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?**

**Ans:** Sequence-to-sequence RNN: Language translation, chatbot responses, and text summarization.

Sequence-to-vector RNN: Sentiment analysis, document classification, and video tagging.

Vector-to-sequence RNN: Image captioning, text generation from a given context, and video generation from a fixed-size latent vector.

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

**Ans:** RNN inputs should have three dimensions: batch size, time steps, and input features. The batch size represents the number of sequences processed simultaneously. The time steps indicate the sequence length, and the input features represent the dimensionality of each time step.

RNN outputs have the same dimensions as inputs, but they represent the hidden states or the outputs at each time step.

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?**

**Ans:** In a deep sequence-to-sequence RNN, all the intermediate RNN layers should have return\_sequences=True to ensure that the output of each time step is passed to the subsequent layers.

In a sequence-to-vector RNN, all the layers except the last one should have return\_sequences=True to maintain sequence information through the network.

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

**Ans:** For forecasting the next seven days in a daily univariate time series, an RNN architecture with a sequence-to-sequence setup and a window size of seven time steps as input would be appropriate.

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

**Ans:** Vanishing or exploding gradients, long-term dependencies, and overfitting are common issues in training RNNs. They can be handled using techniques such as gradient clipping, using gating mechanisms like LSTMs or GRUs, and implementing regularization methods such as dropout or weight regularization.

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

**Ans:** An LSTM cell typically consists of a cell state, an input gate, a forget gate, and an output gate. The cell state retains the memory, and the gates regulate the flow of information by controlling the input, forget, and output of the cell.

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

**Ans:** 1D convolutional layers in RNNs are used to capture local patterns in sequential data efficiently. They help in extracting features from sequential data, allowing the RNN to learn hierarchical representations and long-range dependencies more effectively.

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

**Ans:** To classify videos, a suitable architecture could be a combination of 3D convolutional layers for spatial and temporal feature extraction, followed by recurrent layers or fully connected layers for temporal modeling and classification.

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

**Ans:** To train a classification model for the SketchRNN dataset, the typical process would involve data preprocessing, model building with appropriate layers (such as LSTM or GRU), compiling the model with suitable loss and optimizer, and training the model using the dataset. This process would also include evaluation and fine-tuning for optimal performance.