Types of RNN

Why are there different types of RNNs?

Because **different tasks need different ways of handling sequences**.

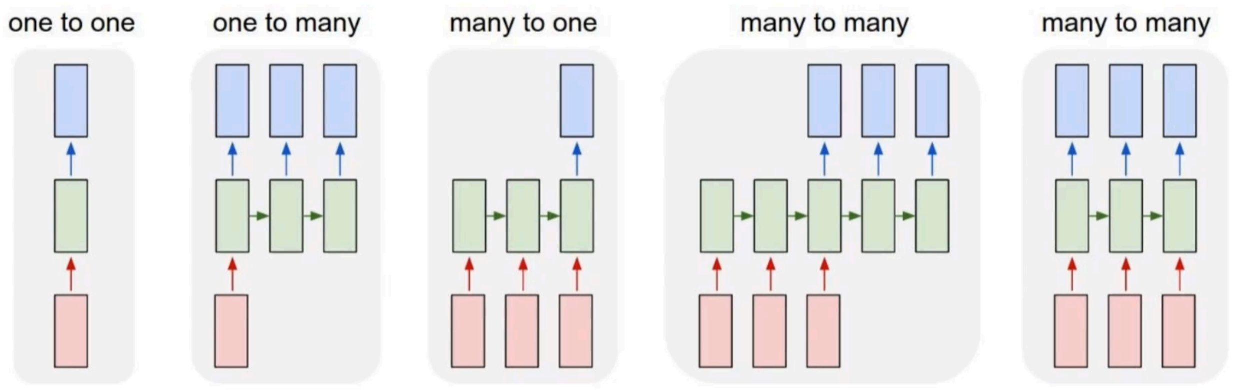
RNNs are like chains that process inputs step by step. But depending on:

- What kind of input we give (single item or sequence),
- What kind of output we want (single item or sequence),

we build **different RNN architectures** to suit the task.

🔷 Overview of RNN Types (with real-world examples)

Type	Input	Output	Example Task
1. One-to-One	Single	Single	Image classification
2. One-to-Many	Single	Sequence	Image captioning
3. Many-to-One	Sequence	Single	Sentiment analysis, fraud detection
4. Many-to-Many (same length)	Sequence	Sequence	POS tagging, music generation
5. Many-to-Many (different length)	Sequence	Sequence	Machine translation (e.g., English → Hindi)



🔶 1. One-to-One (Basic Neural Network)

This is NOT an RNN, but a standard neural network like MLP or CNN.

- Input: Single object (e.g., image)
- Output: Single prediction (e.g., cat or dog)

> Example: Classifying an image as "cat" or "dog".

- **Problem**: Struggles with long sequences due to vanishing gradients.

📌 Layer in Keras: `SimpleRNN`

Used in regular, non-sequential tasks.

🔶 2. One-to-Many

- Input: One object
- Output: A sequence of items

> Example: Image captioning, Music generation

- You give one image (input)
- The network generates a sentence word by word (output)

This means:

- One fixed input is passed through an encoder (like a CNN),
- Then RNN generates output over time, step by step.

🔶 3. Many-to-One

- Input: A **sequence**

    - Sentence, character, time series data
- Output: One prediction
    - Number(1,0)

> ### Example: Sentiment analysis, rating prediction

- You give a whole sentence (sequence of words),
- The network gives one output: positive/negative sentiment

This is **very common** in classification tasks involving sequences.

Behind the scenes:

- RNN reads each word one by one,
- Builds up a memory (hidden state),
- At the end, it uses the final memory to make a single prediction.

🔶 4. Many-to-Many (same length)

- Input: Sequence
- Output: Sequence (of the same length)

> ### Example: Part-of-speech (POS) tagging

- You input a sentence: "The cat sleeps"
- You get a tag per word: "DET NOUN VERB"

Another example:

- Music generation where for each input note, the RNN generates a corresponding note.

The RNN here:

- Produces one output for every time step

- Keeps passing memory along

🔶 5. Many-to-Many (different length)

- Input: Sequence
- Output: Sequence (of **different length**)

> Example: **Machine translation**

- Input: "How are you?"
- Output: "तुम कैसे हो?"

Input is 3 words, output is 4.

This is more advanced — we use **Encoder-Decoder architecture**:

- **Encoder**: reads entire input and stores the meaning in memory
- **Decoder**: uses that memory to produce the output, one step at a time

Used in:

- Translation
- Question answering
- Text summarization