RNN Architecture

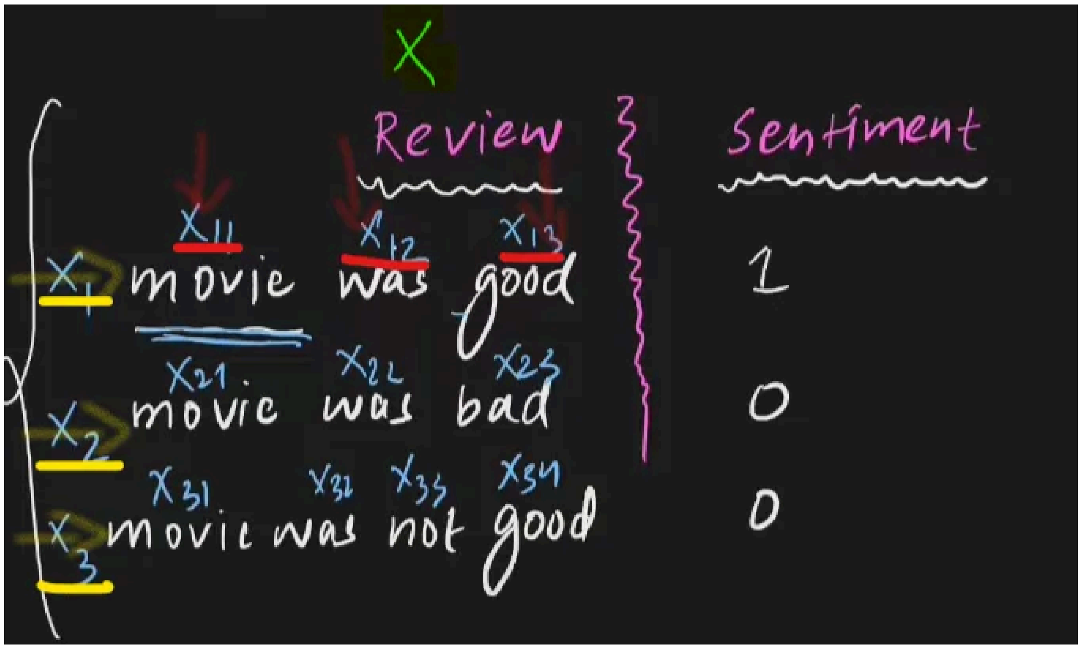
Input data

- When you input data in RNN, it'n in the form → `(timesteps, input_features)`
- In Keras → SimpleRNN → `(batch_size, timesteps, input_features)`
    - **Example:**

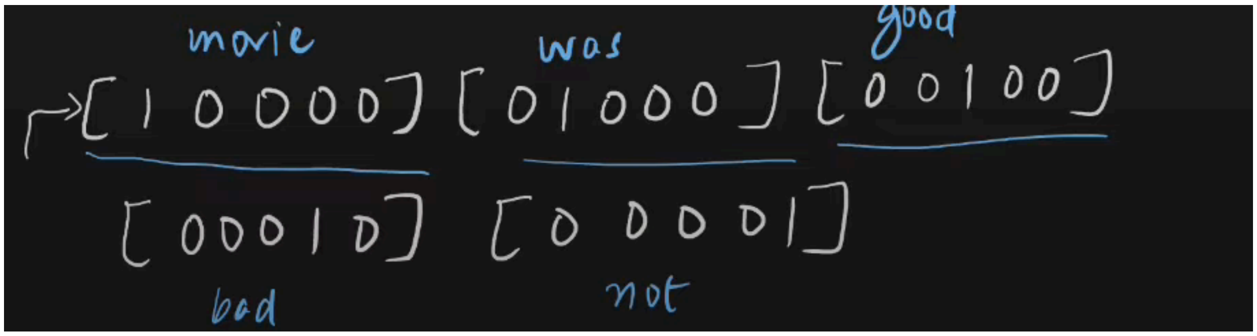
      Let's say you have a dataset of 100 training sequences, and each sequence has 20 time steps (e.g., 20 days of data), and at each time step, you have 3 features (e.g., temperature, humidity, pressure).

      The input data would then have a shape of:

      (100, 20, 3)



X_{11}, X_{12}, X_{13}**, etc. are vectors like:** 👇



🔷 What is the architecture of an RNN?

