Module 6 | Recurrent Neural Networks (RNNs)

13 December 2025 11:55

What is RNN?

↳ Recurrent Neural Network

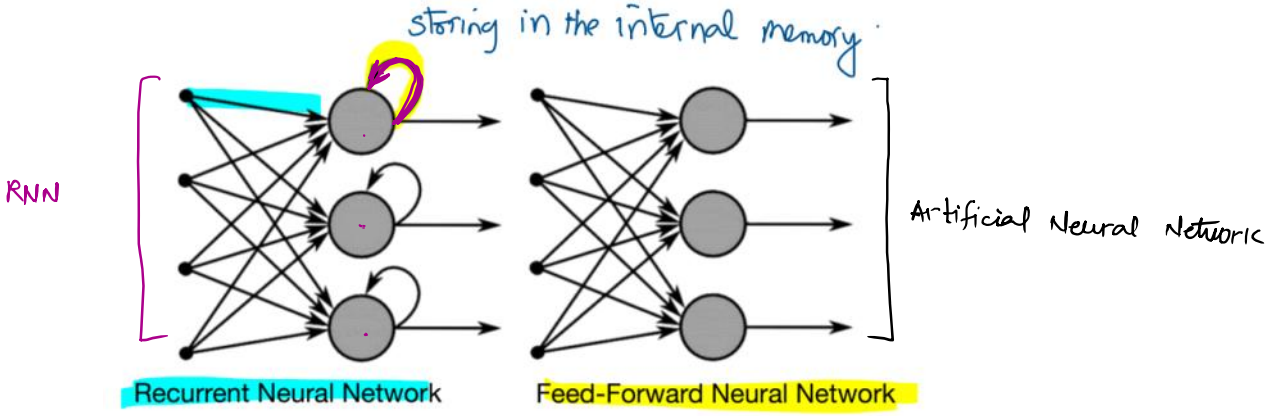
Why is it called recurrent?

- the term 'recurrent' itself in RNN comes from the repetition (or recurrence) of operations over time series/ time steps in a sequence.

In RNN,

- the same operation is applied repeatedly at every step.
- n/w feeds its output (hidden state) from the previous time step back into itself

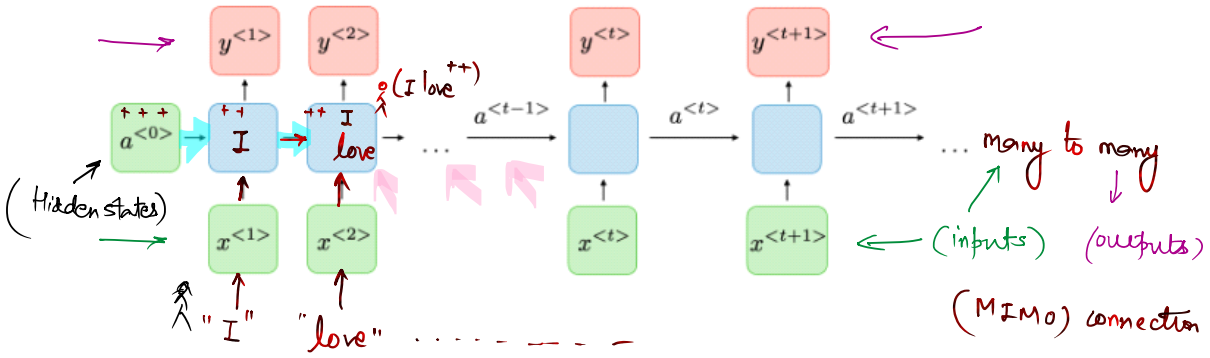
- this loop-like architecture allows it to remember previous information
  ↳ hence making it suitable for time-series data.



RNN allows previous outputs to be used as inputs while having hidden states

[STOCK PRICES] → RNN

"I love to teach deep learning".



① $x^{<t>}$ represents sequence of inputs

All the green boxes at the bottom denote the input sequences labelled as $x^{<1>}, x^{<2>}, x^{<3>}, \ldots x^{<t>}, x^{<t+1>}$

Say if input is a sentence then $x^{<t>}$ represents a word token in general.

② $a^{<t>}$: Hidden state

Blue boxes in the middle indicate/represent hidden states which maintain the memory of the nlw.

