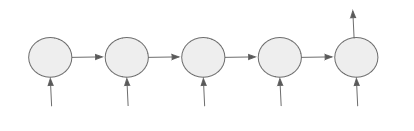
**Assignment 04**

**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: Application of Sequence to sequence RNN to solve complex Language problems like Machine Translation, Question Answering, creating Chatbots, Text Summarization, etc.

**sequence-to-vector RNN:** In the RNN model, where we give the input, a sequence at output comes out to be a single vector.  
An example of this could be language predictor.

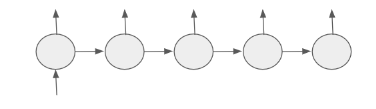
(input sequence) gives *Chinese*(output vector)



Seq2Vec model

**Vector to Sequence(Vec2Sec) RNN**

The RNN model where input is a single vector and outputs a sequence.



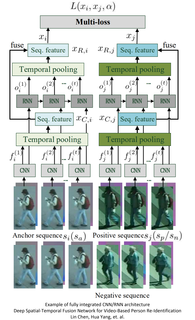
Vec2Seq model

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

Ans: The key benefits of the approach are the ability to train a single end-to-end model directly on source and target sentences and the ability to handle variable length input and output sequences of text.

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

* **Ans: CNNs** are good with hierarchical or spatial data and extracting unlabeled features. Those could be images or written characters.  CNNs take fixed size inputs and generate fixed size outputs.
* **RNNs** are good at temporal or otherwise sequential data. Could be letters or words in a body of text, stock market data, or speech recognition.  RNNs can input and output arbitrary lengths of data.  LSTMs are a variant of RNNs that allow for controlling how much of prior training data should be remembered, or more appropriately forgotten.



**Video Scene Labeling**

The classical approach to scene labeling is to train a CNN to identify and classify the objects within a frame and perhaps to further classify the objects into a higher level logical group.  For example, the CNN identifies a stove, a refrigerator, a sink, etc. and also up-classifies them as a kitchen.

Clearly the element that’s missing is the meaning of the motion over several frames (time).  For example, several frames of a game of pool might correctly say, the shooter sinks the eight ball in the side pocket.  Or several frames of a young person learning to ride a two-wheeler followed by the frame of the rider on the ground, might reasonably be summarized as ‘boy falls off bike’.

Researchers have used layered CNN-RNN pairs where the output of the CNN is input to the RNN.

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

Ans: Static: Internally, tf.nn.rnn creates an unrolled graph for a fixed RNN length. That means, if you call tf.nn.rnn with inputs having 200 time steps you are creating a static graph with 200 RNN steps. First, graph creation is slow. Second, you’re unable to pass in longer sequences (> 200) than you’ve originally specified.

Dynamic: tf.nn.dynamic\_rnn solves this. It uses a tf.While loop to dynamically construct the graph when it is executed. That means graph creation is faster and you can feed batches of variable size.

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

Ans: To address this issue, we can use padding and masking techniques. Padding involves adding zeros to the end of sequences to make them the same length, while masking involves ignoring the padded values during training.

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

**Ans:** **tf.distribute.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.