Problems with RNN

- RNNs → Suitable for sequential data
- They're not used much
- **LSTM are used more**

Problem 1: Problem of long term dependency

- If sequence is big, the previous memory is lost.
- It arises due to **vanishing gradient problem**
- RNNs are **supposed** to remember useful information from earlier in the sequence and use it later.

But in reality, they **forget** long-term information very easily.

> Think of reading a paragraph and forgetting what the first sentence said by the time you reach the last.

🧠 Why does this happen?

This is due to the **vanishing gradient problem** during backpropagation.

Let's break it down:

- During training, RNNs use **Backpropagation Through Time (BPTT)**.
- This means gradients (error signals) are passed **backward through time steps**.
- In long sequences, gradients **shrink (vanish)** as they pass back through each step.

So:

- Early steps in the sequence get **tiny gradients**, close to zero.

- The weights connected to early steps **don't get updated** effectively.
- That means the RNN **can't learn** to connect distant information in the sequence.

🔬 Real Example:

Imagine you're translating:

> "The boy who had a dog and a cat and went to the park and met a friend is very happy."

To understand the word **happy**, the model should remember **boy** from much earlier.

But vanilla RNNs forget that context due to vanishing gradients.

Problem 2: Problem of Stagnated training

- Many times, we cannot train RNNs
- This happens due to **exploding gradient problem**
- Sometimes during training, the gradients (error signals) **grow too large** — they **explode**.

This causes:

- Very large updates to weights
- **Instability** in training
- Loss becomes **NaN** (Not a Number)
- The model **fails to learn** — it's like giving the model too much caffeine; it loses control

🧠 Why does this happen?

Just like gradients can vanish (become too small), they can also **explode (become too big)**.

This happens especially in deep/unrolled networks (like long RNN sequences) where:

- Small errors multiply across many time steps
- They grow exponentially → weights go out of control