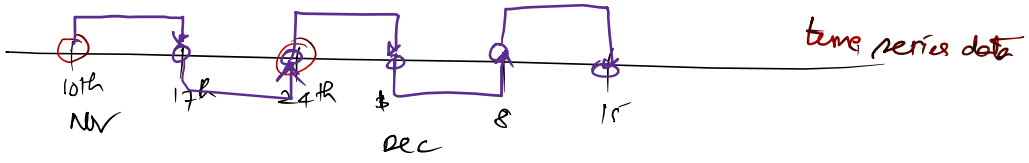
Each hidden state $a^{<t>}$ carries the information from the past inputs upto the current state 't'

③ Outputs $y^{<t>}$

Red boxes at the top denote the outputs of RNN at each time step labelled as $y^{<1>}, y^{<2>}, y^{<3>}$ ..... and so on.

RNN key features

① Sequential data processing



RNNs are designed to handle sequences of data

- Time-series - Tesla stock prices Hands-on
- Text
- speech

- Video Frames

RNNs are designed to process one element at a time while maintaining a memory of previous inputs.

② Memory of previous inputs (Hidden State Memory)

- RNNs have an internal hidden state that is updated at each time step and this allows them to capture the context.

↓ (understand the intent)

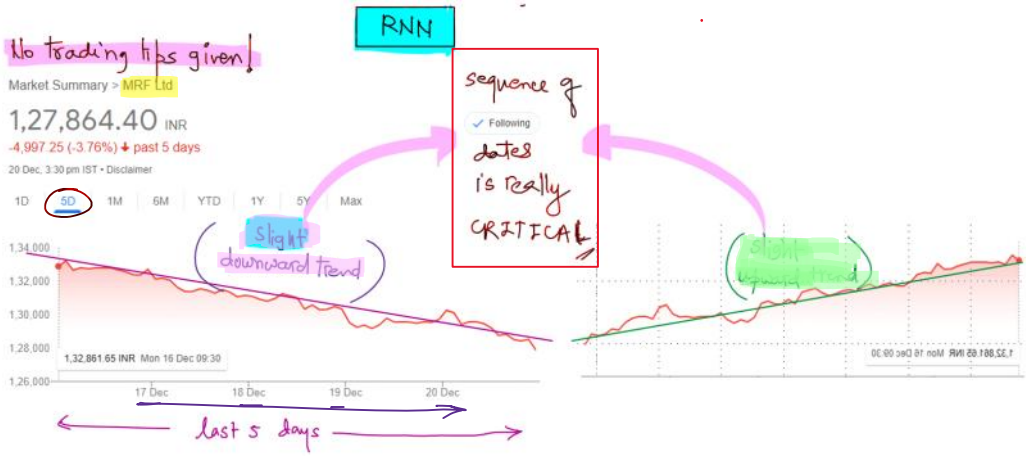
RNNs can interpret new data in the context of prior information which is essential for understanding the meaning and the flow in natural language tasks.

③ Dynamic Adaption RNNs continuously updates internal states allowing them to adjust to evolve patterns within the sequences.

Applications of RNN

Stock Prediction

No trading tips given!

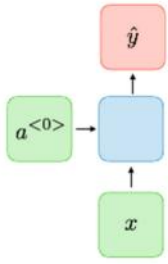
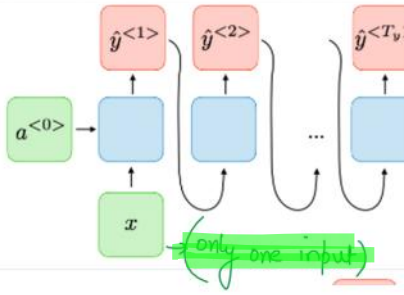


- RNN needs proper time-series data or sequential data in general (meaningful) as the right analysis can't be done without the right sequence.

There are certain applications where RNN is commonly used in :

- Time Series Forecasting ✓ stock prediction using Tesla share Prices "
- Speech Recognition
- Machine (Google) Translation
- language Modelling

Types of RNN

Type of RNN	Illustration	Example
1 One-to-one $T_x = T_y = 1$ (Vanilla mode)	 → SISO : single input, single output	Recurrent Traditional neural network
2 One-to-many $T_x = 1, T_y > 1$ (sequence output)	 → SIMO	Music generation Once upon → 5 words input not "C" → a sequence

Musenet & Magneta (Google) are models to create larger composition starting with a single note.

2nd example:

one input : one random image

↓

output : sequence of text .

