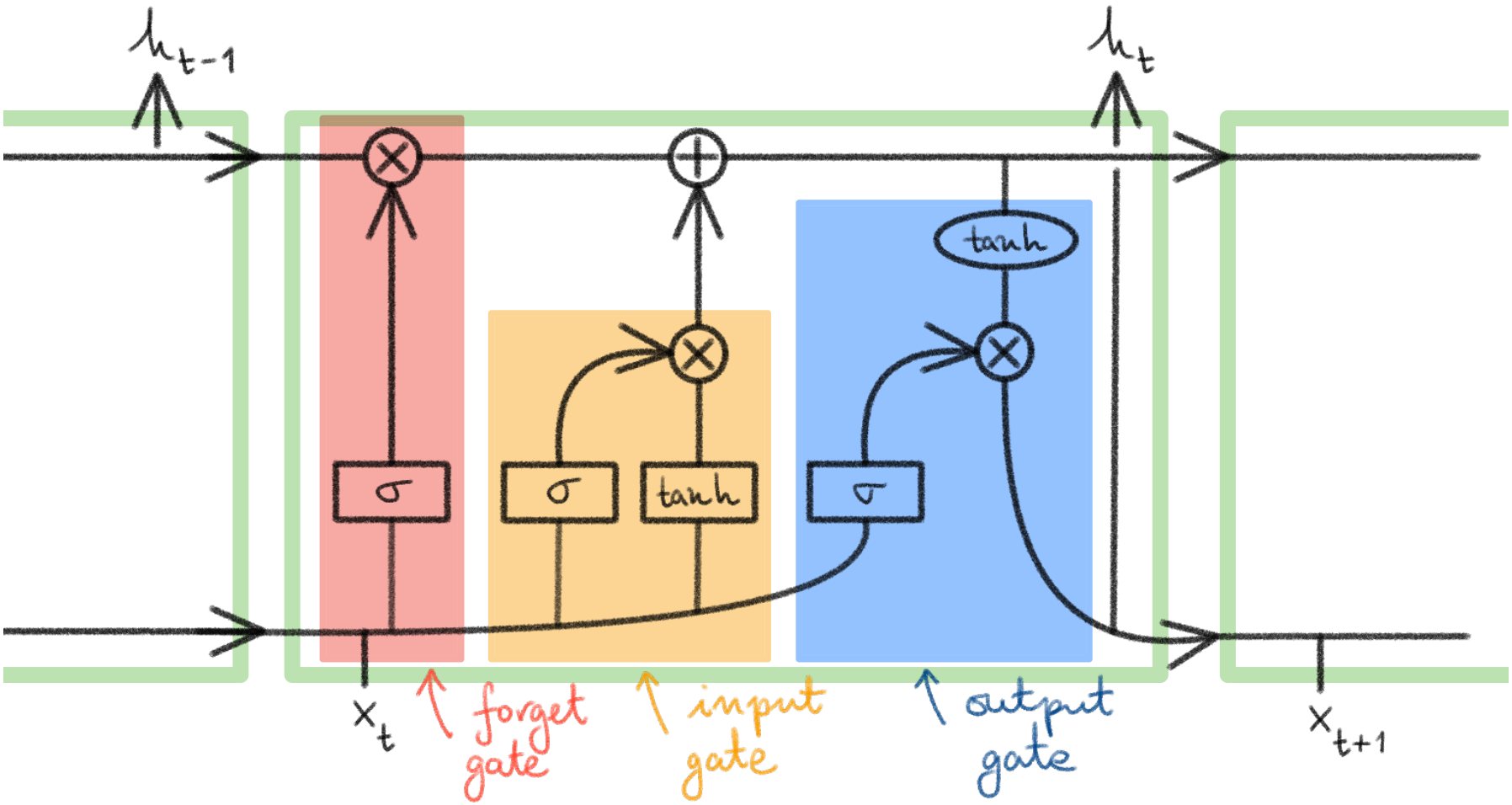
- Exploding gradient ⇒ very bad for learning
- Vanishing gradient ⇒ very short memory

