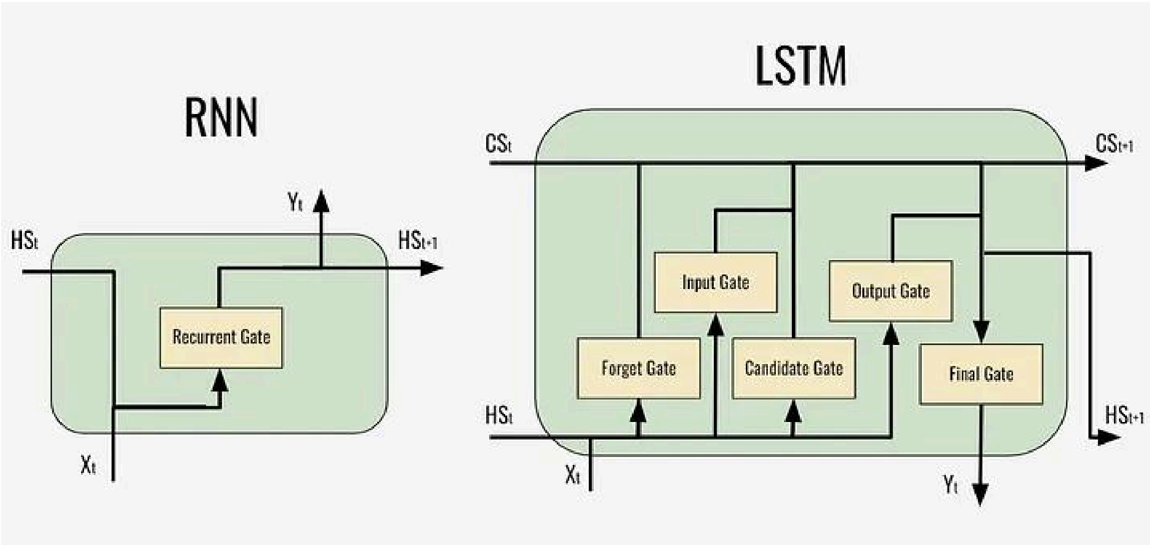
📌 Long Short Term Memory (LSTM)

LSTM is an advanced type of Recurrent Neural Network (RNN) designed to handle long-term dependencies in sequential data. Unlike traditional RNNs, LSTMs use **memory cells** and **gating mechanisms** to retain important information while preventing vanishing gradient issues.

✒️ Components of LSTM:

An LSTM unit consists of the following key components:

- **Cell State (Ct):** This acts as the memory of the LSTM, carrying information across time steps.
- **Candidate Cell State (C̄t):** A new potential value for the cell state.
- **Forget Gate (ft):** Decides what part of the previous memory to discard.
- **Input Gate (it):** Determines which new information to store.
- **Output Gate (ot):** Controls what information is output at each time step



✒️ Working of LSTM with Gates::

Step 1: Forget Gate (ft) – Removing Irrelevant Information

- Determines which part of the past memory (Ct−1) should be forgotten.
- Uses a sigmoid activation function (σ) to output values between 0 (forget completely) and 1 (keep completely).

Equation:

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

where:

- W_f and b_f are weight matrix and bias for the forget gate.
- h_{t-1} is the previous hidden state.
- x_t is the current input.

Example Use Case: In a sentiment analysis model, if a new sentence begins, past irrelevant words may be forgotten.

Step 2: Input Gate (it) – Deciding What New Information to Store

- Determines which parts of the **new input** xt should be added to the memory.
- Uses a **sigmoid function** to decide what to update.
- Uses a **tanh function** to create new candidate values.

Equations:

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

$$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$$

where:

- $\tilde{C}_t$ is the candidate memory update.

Step 3: Cell State Update (Ct) – Updating Long-Term Memory

- The cell state is updated using the forget and input gates:

Equation:

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

This ensures important past information is retained while incorporating new relevant information.

Step 4: Output Gate (ot) – Deciding What to Output

- Controls what information from the current cell state should be passed as the output.
- Uses a **sigmoid function** to decide the importance.
- Uses a **tanh function** to regulate the output range.

Equation:

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

where h_t is the new hidden state that will be passed to the next time step.

✒️ Summary of LSTM Equations:

Gate	Equation	Purpose
Forget Gate	$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f)$	Decide what to forget from past memory.
Input Gate	$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i)$	Decide what new information to add.
Candidate Memory	$\tilde{C}_t = \tanh(W_c[h_{t-1}, x_t] + b_c)$	Generate new memory values.
Cell State Update	$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$	Update long-term memory.
Output Gate	$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o)$	Decide what to output.
Final Hidden State	$h_t = o_t * \tanh(C_t)$	Compute the new hidden state.

📌 Gated Recurrent Unit (GRU)

GRU is a type of **Recurrent Neural Network (RNN)** similar to **LSTM** but with a **simpler architecture**. It helps solve the **vanishing gradient problem** and efficiently captures **long-term dependencies** in sequential data.

✒️ How Gated Recurrent Unit (GRU) Works?:

GRU has **two gates** instead of three (compared to LSTM):

- **Reset Gate (rt)** – Decides how much of the past information to forget.
- **Update Gate (zt)** – Controls how much of the new information should be kept.

Unlike LSTMs, **GRUs do not have a separate cell state**; instead, they directly update the **hidden state ht**.

✒️ Key Difference between RNN, LSTM and GRU:

Feature	RNN (Vanilla)	LSTM (Long Short-Term Memory)	GRU (Gated Recurrent Unit)
Handles Long-Term Dependencies?	❌ No (suffers from vanishing gradient)	✅ Yes (memory cell & gates)	✅ Yes (simplified gating mechanism)
Memory Control?	❌ No explicit control	✅ Uses forget, input, and output gates	✅ Uses reset and update gates
Computational Efficiency	✅ Fast (simple structure)	❌ Slower (more parameters)	⚖️ Faster than LSTM, slower than RNN
Performance on Long Sequences?	❌ Poor	✅ Good	✅ Good
Parameter Complexity?	✅ Fewest parameters	❌ Most parameters	⚖️ Fewer parameters than LSTM