1. Can you think of a few applications for a sequence-to-sequence RNN? What about a sequence-to-vector RNN, and a vector-to-sequence RNN?

Sequence-to-Sequence RNN: These are commonly used in machine translation, text summarization, speech recognition, and any task where the input and output are both sequences of varying lengths.

Sequence-to-Vector RNN: This is useful for tasks like sentiment analysis, where the input is a sequence (e.g., a sentence) and the output is a single vector (e.g., sentiment score).

Vector-to-Sequence RNN: This can be used for tasks like image captioning, where the input is a single vector (e.g., an image representation) and the output is a sequence (e.g., a caption for the image).

1. How many dimensions must the inputs of an RNN layer have? What does each dimension represent? What about its outputs?

**Dimensions in RNNs:**

* The input to an RNN layer has three dimensions: (batch\_size, time\_steps, input\_features). Here, batch\_size is the number of sequences in each batch, time\_steps is the length of each sequence, and input\_features is the number of features at each time step.
* The output of an RNN layer also has three dimensions: (batch\_size, time\_steps, output\_features). Here, output\_features represent the hidden states or features produced by the RNN.

1. If you want to build a deep sequence-to-sequence RNN, which RNN layers should have return\_sequences=True? What about a sequence-to-vector RNN?

**Setting return\_sequences in Deep RNNs:**

* In a deep sequence-to-sequence RNN, you typically set return\_sequences=True for all intermediate RNN layers to pass the sequences of hidden states to the next layer.
* In a sequence-to-vector RNN, you set return\_sequences=True for intermediate layers if you want to capture sequential information before pooling or selecting the final output.

1. Suppose you have a daily univariate time series, and you want to forecast the next seven days. Which RNN architecture should you use?

For forecasting the next seven days of a daily univariate time series, you would typically use a sequence-to-sequence RNN where both input and output sequences have a length of seven.

1. What are the main difficulties when training RNNs? How can you handle them?

Gradient Vanishing/Exploding: RNNs can have difficulty learning long-range dependencies. Techniques like LSTM and GRU cells help mitigate this.

Overfitting: RNNs, especially with many parameters, can easily overfit. Regularization and dropout can help.

Training Time: Training deep RNNs can be time-consuming.

1. Can you sketch the LSTM cell’s architecture?

**LSTM Cell Architecture:** An LSTM (Long Short-Term Memory) cell consists of:

* Input Gate: Controls input information.
* Forget Gate: Controls what information should be discarded.
* Output Gate: Produces the cell's output.
* Cell State: Internal memory.

1. Why would you want to use 1D convolutional layers in an RNN?

1D convolutional layers can capture local patterns in sequential data. They are useful when you suspect that local patterns are important for the task, like in text classification.

1. Which neural network architecture could you use to classify videos?

For video classification, you can use a 3D CNN (Convolutional Neural Network) combined with an RNN or Transformer to model both spatial and temporal features.

1. Train a classification model for the SketchRNN dataset, available in TensorFlow Datasets.