Graph view of a RNN

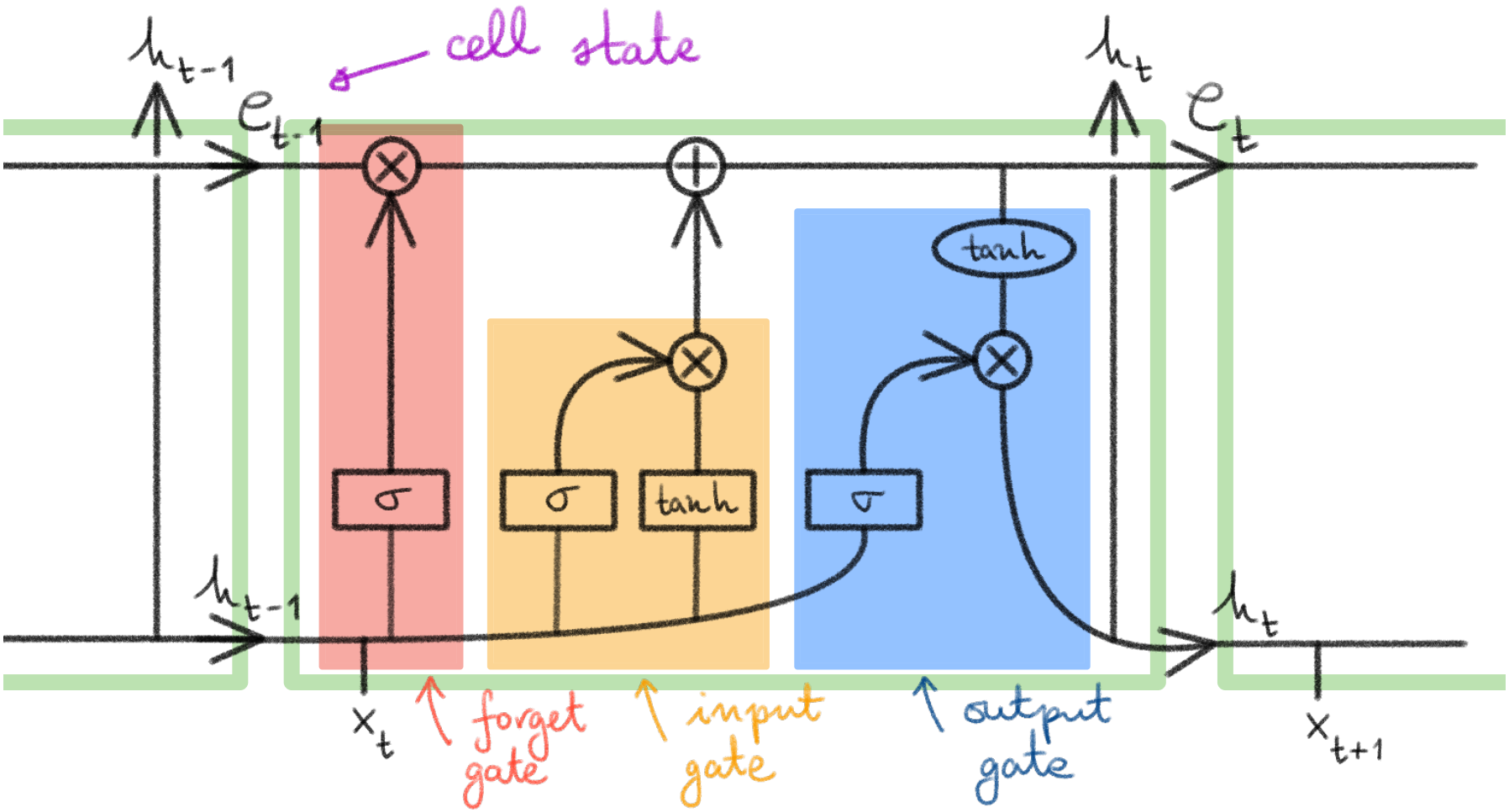
unfolding ⇒



