RNNS

**NAME: AKSHAT DEV**

**ME24B003**

**TASK -2.1**

Recurrent Neural Networks (RNNs) are a type of neural network designed to process sequential data. Unlike traditional neural networks, which handle each input independently, RNNs have a built-in memory that allows them to retain information from previous steps. This makes them useful for tasks like speech recognition, text generation, and time series prediction.

How RNNs Work ?

RNNs process data step by step, where each step’s output depends not only on the current input but also on past inputs. This is achieved through a hidden state, which carries information from previous steps. The network updates this hidden state at each step to refine its understanding of the sequence.

At each time step, the RNN performs the following operations:

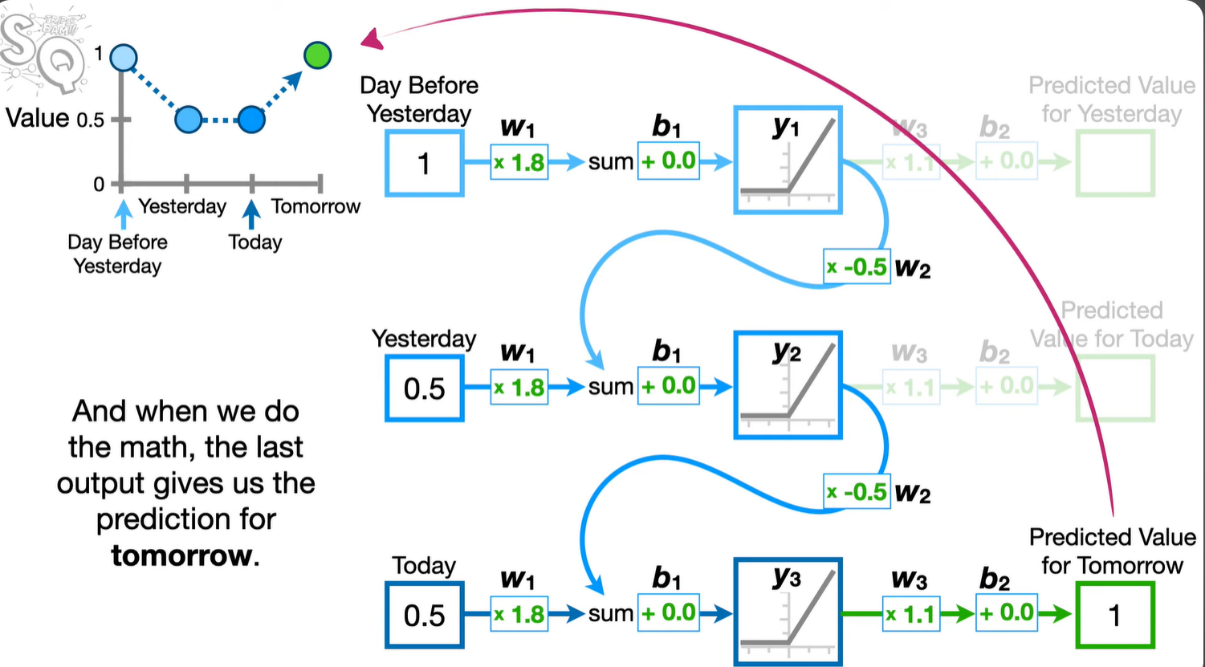
1. Receives an input from the sequence.
2. Combine it with the previous hidden state using a weighted transformation and an activation function (usually tanh).
3. Generates an output based on the updated hidden state.
4. Passes the hidden state to the next step to retain memory.

This loop continues until the entire sequence is processed. The network then makes predictions based on the final hidden state.

Challenges of RNNs

A major issue with standard RNNs is the vanishing gradient problem. When training long sequences, the gradients (which help update the model’s weights) become very small, making it hard for the network to remember information from earlier steps. This limits its ability to learn long-term dependencies.

RNN Architecture



* Input layer – Takes in the sequence data one step at a time.
* Hidden layer – Stores and updates memory using recurrent connections.
* Output layer – Produces predictions based on the processed sequence.

Because of their memory limitations, advanced versions like LSTMs (Long Short-Term Memory) and GRUs (Gated Recurrent Units) were developed to improve long-term learning.

LSTM

LSTM is an improved version of Recurrent Neural Networks (RNNs) designed to retain long-term information while avoiding the vanishing gradient problem. It is useful for tasks like speech recognition, text generation, and time series forecasting.

