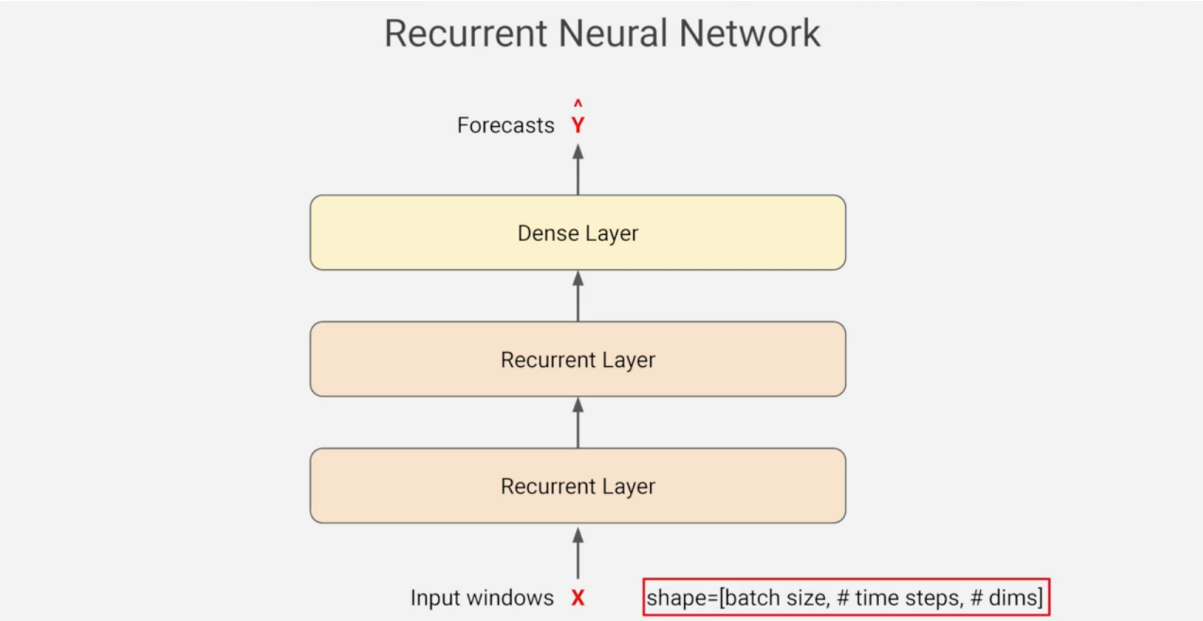
Shape of the inputs to the RNN

While the following diagram shows several RNN layers, in reality it is a single RNN layer that's being used multiple times.



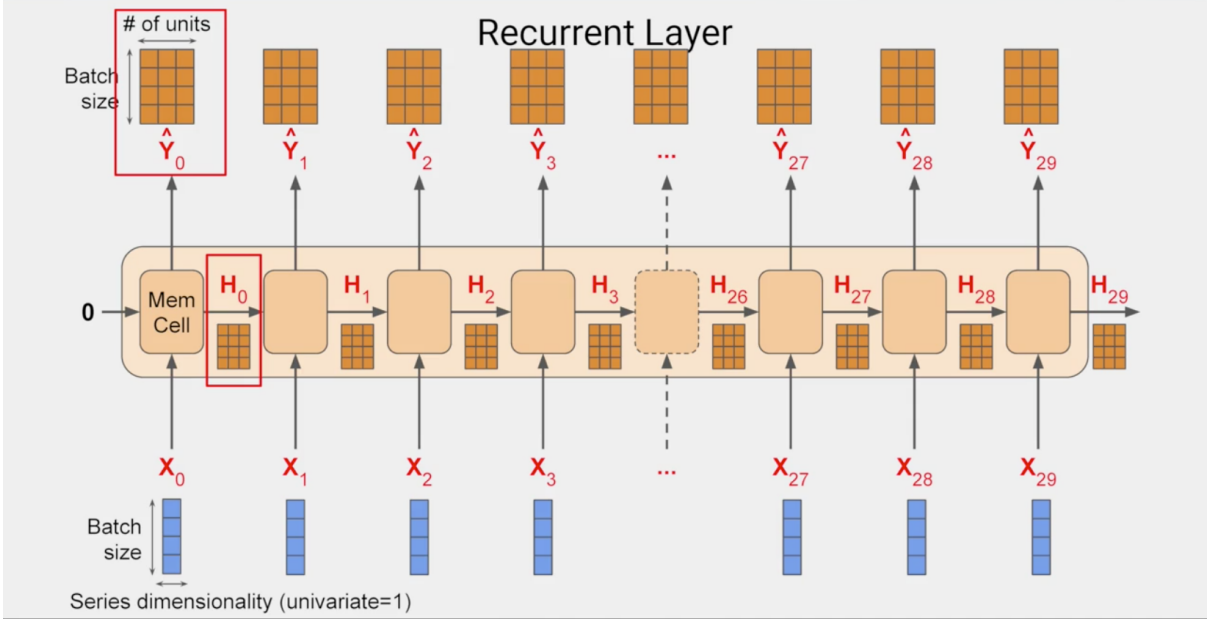
It's important to understand the dimensions of the inputs and outputs when using an RNN with time series data. Consider the following example:

```
import numpy as np
# Batch size = 2, sequence length = 3, number features = 1, shape=(2, 3, 1)
values231 = np.array([
    [[1], [2], [3]],
    [[2], [3], [4]]
])

# Batch size = 3, sequence length = 5, number features = 2, shape=(3, 5, 2)
values352 = np.array([
    [[1, 4], [2, 5], [3, 6], [4, 7], [5, 8]],
    [[2, 5], [3, 6], [4, 7], [5, 8], [6, 9]],
    [[3, 6], [4, 7], [5, 8], [6, 9], [7, 10]]
])
```

