LSTM

✅ Here's what you're covering and why it works well:

Part	What You're Teaching	Why It's Important
1️⃣	Brief on **ANN**	Sets the base — students know how traditional networks work
2️⃣	Problem with ANN for sequence data	Shows the **need** for RNNs (no memory, fixed input size)
3️⃣	Introduction to **RNN**	Introduces recurrence, hidden state, time step-based processing
4️⃣	Problems with RNN (e.g., vanishing gradients)	Prepares the ground for why LSTM was invented
5️⃣	Introduction to **LSTM**	Logical next step — introduce as a solution
6️⃣	Basic LSTM Architecture	Gives structural understanding — memory cell + gates
7️⃣	Overview of **Forget Gate, Input Gate, Output Gate**	Core components — gives interpretability and depth

📘 Introduction to LSTM (Long Short-Term Memory)

🧠 What is LSTM?

LSTM is a special type of **Recurrent Neural Network (RNN)** designed to **remember information for long periods**.

- Introduced by **Hochreiter & Schmidhuber (1997)**
- It solves key issues in traditional RNNs such as:
  - **Vanishing gradient**
  - **Difficulty in learning long-term dependencies**

🔁 Why was LSTM needed?

Let's first understand the problems with RNNs.

❌ Problems in RNN

---

1. Short-Term Memory

- RNNs can only remember **recent** inputs.
- Struggle with remembering **long-range dependencies** (e.g., "The boy who wore a red hat... *was* playing.")

---

2. Vanishing Gradient Problem

- During backpropagation through time (BPTT), gradients are **repeatedly multiplied by weights < 1**.
- This causes the gradients to shrink exponentially.
- Eventually, **weights stop updating** → the model forgets earlier information.

---

3. Exploding Gradient Problem (less common)

- If weights > 1, gradients grow too large → unstable training.

---

4. Difficulty in Capturing Long-Term Context

- Language and time series often require remembering context far back (e.g., meaning of a sentence).
- RNNs just can't **bridge that gap** effectively.

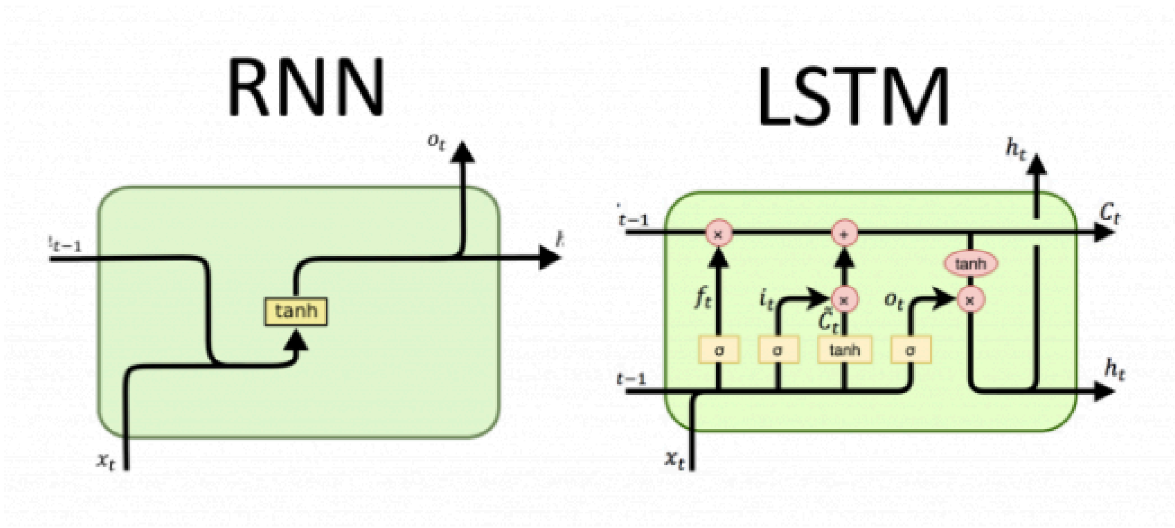
✅ How LSTM Solves These Problems

🔑 Core Idea:

Instead of having just one hidden state like RNN, LSTM introduces a **Cell State (Ct)** — a **"conveyor belt"** that carries information through time steps **with minimal modification**.