How LSTM Works ?

LSTMs introduce a cell state and gates to control information flow. These help decide what to keep, update, or discard at each step, allowing the network to focus on relevant data while forgetting unnecessary details.

At each time step, LSTMs perform the following:

1. Forget gate – Determines which past information to discard.
2. Input gate – Decides what new information to store in memory.
3. Cell state update – Updates the memory based on relevant new data.
4. Output gate – Controls what part of the memory contributes to the output.

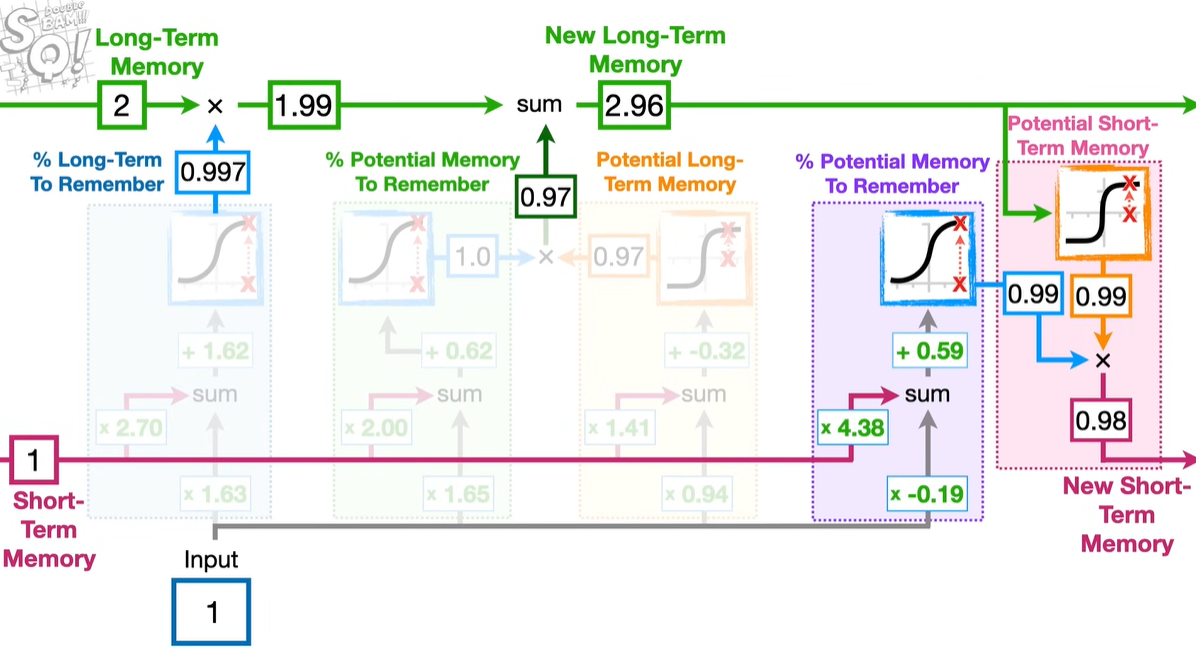
LSTM Architecture ?

* Input layer – Receives sequential data step by step.
* Gates (Forget, Input, Output) – Manage memory retention and updates.
* Cell state – Stores long-term memory and gets updated at each step.
* Output layer – Produces final predictions.

By effectively managing memory, LSTMs capture long-term dependencies better than standard RNNs

Long Short-Term Memory (LSTM) networks are highly effective at learning long-term dependencies. The use of input, forget, and output gates provides better control over the memory cell, helping to address challenges like the vanishing gradient problem.

However, they are more complex than standard RNNs, requiring additional parameters, which increases computational costs.



GRU’S

GRUs are a simplified version of Recurrent Neural Networks, designed to enhance and refine the architecture of LSTMs. They provide an efficient way to process sequential data, especially in tasks requiring long-term memory retention.

It consists of two gates :-

Update Gate: This gate decides the extent to which the information from the previous state should be carried over to the current state. It is a blend of the forget and input gates found in LSTMs.

 Reset Gate: It determines how much of the past information to forget, effectively allowing the model to decide how much of the past information is relevant for the current prediction.

GRUs are effective in handling long-term dependencies, similar to LSTMs, but with a simpler design. By combining the input and forget gates into a single update gate, they reduce architectural complexity.

They are generally faster to train than LSTMs since they have fewer parameters while still delivering comparable performance in many cases.