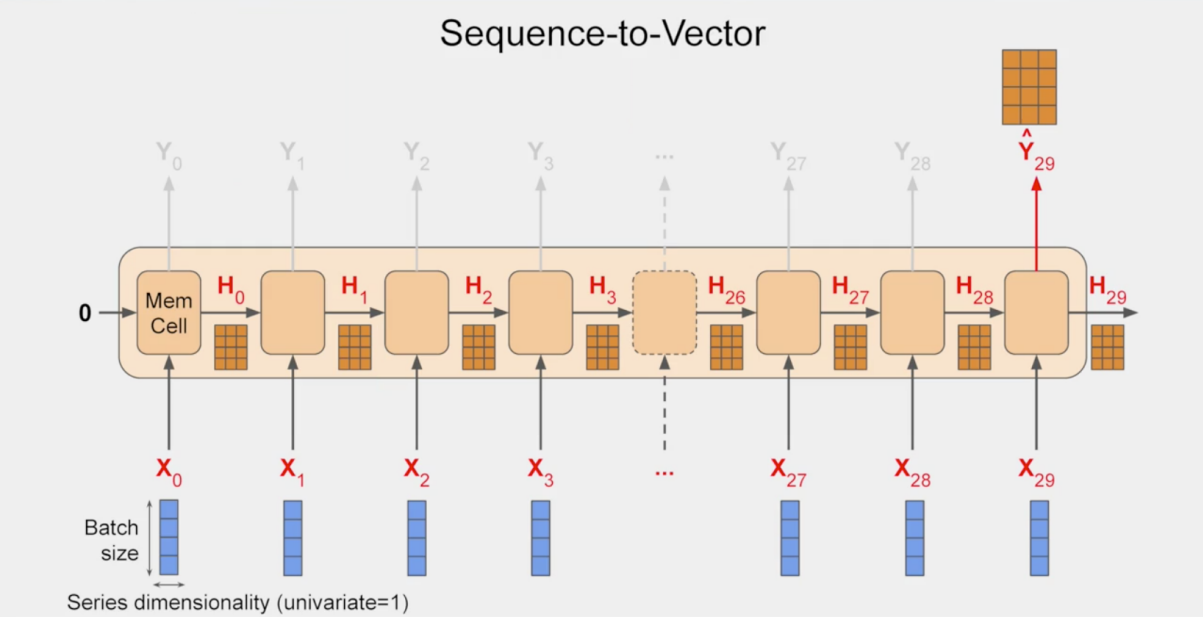
- An RNN will feed each time step into the next cell. Taking the second example from above, the first time step takes [1, 4] as input, second step [2, 5] etc.
- Even a sequence of single numbers needs to have the shape of (batch_size, seq_length, num_features).
- If you have a sample with a sequence of single numbers (say [[1, 2, 3], [2, 3, 4]]), you can do np.reshape(2, 3, 1) to reshape from (2, 3) into a sequence dataset.

The input of each time step is of dimension batch size x variate (univariate so variate = 1 in this case) and the output of each step is batch size x no. of features. Think of it as having a separate neuron to extract each feature, so no. of features = no. of units. In this case, we're using a series with window size = 30 and batch size = 4, so the input layer (yes, the entire layer, not just a single step) is of dimension 30 x 4 x 1 since the total number of time steps = window size = 30.



In a simple RNN, the cell state output H is just a copy of the output matrix Y. So in this case, Y0 = H0, Y1 = H1, and so on. At each time step, the memory

cell gets the current input as well as the previous output.



If you only want a single output vector for the batch and not the entire output layer, you leave the return_sequences parameter as False in Keras. This is the default state. If you want the entire sequence, as you would when stacking multiple RNN layers (i.e one RNN layer feeds another), you set return_sequences = True.