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:

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.

**The Encoder RNN** reads the input sequence and generates the **ﬁxed-size context vector** which represents a semantic summary of the input sequence.

The **fixed-size context vector**is given as input to the decoder RNN.

The **fixed-size context** can be provided as the **initial state of the Decoder RNN**, or it can be connected to the **hidden units at each time step**. These two ways can also be combined.

**The number of time steps** in the Encoder and Decoder need not to be equal.

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

ANS:

Before we get down to business, an important thing to note is that the RNN input needs to have 3 dimensions. Typically it would be batch size, the number of steps and number of features.

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:

The RNN model takes a single vector as input and produces a sequence as output. An example of these models can be image to sentence model, which takes an image(consider it as a vector) and then produces a sentence to describe that image.

There are three built-in RNN layers in Keras: keras.

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:

The batter hits the ball. You immediately start running, anticipating the ballâs trajectory. You track it and adapt your movements, and finally catch it (under a thunder of applause). Predicting the future is what you do all the time, whether you are finishing a friendâs sentence or anticipating the smell of coffee at breakfast. In this chapter, we are going to discuss recurrent neural networks (RNN), a class of nets that can predict the future (well, up to a point, of course). They can analyze time series data such as stock prices, and tell you when to buy or sell. In autonomous driving systems, they can anticipate car trajectories and help avoid accidents. More generally, they can work on sequences of arbitrary lengths, rather than on fixed-sized inputs like all the nets we have discussed so far. For example, they can take sentences, documents, or audio samples as input, making them extremely useful for natural language processing (NLP) systems such as automatic translation, speech-to-text, or sentiment analysis (e.g., reading movie reviews and extracting the raterâs feeling about the movie).

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

ANS:

RNNs suffer from the problem of vanishing gradients. The gradients carry information used in the RNN, and when the gradient becomes too small, the parameter updates become insignificant. This makes the learning of long data sequences difficult.

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

ANS:

LSTMs deal with both Long Term Memory (LTM) and Short Term Memory (STM) and for making the calculations simple and effective it uses the concept of gates. Forget Gate: LTM goes to forget gate and it forgets information that is not useful.

LSTM is widely used in Sequence to Sequence (Seq2Seq) models, a type of neural network architecture used for many sequence-based tasks such as machine translation, speech recognition, and text summarization. LSTM is widely used in Sequence to Sequence (Seq2Seq) models, a type of neural network architecture used for many sequence-based tasks such as machine translation, speech recognition, and text summarization.

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

ANS:

The data considered here are one dimensional time varying signals and hence the 1-D convolutional neural networks are used to train, test and to analyze the learned weights.

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

ANS:

Stay organized with collections Save and categorize content based on your preferences. This tutorial demonstrates training a 3D convolutional neural network (CNN) for video classification using the UCF101 action recognition dataset.

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

ANS:

In order to use a Dataset we need three steps:

1. Importing Data. Create a Dataset instance from some data.
2. Create an Iterator. By using the created dataset to make an Iterator instance to iterate through the dataset.
3. Consuming Data. By using the created iterator we can get the elements from the dataset to feed the model.

Create your model

1. Import the Fashion MNIST dataset.
2. Train and evaluate your model.
3. Add TensorFlow Serving distribution URI as a package source:
4. Install TensorFlow Serving.
5. Start running TensorFlow Serving.
6. Make REST requests.