Think of an RNN as a **chain of repeating cells**, where each cell:

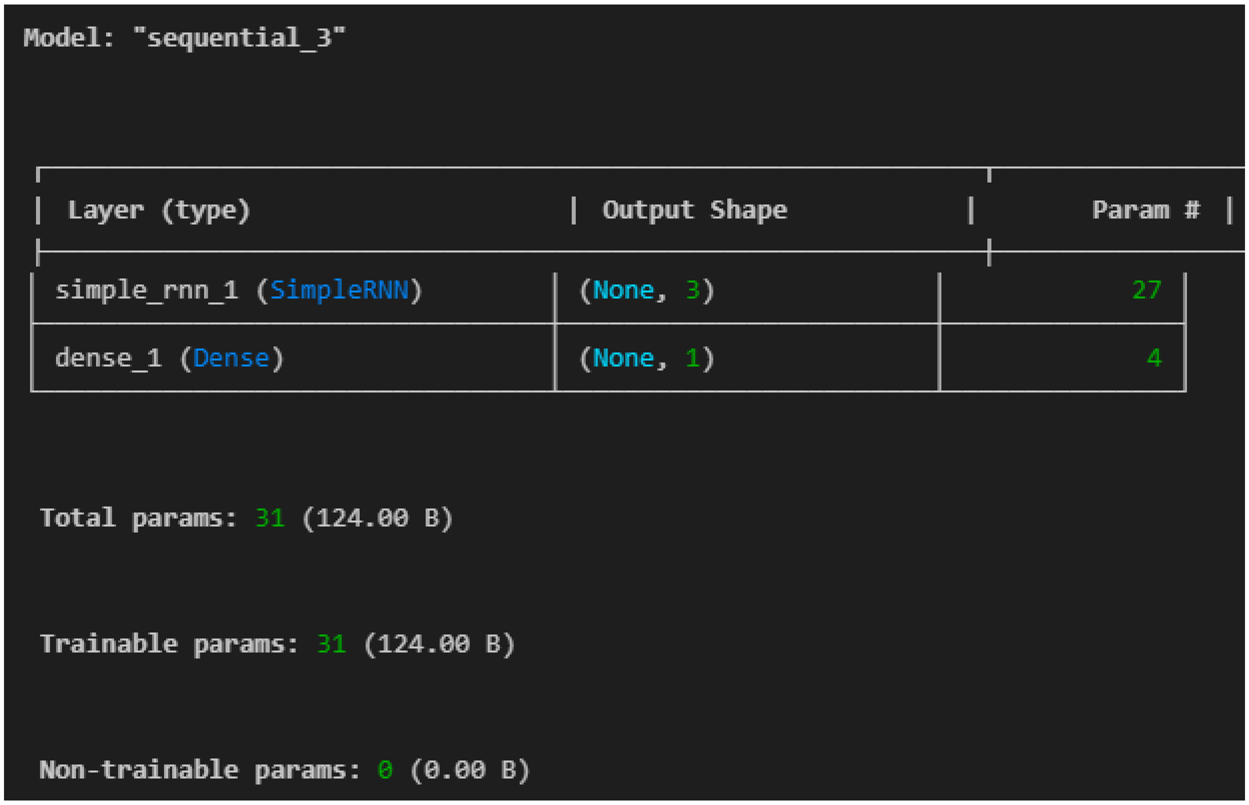
1. Takes in one element of the input sequence (like one word),
2. Receives memory from the previous cell,
3. **Updates that memory**,
4. Produces an **output**,
5. Passes the new memory to the next cell.

Even though every cell processes a different time step, **all the cells are identical** and **share the same weights**. That's what keeps the model small and efficient.

Python Code:

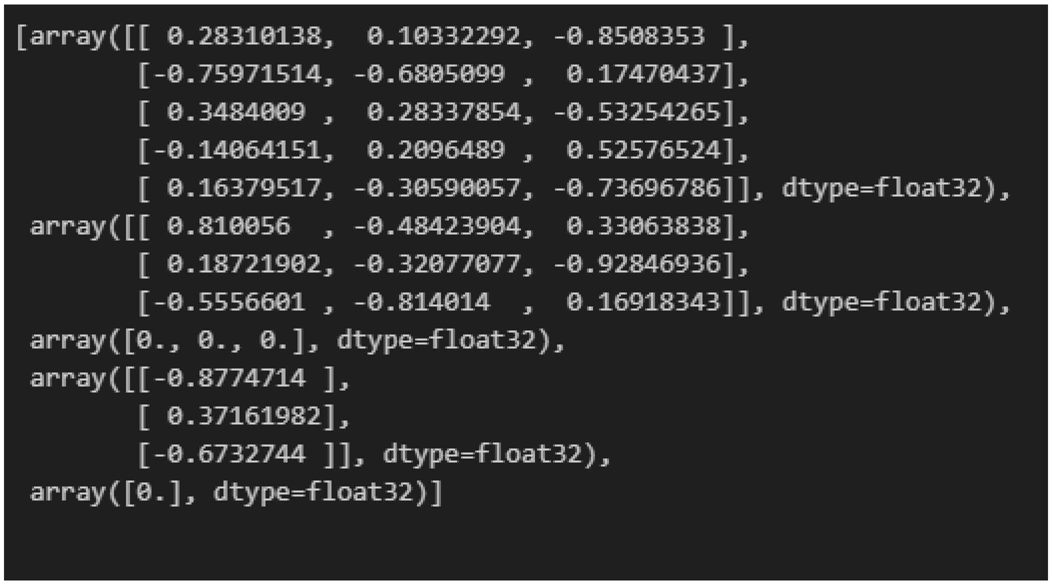
```
from keras.layers import SimpleRNN,Dense
from keras import Sequential
```

```
model = Sequential()
model.add(SimpleRNN(3, input_shape=(4, 5)))  # Corrected input shape
model.add(Dense(1, activation='sigmoid'))

model.summary()
```



