Recurrent Neural Network (RNN) – Step by Step Explanation

1. Definition

A **Recurrent Neural Network (RNN)** is a special type of neural network used to process **sequential data**, where the current output depends on both the current input and previous inputs.

Examples of sequence data:

- Words in a sentence (text)
- Sound waves (speech)
- Daily stock prices or temperature readings

Simple meaning: RNNs have a kind of "memory" that remembers what happened before.

2. Why RNN is Needed

A normal neural network (ANN) treats each input independently. But in real life, many tasks depend on **context** or **previous information**.

Example: To understand the meaning of the word "bank":

- In "river bank" → related to nature
- In "money bank" → related to finance

The meaning depends on earlier words, so the network must remember what came before — this is why we use RNNs.

3. How RNN Works

In an RNN, information flows not only forward but also loops back to itself.

At each time step t:

$$h_t = \tanh(W_x x_t + W_h h_{t-1} + b_h)$$
$$y_t = \sigma(W_y h_t + b_y)$$

Where:

- x_t = current input
- h_t = hidden state (memory at current step)
- h_{t-1} = memory from previous step
- W_x, W_h, W_y = weight matrices
- b_h, b_y = biases
- tanh and σ = activation functions

Meaning: Each step's output depends on both the new input and what the network remembers from before.

4. Step-by-Step Numerical Example

Let's calculate how an RNN works on a tiny sequence.

Given:

$$W_x = 0.6, \quad W_h = 0.4, \quad b_h = 0.0, \quad W_y = 1.0, \quad b_y = 0.0$$

Initial hidden state:

$$h_0 = 0$$

Input sequence:

$$x_1 = 1.0, \quad x_2 = 2.0, \quad x_3 = -1.0$$

—

Step 1: Time $t = 1$

$$h_1 = \tanh(W_x x_1 + W_h h_0 + b_h)$$

$$h_1 = \tanh(0.6 \times 1.0 + 0.4 \times 0 + 0) = \tanh(0.6) = 0.537$$

$$y_1 = W_y h_1 = 1.0 \times 0.537 = 0.537$$

So at time 1:

$$h_1 = 0.537, \quad y_1 = 0.537$$

—

Step 2: Time $t = 2$

$$h_2 = \tanh(W_x x_2 + W_h h_1)$$

$$h_2 = \tanh(0.6 \times 2.0 + 0.4 \times 0.537) = \tanh(1.2 + 0.2148) = \tanh(1.4148) = 0.888$$

$$y_2 = W_y h_2 = 0.888$$

So at time 2:

$$h_2 = 0.888, \quad y_2 = 0.888$$

—

Step 3: Time $t = 3$

$$h_3 = \tanh(W_x x_3 + W_h h_2)$$

$$h_3 = \tanh(0.6 \times (-1.0) + 0.4 \times 0.888) = \tanh(-0.6 + 0.3552) = \tanh(-0.2448) = -0.240$$

$$y_3 = W_y h_3 = -0.240$$

So at time 3:

$$h_3 = -0.240, \quad y_3 = -0.240$$

—

5. Step-by-Step Summary Table

Time (t)	Input (x_t)	Hidden State (h_t)	Output (y_t)
1	1.0	0.537	0.537
2	2.0	0.888	0.888
3	-1.0	-0.240	-0.240

Interpretation:

- The RNN updates its hidden state each time using both the new input and the previous memory.
- Even when x_3 is negative, the network "remembers" that the previous state (h_2) was high (positive), so h_3 does not drop too much.
- This shows how the RNN mixes new and old information.

6. Real-Life Example: Predicting Weather Patterns

Imagine we want to predict the weather (sunny or rainy) based on the last few days' temperatures.

Inputs:

$$x_1 = 30^\circ\text{C}, \quad x_2 = 32^\circ\text{C}, \quad x_3 = 29^\circ\text{C}$$

We scale the temperatures to [0,1] range:

$$x_1 = 0.3, \quad x_2 = 0.32, \quad x_3 = 0.29$$

We use the same weights as before.

—

Step 1:

$$h_1 = \tanh(0.6 \times 0.3 + 0.4 \times 0) = \tanh(0.18) = 0.178$$

$$y_1 = h_1 = 0.178$$

Step 2:

$$h_2 = \tanh(0.6 \times 0.32 + 0.4 \times 0.178) = \tanh(0.192 + 0.071) = \tanh(0.263) = 0.258$$

$$y_2 = h_2 = 0.258$$

Step 3:

$$h_3 = \tanh(0.6 \times 0.29 + 0.4 \times 0.258) = \tanh(0.174 + 0.103) = \tanh(0.277) = 0.270$$

$$y_3 = h_3 = 0.270$$

—

Interpretation:

- The RNN "remembers" previous temperatures when predicting the current weather trend.
- If earlier days were hot, the network's memory (h_t) stays high, indicating a "sunny" trend.
- If several cooler days occur in a row, h_t decreases, showing a transition toward "rainy" weather.

7. Key Points

- RNNs have a memory of the past, stored in h_t.
- Each time step uses both the new input (x_t) and the old memory (h_{t-1}).
- They are useful for sequences: text, speech, stock prices, and weather.
- The main problem is the "vanishing gradient" — they can forget long-term patterns.

8. Summary

A **Recurrent Neural Network (RNN)** is a neural model that processes data one step at a time, using its hidden state to remember previous information. It can predict sequences, understand context, and learn temporal patterns, making it ideal for time-dependent tasks such as text, speech, or weather forecasting.