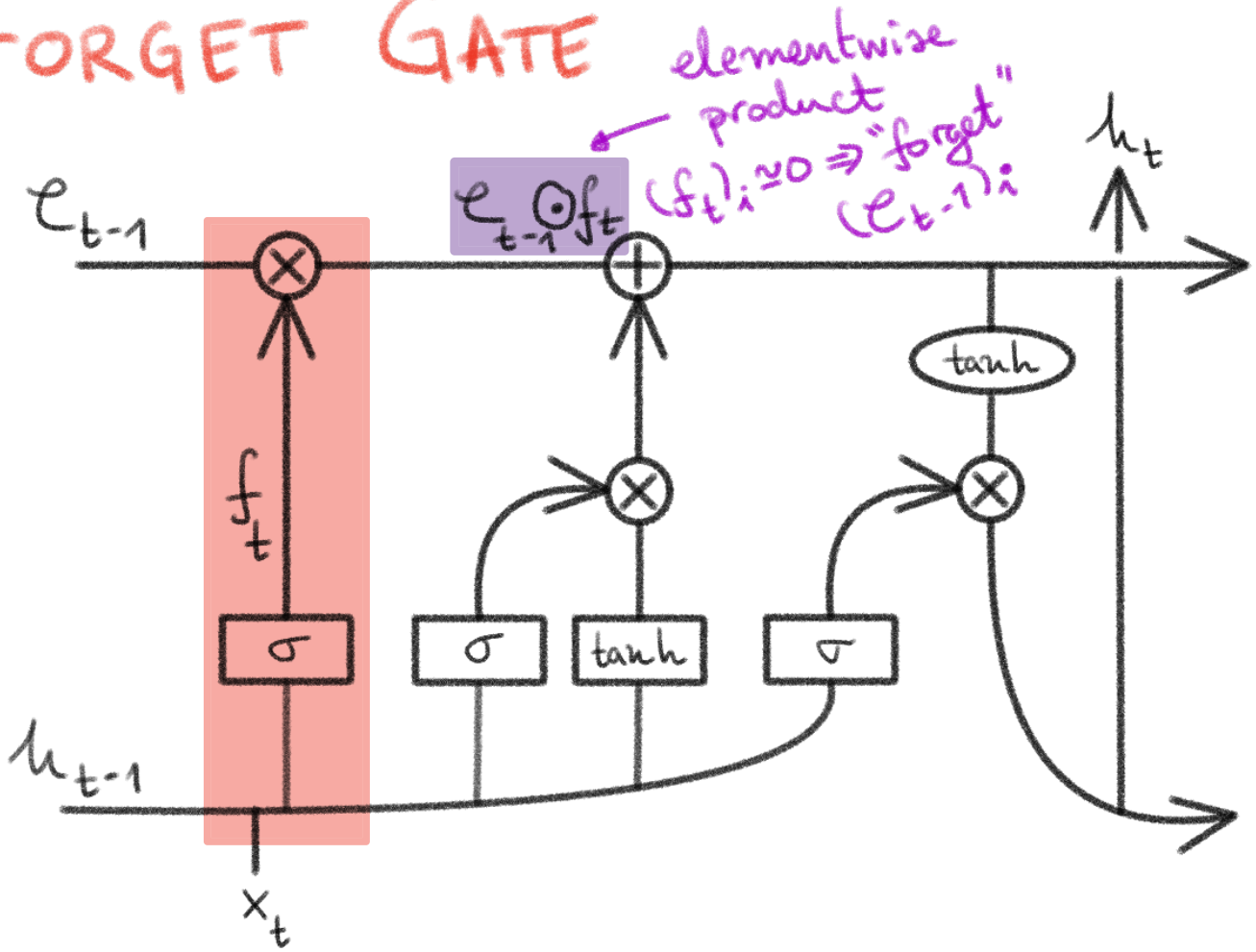
Long Short Term Memory (LSTM)



Long Short Term Memory (LSTM)