Taj Mahal: Facts and History

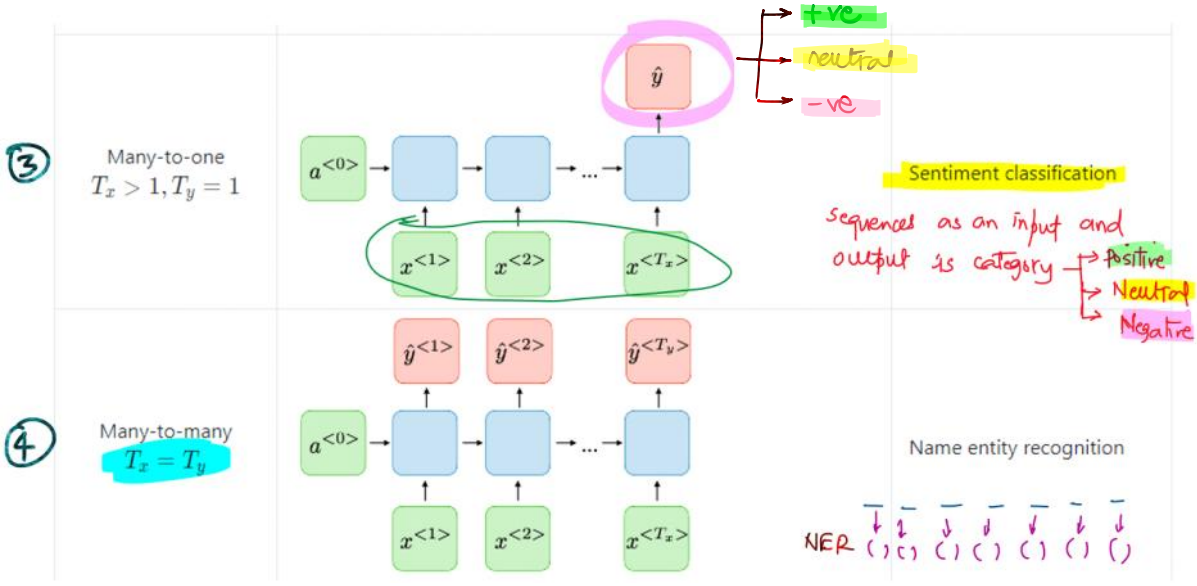
Analysis

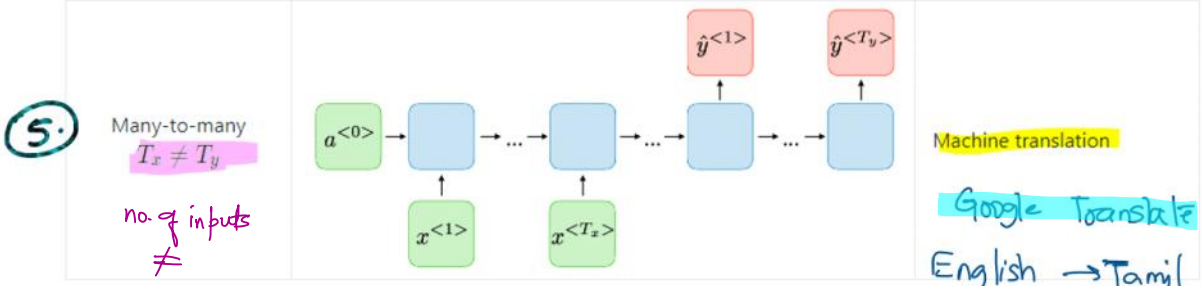
That image is of the **Taj Mahal** in **Agra, India**.

It is one of the world's most famous monuments and a UNESCO World Heritage Site.

Key Facts About the Taj Mahal

- **Builder:** Mughal Emperor **Shah Jahan**.
- **Purpose:** Built as a mausoleum for his favorite wife, **Mumtaz Mahal**.
- **Construction Period:** Construction began around **1632** and was completed around **1648** (the surrounding buildings and garden took five more years).
- **Architecture:** It is a superb example of **Mughal architecture**, combining elements from Islamic, Persian, Ottoman Turkish, and Indian styles.





