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.

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.

2. Why do people use encoder–decoder RNNs rather than plain sequence-to-sequence RNNs

for automatic translation?

The encoder-decoder architecture for recurrent neural networks is the standard neural machine translation method that rivals and in some cases outperforms classical statistical machine translation methods. This architecture is very new, having only been pioneered in 2014, although, has been adopted as the core technology inside Google’s translate service.

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

The goal of RNN models is to extract the temporal correlation between the images by keeping a memory of past images. The block diagram of the model is as follows: The images of a video are fed to a CNN model to extract high-level features.

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

It is based on awhile\_loop()operation that is able to swap the GPU’s memory to theCPU’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

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

sequences?

The most common way people deal with inputs of varying length is padding. You first define the desired sequence length, i.e. the input length you want your model to have. Then any sequences with a shorter length than this are padded either with zeros or with special characters so that they reach the desired length

6. What is a common way to distribute training and execution of a deep RNN across multiple

GPUs?

There are two most common ways how to distribute training between mini-processors (in our case GPUs): data parallelism and model parallelism