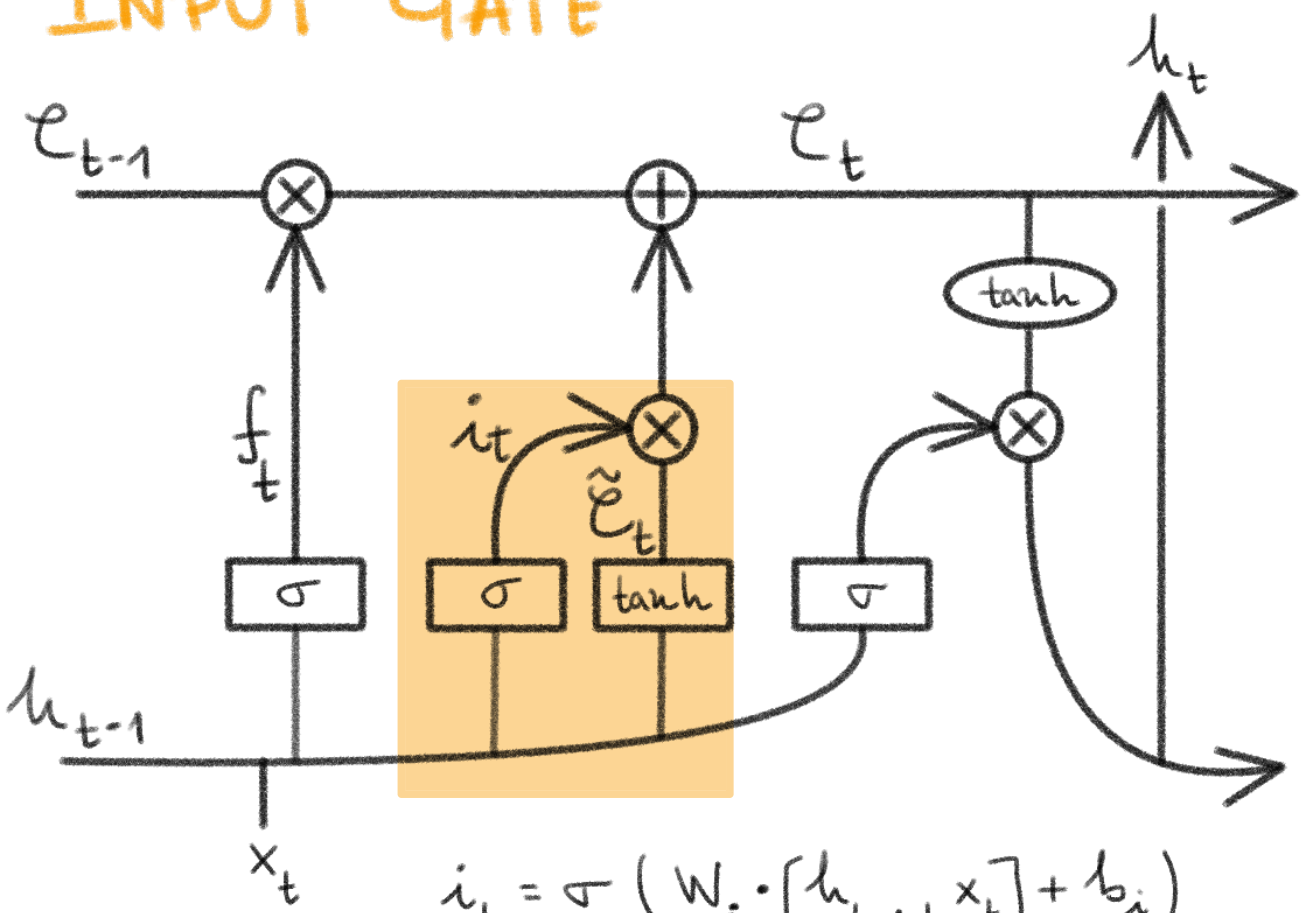


Forget Gate



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

Input Gate

