1) can you think of a few application for a sequence to Sequence RNN ? What about a sequence to vector RNN and a vector to sequence RNN ?

Ans : Applications : sequence to sequence RNN

speech recognition, machine translation, image captioning and question answering.

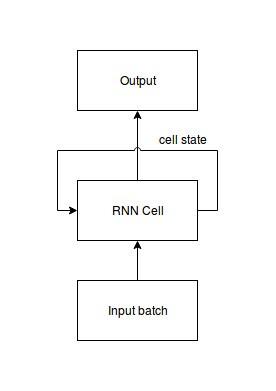
Sequence to vector :Sequence-to-Vector network called an encoder and Vector-to-Sequence network called a decoder.Google translation uses the Seq2Seq model, the model uses an encoder and decoder for translating words from one language to other languages.

Most of the Chatbot applications use the Sequence to Sequence model.

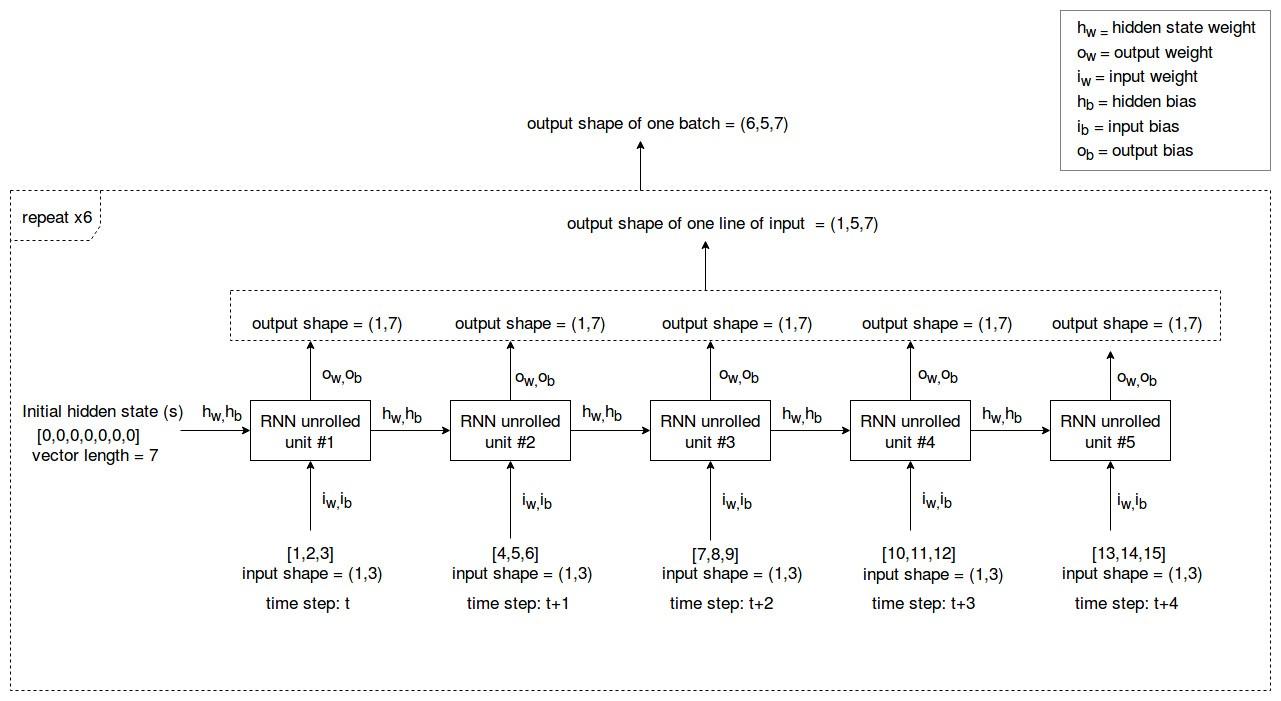
2) How many dimensions must the inputs of an RNN layer have ?what does each dimension Represent ?what about its output ?

Ans : There are 3 inputs dimension of an RNN layer.Typically it would be batch size, the number of steps and number of features. The number of steps depicts the number of time steps/segments you will be feeding in one line of input of a batch of data that will be fed into the RNN.There are many questions on Stackoverflow that inquire if “RNN cell” refers to one single cell or the whole layer. Well, it’s more like the whole layer. The reason for this is that the connections in RNNs are recurrent, thus following a “feeding to itself” approach

This is due to the hidden state in the RNN. It retains information from one time step to another flowing through the unrolled RNN units. Each unrolled RNN unit has a hidden state. The current time steps hidden state is calculated using information of the previous time step’s hidden state and the current input. This process helps to retain information on what the model saw in the previous time step when processing the current time steps information. Also, something to note is that all the connections in RNN have weights and biases. The biases can be optional in some architectures. This process will be explained further in later parts of the article.