no. of outputs

Google Translation

Limitations of RNN

Pro-tip

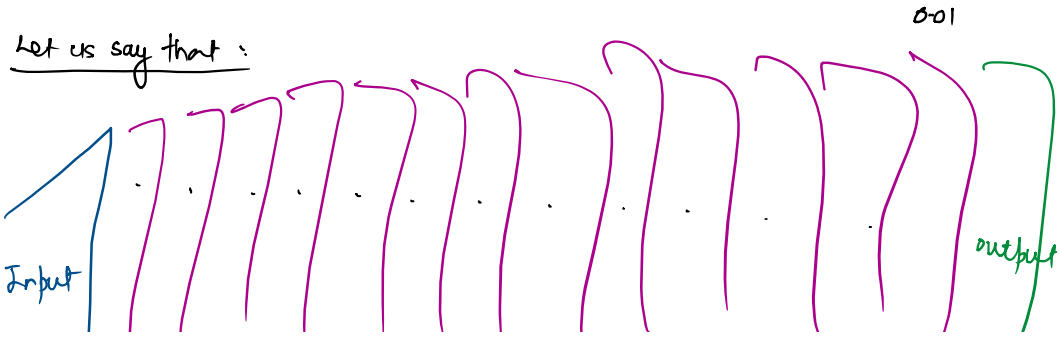
Vanishing / Exploding Gradients

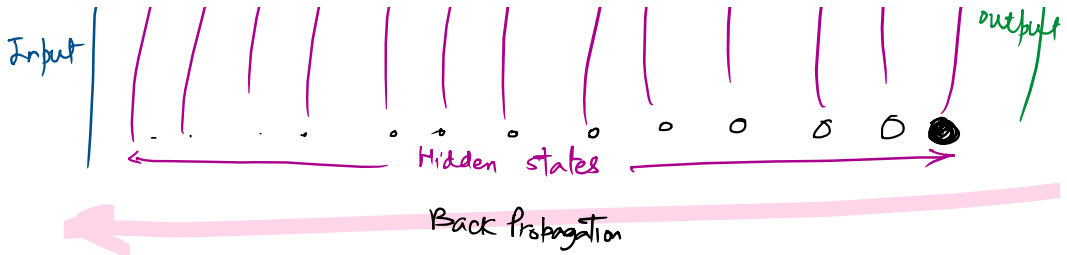
Vanishing Gradients

RNNs are not able to remember long sequences as it gets into the problem of vanishing gradients

Vanishing gradients problem happens during backpropagation in deep (20+ layers) neural networks

Let us say that :





Back Propagation

Illustration $0.01 * 0.01 * 0.01 * 0.01 * \dots$ 12 times $= (0.01)^{12} = \frac{1}{100} \times \frac{1}{100} \times \dots$ 12 times $\triangleq 0$

(Vanishing gradients)

Gradients (partial derivatives of the loss w.r.t. weight)

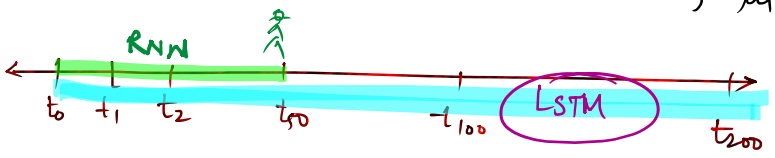
shrink exponentially as they propagate backwards the layers

As a result of this,

- early layer(s) stop learning
- and the n/w fails to converge or trains extremely slowly

Vanishing Gradients

Consequence: RNN n/w forgets long-term dependencies → things far back in time



Exploding Gradients

When the gradients become excessively large, growing exponentially as they propagate backward through the n/w.

This causes large updates to weights leading to numerical instability or divergence of the model.

12 * 3.8 *2.5*6.9*5.7=**4,483.62**

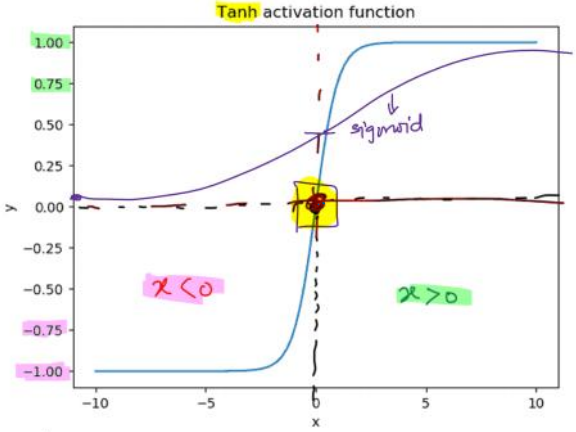
Gradient # 1.8

1.8*1.8*1.8*1.8*1.8 = **18.8957**

Note: Exploding gradient problem is the opposite of vanishing gradient problem.

Activation Function: tanh

tanh : tangent hyperbolic



It's a non-linear activation function to introduce non-linearity, allowing neural n/w models to learn complex patterns by mapping values between $-1 \leq f(z) \leq 1$

$$\tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

$-\infty < x < \infty$

$-1 < \tanh(x) < 1$

Note: tanh(x) is zero-centred and outputs values b/w -1 and 1 which makes the activation values centered around origin unlike the other activation functions.

Given that it's the balanced activation function, it leads to faster convergence in the gradient descent

Tx ≠ Ty

Many - to - many RNN Model Examples:

↳ different input/output lengths

English: I love dogs

spanish: Me encantan los perros