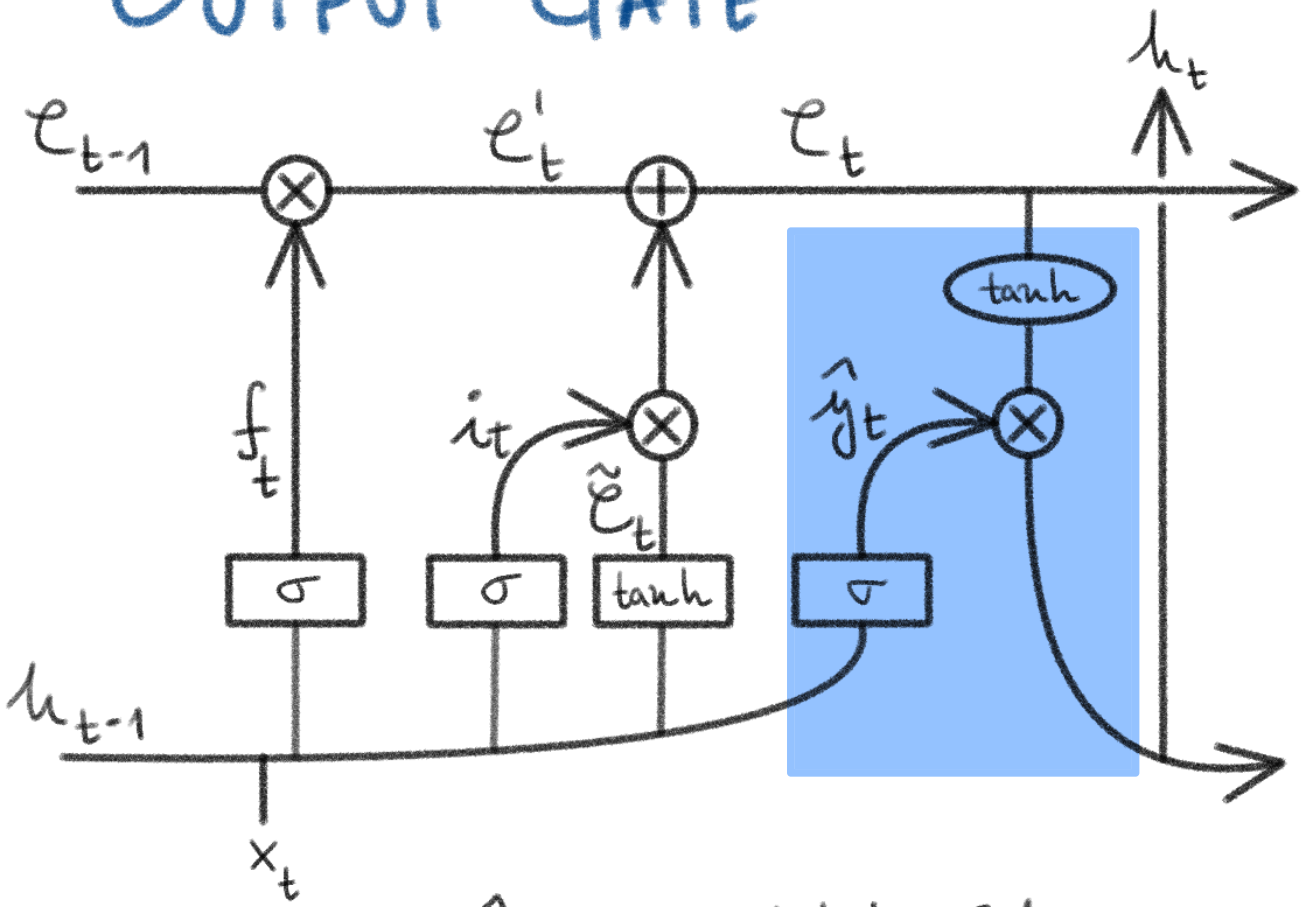


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] + b_i\right)$$

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