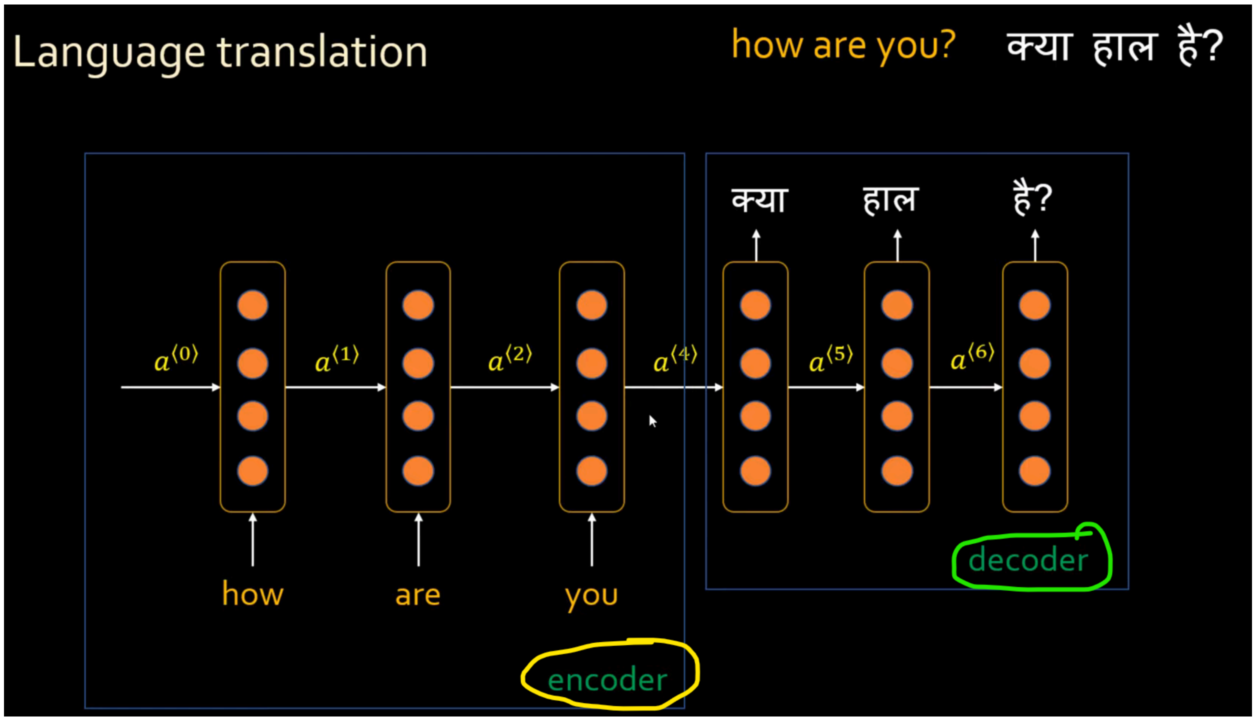
```
Model: "sequential_3"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━┓
┃ Layer (type)                    ┃ Output Shape           ┃       Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━┩
│ simple_rnn_1 (SimpleRNN)        │ (None, 3)              │            27 │
├─────────────────────────────────┼────────────────────────┼───────────────┤
│ dense_1 (Dense)                 │ (None, 1)              │             4 │
└─────────────────────────────────┴────────────────────────┴───────────────┘

 Total params: 31 (124.00 B)

 Trainable params: 31 (124.00 B)

 Non-trainable params: 0 (0.00 B)
```

```
model.get_weights()
```



```
[array([[ 0.28310138,  0.10332292, -0.8508353 ],
        [-0.75971514, -0.6805099 ,  0.17470437],
        [ 0.3484009 ,  0.28337854, -0.53254265],
        [-0.14064151,  0.2096489 ,  0.52576524],
        [ 0.16379517, -0.30590057, -0.73696786]], dtype=float32),
 array([[ 0.810056  , -0.48423904,  0.33063838],
        [ 0.18721902, -0.32077077, -0.92846936],
        [-0.5556601 , -0.814014  ,  0.16918343]], dtype=float32),
 array([0., 0., 0.], dtype=float32),
 array([[-0.8774714 ],
        [ 0.37161982],
        [-0.6732744 ]], dtype=float32),
 array([0.], dtype=float32)]
```

```
Input 1 → Hidden state → Output 1
Input 2 + Hidden state from step 1 → Output 2
Input 3 + Hidden state from step 2 → Output 3
...
```


Language translation
how are you?
क्या हाल है?
क्या
हाल
है?
$a^{\langle 0 \rangle}$
$a^{\langle 1 \rangle}$
$a^{\langle 2 \rangle}$
$a^{\langle 4 \rangle}$
$a^{\langle 5 \rangle}$
$a^{\langle 6 \rangle}$
how
are
you
decoder
encoder