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?**

* In Sequence to Sequence Learning, RNN is trained to map an input sequence to an output sequence which is not necessarily of the same length. Applications are **speech recognition, machine translation, image captioning and question answering**.
* Feed sequence of inputs and take only last output, so it is called **Sequence-to-Vector**.  For example, take an IMDB review of a movie and ignore all outputs in the middle to predict the movie horror or drama.
* Feed the same input vector multiple times at each time and produce an output, so it is called **Vector-to-Sequence.**This is useful in Convolutional Neural Network(CNN) where we need to input an image and expect to predict what is that image.

1. **Why do people use encoder–decoder RNNs rather than plain sequence-to-sequence RNNs for automatic translation?**

RNN maintains internal memory, due to this they are very efficient for machine learning problems that involve sequential data. RNNs are also used in time series predictions as well. The main advantage of using RNNs instead of standard neural networks is that the features are not shared in standard neural network.

1. **How could you combine a convolutional neural network with an RNN to classify videos?**

Each video is converted into sequential images and passed onto the CNN to extract spatial features. The outputs are then passed into a recurrent sequence learning model (i.e. LSTM) to identify temporal features within the image sequence.

1. **What are the advantages of building an RNN using dynamic\_rnn() rather than static\_rnn()?**

Building an RNN usingdynamic\_rnn() rather than static\_rnn() offers several advantages:

It is based on a while\_loop()operation that is able to swap the GPU’s memory to the CPU’s memory during backpropagation, avoiding out-of-memory errors. It is arguably easier to use, as it can directly take a single tensor as input and output(covering all time steps), rather than a list of tensors (one per time step). No need to stack,unstack, or transpose. It generates a smaller graph, easier to visualize in TensorBoard.

1. **How can you deal with variable-length input sequences? What about variable-length output sequences?**

* The first and simplest way of handling variable length input is to **set a special mask value in the dataset, and pad out the length of each input to the standard length with this mask value set for all additional entries created**. Then, create a Masking layer in the model, placed ahead of all downstream layers.

1. **What is a common way to distribute training and execution of a deep RNN across multiple GPUs?**

**Strategy** is a TensorFlow API to distribute training across multiple GPUs, multiple machines, or TPUs. Using this API, you can distribute your existing models and training code with minimal code changes.