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

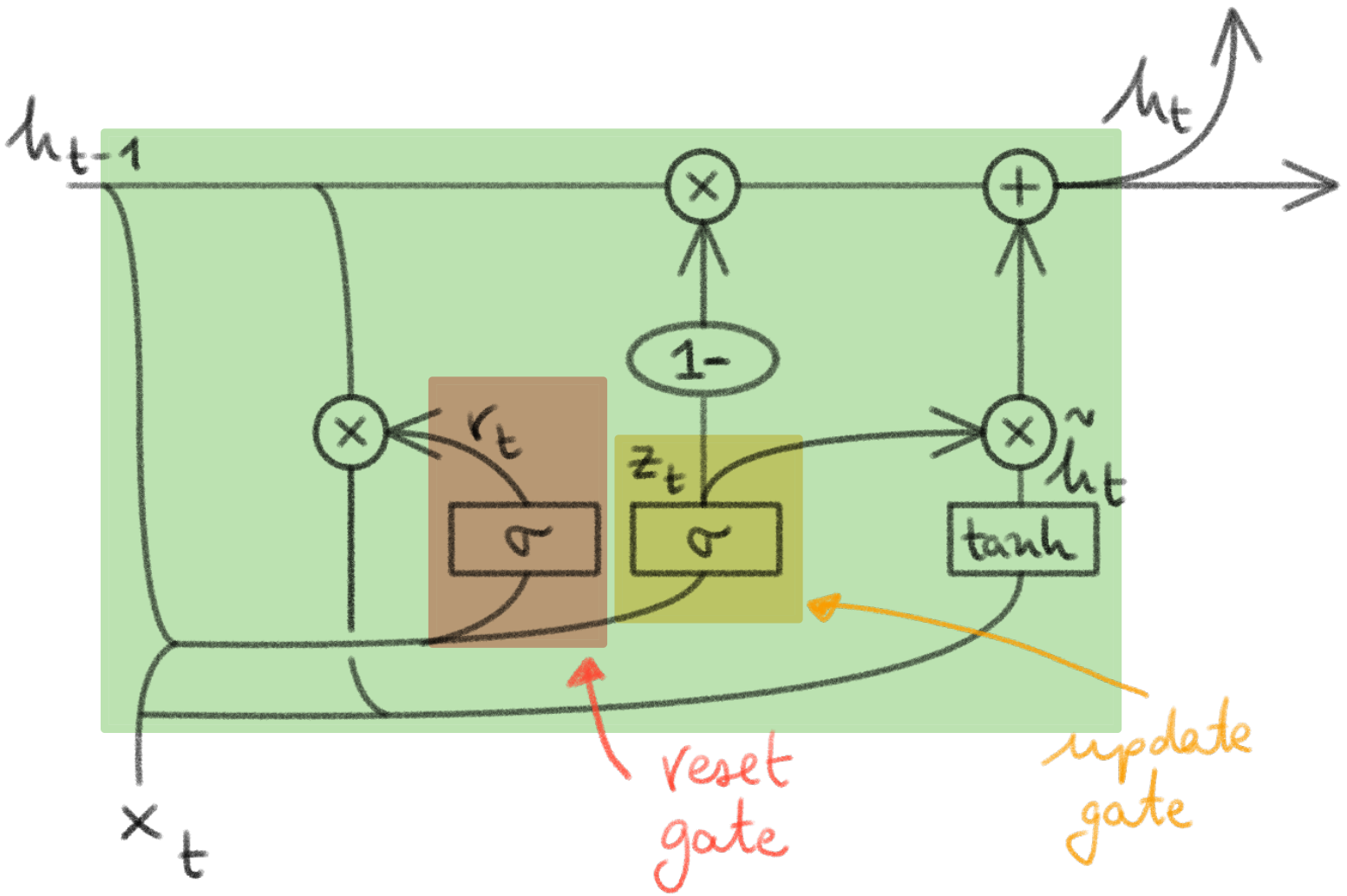
Output Gate



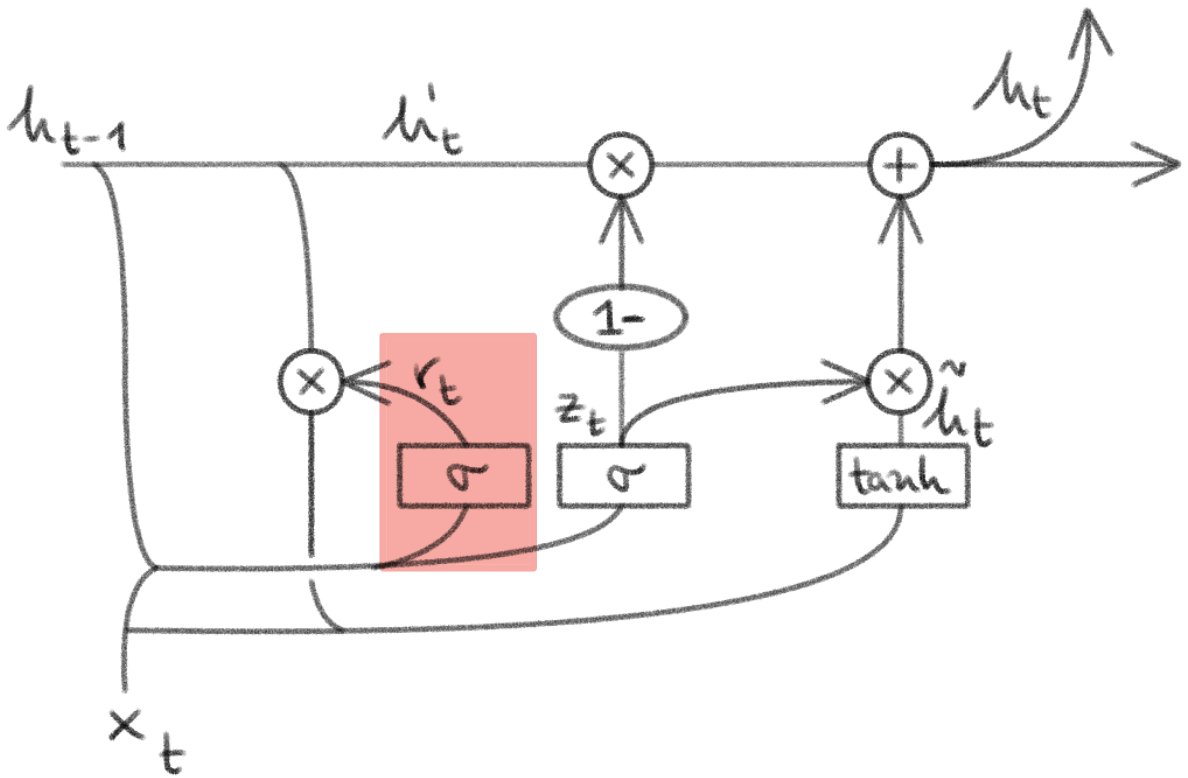
$$\hat{y}_t = \sigma\left(W_y \odot [h_{t-1}, x_t] + b_y\right)$$

$$h_t = \hat{y}_t \odot \tanh(c_t)$$

Gated Recurrent Unit (GRU)

