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: Used for tasks where the input and output are both sequential data of variable lengths. Applications include machine translation, speech recognition, and text summarization.**

**Sequence-to-Vector RNN: Suitable for tasks where the input is a sequence, but the output is a fixed-size vector. Examples include sentiment analysis of text and extracting features from sequences.**

**Vector-to-Sequence RNN: Used when the input is a fixed-size vector, but the output needs to be a sequence. Applications include generating natural language descriptions from images and video captioning.**

2. How many dimensions must the inputs of an RNN layer have? What does each dimension

represent? What about its outputs?

**RNN inputs typically have three dimensions: (batch\_size, timesteps, input\_features).**

**batch\_size: Number of sequences in a batch.**

**timesteps: Number of time steps in each sequence.**

**input\_features: Number of features at each time step.**

**RNN outputs depend on the task:**

**In sequence-to-sequence tasks, outputs have the same dimensions as inputs, but they represent the model's predictions or hidden states.**

**In sequence-to-vector tasks, outputs have dimensions (batch\_size, output\_features) and represent the final output of the sequence.**

**In vector-to-sequence tasks, outputs have dimensions (batch\_size, timesteps, output\_features), producing sequences as output.**

3. 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?

**In a deep sequence-to-sequence RNN, typically, all RNN layers should have return\_sequences=True to propagate sequential information through the layers.**

**In a sequence-to-vector RNN, only the last RNN layer typically has return\_sequences=False to produce a fixed-size vector as the output.**

4. 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 in a daily univariate time series, you can use a sequence-to-sequence RNN architecture. This allows you to input a sequence of historical data and generate a sequence of future predictions.**

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

**Vanishing and Exploding Gradients: RNNs are prone to gradients that either become too small (vanishing) or too large (exploding) during training. Solutions include using gradient clipping, using specialized RNN cells like LSTMs or GRUs, and employing skip connections.**

**Long-Term Dependencies: Traditional RNNs struggle to capture long-term dependencies in sequences. LSTMs and GRUs are designed to mitigate this issue.**

**Training Time: RNNs can be slow to train, especially on long sequences. Techniques like mini-batch training, using GPU acceleration, and optimizing the training pipeline help speed up training.**

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

**An LSTM (Long Short-Term Memory) cell consists of three main gates and a memory cell:**

**Forget Gate: Controls what information to discard from the cell state.**

**Input Gate: Modifies the cell state based on new input information.**

**Output Gate: Determines the output of the cell based on the current cell state.**

**Memory Cell: Stores and passes information across time steps.**

**Each gate and the memory cell are governed by sigmoid and tanh activation functions.**

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

**1D convolutional layers can be used in an RNN to capture local patterns and features within sequential data. They can help identify short-term dependencies and patterns, enhancing the RNN's ability to learn relevant information from the data.**

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

**To classify videos, you can use architectures like 3D Convolutional Networks (3D CNNs) or Convolutional Recurrent Networks (CRNs). These models can capture both spatial and temporal features in video data.**

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

**Training a classification model for the SketchRNN dataset involves loading the dataset, preprocessing the data, defining a neural network architecture suitable for sketch classification, training the model, and evaluating its performance. The specific code and steps may vary depending on the framework used, such as TensorFlow or PyTorch.**