This lets LSTM **retain long-term memory**.

🏗️ LSTM Architecture Components



LSTM uses **three gates** to regulate the flow of information:

---

1. Forget Gate (ft)

- **Decides what to forget** from the previous cell state.
- Looks at ht−1h_{t-1}ht−1 and xtx_txt, and outputs a number between 0 and 1.
- 0 = forget completely, 1 = keep everything.

---

2. Input Gate (it)

- **Decides what new information** to store in the cell state.
- Two parts:
  - A sigmoid layer: which values to update
  - A tanh layer: creating candidate values

---

3. Cell State Update

This line is the **heart of LSTM**.
It controls **what to forget + what to add** into memory.

---

4. Output Gate (ot)

- **Decides what part of the memory to output**

📌 Summary Table:

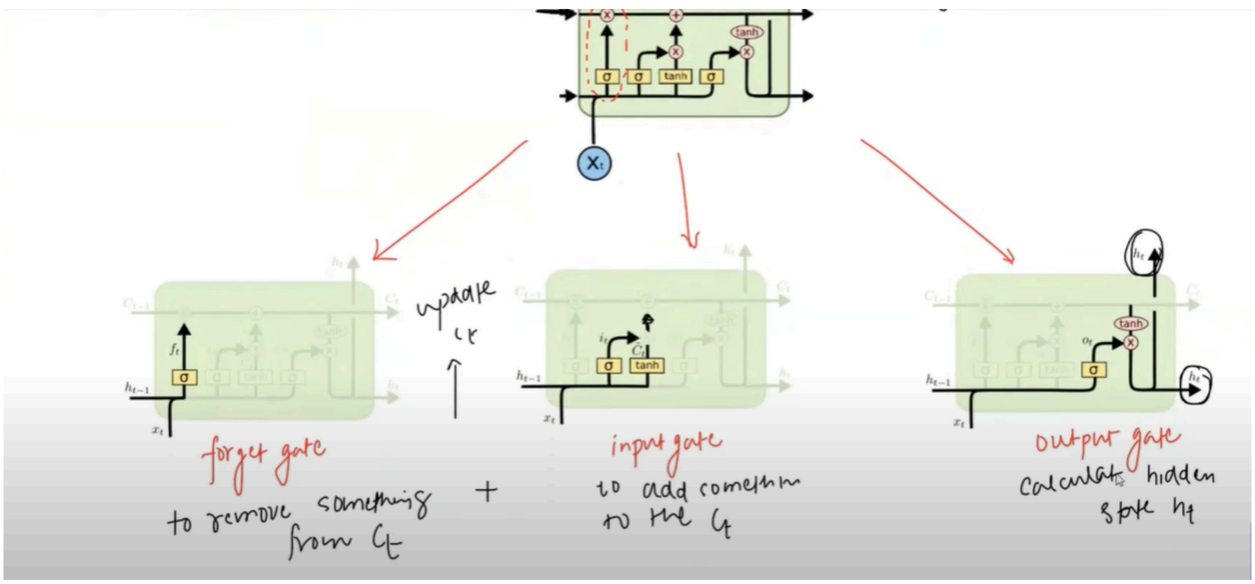
Problem in RNN	How LSTM Solves It
Can't remember long-term info	Cell state carries long-term memory
Vanishing gradient	Additive updates to cell state, avoids exponential shrinkage
Forgetting earlier inputs	Forget gate learns to retain or discard information
Inflexible memory update	Input + Output gates allow precise memory control

---

🧪 Intuition Recap

- **Forget gate**: What to throw away?
- **Input gate**: What to store?
- **Output gate**: What to pass to the next time step?

Think of it like a **smart memory box** with gates deciding what goes in, what stays, and what comes out.


tanh
X_t
update c_t
C_{t-1}
C_t
h_t
f_t
h_{t-1}
x_t
σ
tanh
i_t
$\tilde{C}_t$
o_t
forget gate
to remove something from C_t
+
input gate
to add somethin to the C_t
output gate
calculate hidden state h_t