

## 1. Basic RNN – the starting point

- Takes previous hidden state + new input → makes new hidden state
- Very simple loop:  $h_t = \tanh(W_{hh} \times h_{t-1} + W_{xh} \times x_t + b)$

## Big problem: Vanishing Gradient

During backpropagation through time (BPTT), we keep multiplying by the same weight matrix many times. If those weights are  $< 1$ , the gradient becomes super tiny very quickly → the network stops learning long-distance patterns. (Gradients → 0 → weights almost don't update → forgets what happened many steps ago)

## 2. Requirement

We wanted the network to remember important information for a **long time** and forget unimportant stuff.

Two popular improvements appeared:

- **GRU** (Gated Recurrent Unit) – simpler & faster
- **LSTM** (Long Short-Term Memory) – more powerful, became very popular

Both use **gates** to control what information to keep / throw away / update.

## 3. LSTM the most famous

An **LSTM cell** has **three main gates** + a cell state (like a conveyor belt that carries long-term memory).

Main parts inside one LSTM cell:

1. **Cell state ( $C_t$ )** → This is the long memory highway. It can carry information almost unchanged over many time steps.
2. **Forget Gate**  $f_t = \sigma(W_f \times [h_{t-1}, x_t] + b_f)$  → Decides **what to forget** from the old cell state  $C_{t-1}$ . Outputs numbers between 0 and 1: 0 = completely forget this part  
1 = keep everything
3. **Input Gate** (also called Update Gate sometimes) Two parts:  $i_t = \sigma(W_i \times [h_{t-1}, x_t] + b_i)$  ← decides **what new info to add**  
 $\tilde{C}_t = \tanh(W_C \times [h_{t-1}, x_t] + b_C)$  ← the actual new candidate values

→ Only important new information gets added to the cell state.

4. **Output Gate**  $o_t = \sigma(W_o \times [h_{t-1}, x_t] + b_o)$   $h_t = o_t \odot \tanh(C_t)$  → Decides **what parts of the cell state to output** right now (to the next cell and as hidden state).