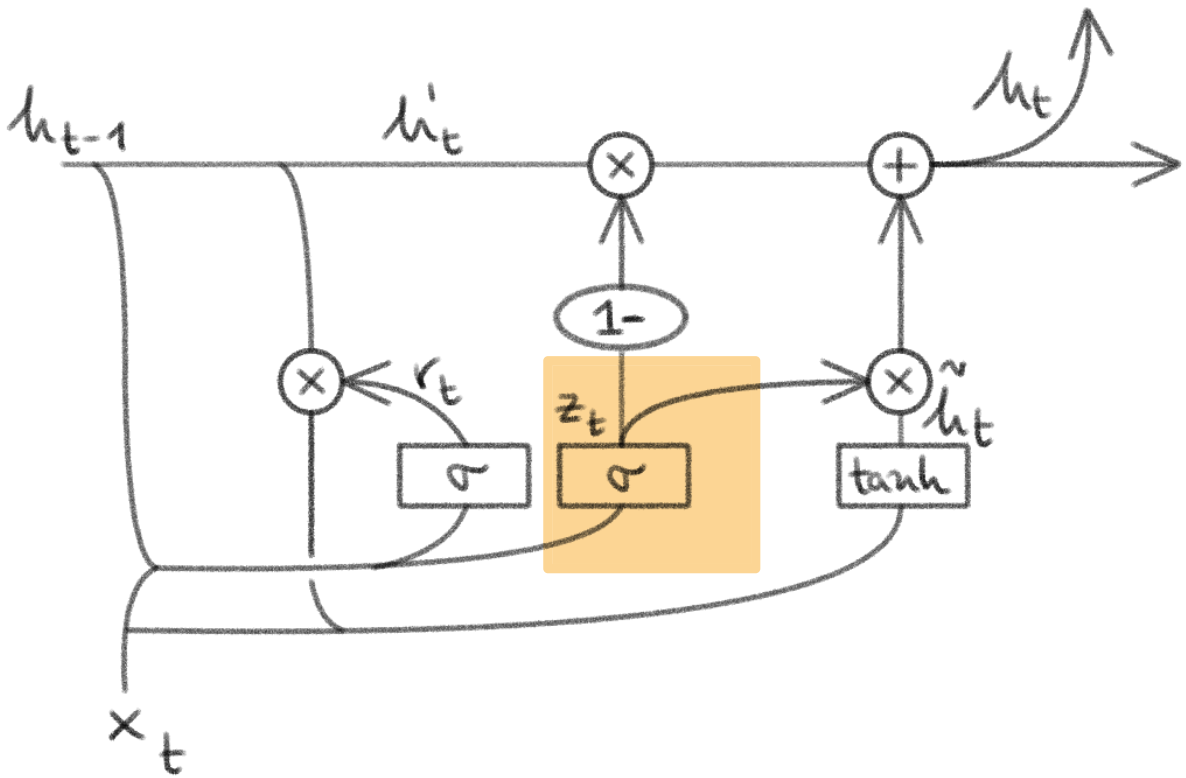


Reset Gate



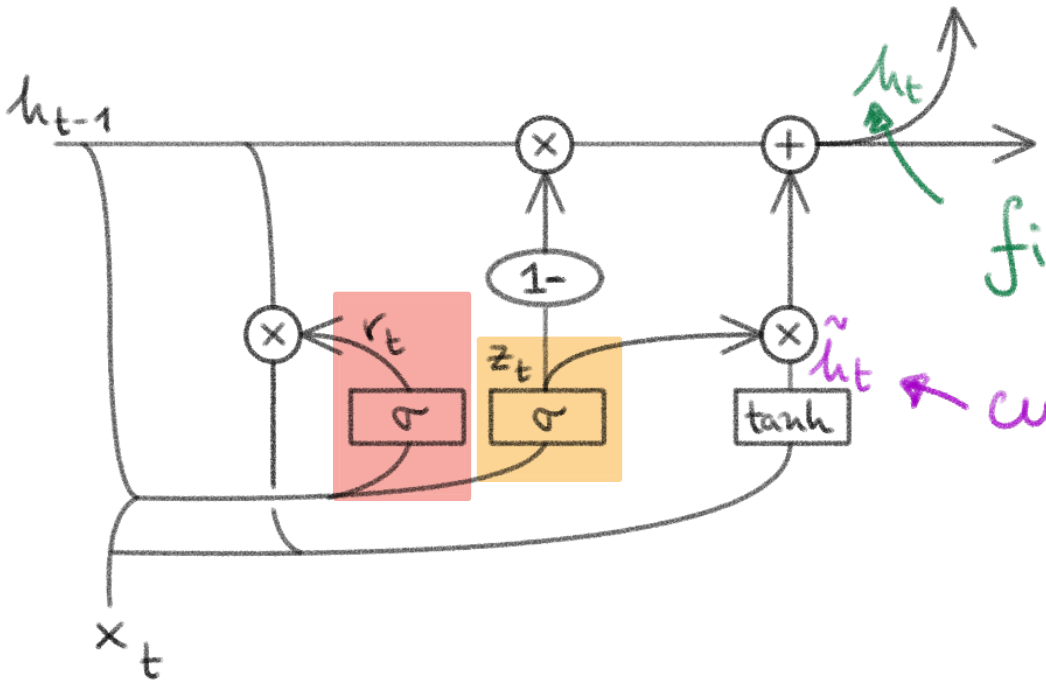
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t] + b_r\right)$$

Update Gate



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

Gated Recurrent Unit (GRU)



final memory

current memory content

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

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

Kahoot Time!

Are RNNs / LSTMs still used?

Yes, in the following contexts:

- Time-series forecasting
  (e.g. finance, trading)
- On smaller dataset
- When hardware / latency constraints

Conclusion

- Draw the (unfolded) graph of a 1-to-many RNN
- Write down the formulas for the 0-th, t-th and T-th step in a 1-to-many RNN
- Write down a question about something you did not fully understand or you would like to know more about.

Further Reading

"Introduction to RNNs" in Jeremy Jordan's blog

"The Unreasonable Effectiveness of RNNs" in Andrej Karpathy's blog