But when the RNN starts to process the data it will unroll and produce outputs as shown below:



Processing a batch:

When you feed a batch of data into the RNN cell it starts the processing from the 1st line of input. Likewise, the RNN cell will sequentially process all the input lines in the batch of data that was fed and give one output at the end which includes all the outputs of all the input lines.

Output = (StOw)+Ob.

3) if you want to build a deep Sequence to sequence RNN which RNN layers should have return-sequence-true ? What about a sequence to vector RNN ?

Ans : Return sequences refer to return the hidden state a<t>. By default, the return\_sequences is set to False in Keras RNN layers, and this means the RNN layer will only return the last hidden state output a<T>. The last hidden state output captures an abstract representation of the input sequence

Sequence-to-Vector network called an encoder and Vector-to-Sequence network called a decoder.Google translation uses the Seq2Seq model, the model uses an encoder and decoder for translating words from one language to other languages.

Most of the Chatbot applications use the Sequence to Sequence model.

4) suppose you have a daily univariate time series and you want to forecast the next seven days which RNN Architecture should you use ?

Ans :LSTMs can be used to model univariate time series forecasting problems.

These are problems comprised of a single series of observations and a model is required to learn from the series of past observations to predict the next value in the sequence.

We will demonstrate a number of variations of the LSTM model for univariate time series forecasting.

This section is divided into six parts; they are:

Data Preparation

Vanilla LSTM

Stacked LSTM

Bidirectional LSTM

CNN LSTM

ConvLSTM

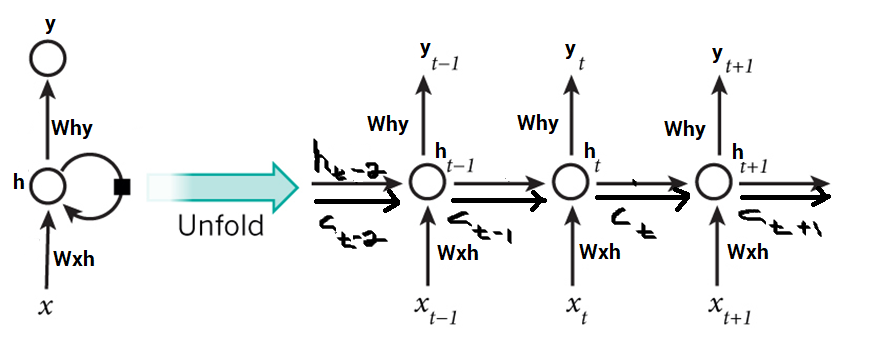
Each of these models are demonstrated for one-step univariate time series forecasting, but can easily be adapted and used as the input part of a model for other types of time series forecasting problems.

5) what are the main difficulties when training RNN's ? how can you handle them?

Ans : There are two widely known issues with properly training Recurrent Neural Networks, the vanishing and the exploding gradient problems.Recurrent Neural Networks enable you to model time-dependent and sequential data problems, such as stock market prediction, machine translation, and text generation. RNN is hard to train because of the gradient problem. RNNs suffer from the problem of vanishing gradients.

6) can you sketch the LSTM cells 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.



7) why would you want to use 1D convolutional layers in an RNN ?

Ans :1D CNN can perform activity recognition task from accelerometer data, such as if the person is standing, walking, jumping etc. This data has 2 dimensions. The first dimension is time-steps and the other is the values of the acceleration in 3 axes.The model learns to extract features from sequences of observations and how to map the internal features to different activity types.

8)which neural network architecture could you use to classify videos ?

Ans : Convolutional Neural Network (CNN) and a Recurrent Neural Network (RNN) consisting of GRU layers.this type of architecture classify the video.

Video Classification is the task of producing a label that is relevant to the video given its frames. A good video level classifier is one that not only provides accurate frame labels, but also best describes the entire video given the features and the annotations of the various frames in the video.

9)train a classification model for the sketch RNN dataset available in tensorflow dataset ?

Ans : This repo contains a set of optional, extra datasets for training sketch-rnn, a generative model for vector drawings.

Although the datasets had been created in the format customized for training sketch-rnn, it can, and should be used for training newer and better models to advance the state of generative vector image modeling.