The hidden-to-hidden recurrent configuration is found to be superior to the output-to-hidden form because it better captures the high-dimensional feature information about the past. In any case, the output-to-hidden recurrent form is less computationally expensive to train and can more easily be parallelized.

Sequence Mappings

Recurrent neural networks can represent sequence problems in a variety of ways. The flexibility of RNN mappings is that it operates on inputs and outputs of the network as sequences, thus freeing the network from the fixed sized input-output constraints found in other neural network architectures such as MLP and CNN.

Here are a few examples of variating sequence problems solved using RNNs:

1. An input to a sequence of output. This configuration is used for image captioning problems when an image is passed as an input to the network, and the output is a sequence of words. See Figure 36-8.

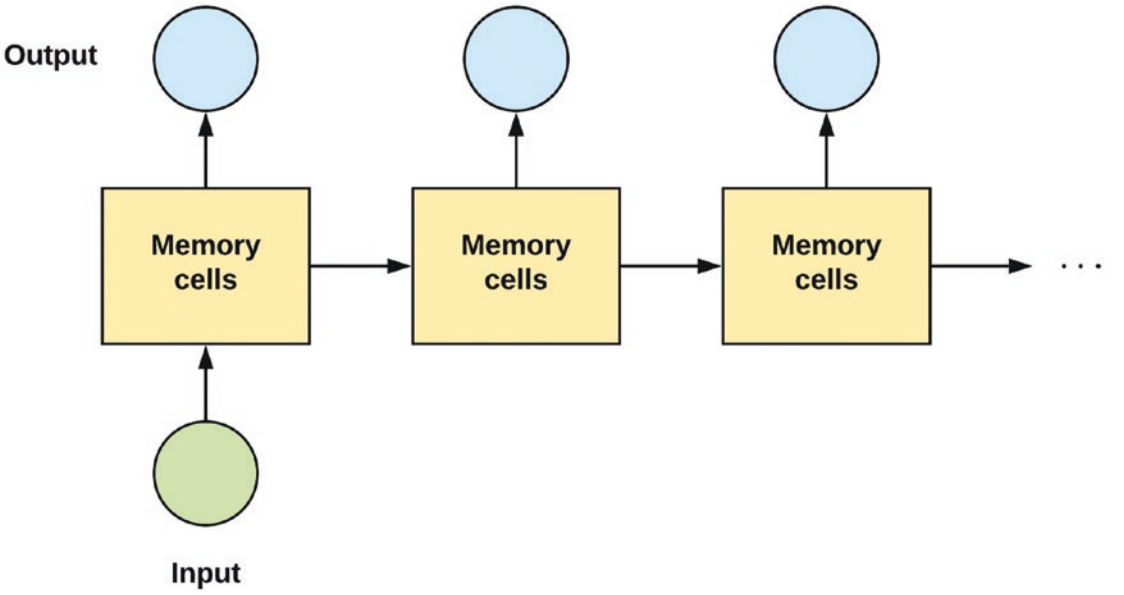

Figure 36-8. *An input to a sequence of output*

2. A sequence of inputs to an output. For example, in sentiment analysis, we need to pass in a sequence of words as input to the network, and the output is a class indicating either a positive or negative review or sentiment. See Figure 36-9.

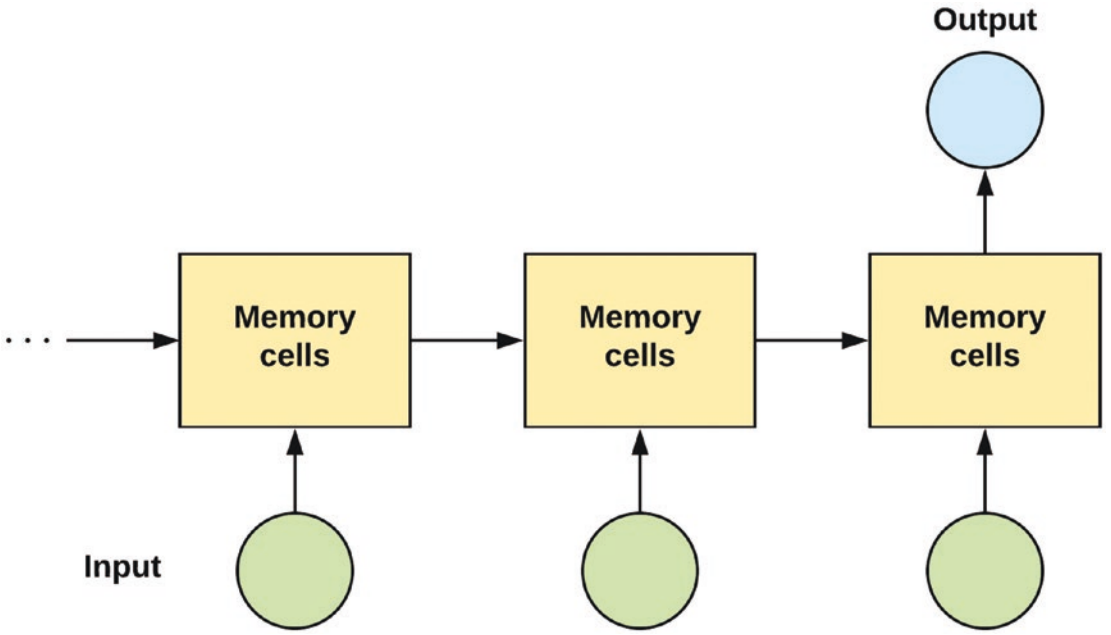

Figure 36-9. *A sequence of inputs to an output*

3. Sequence input to sequence output. This mapping operation is suited in application areas such as machine translation and speech recognition. It is more popularly called the encoder-decoder or sequence-to-sequence architecture. In this case, we may have a sequence of words in a particular language as input, and we want a sequence of words as output in another language. See Figure 36-10.

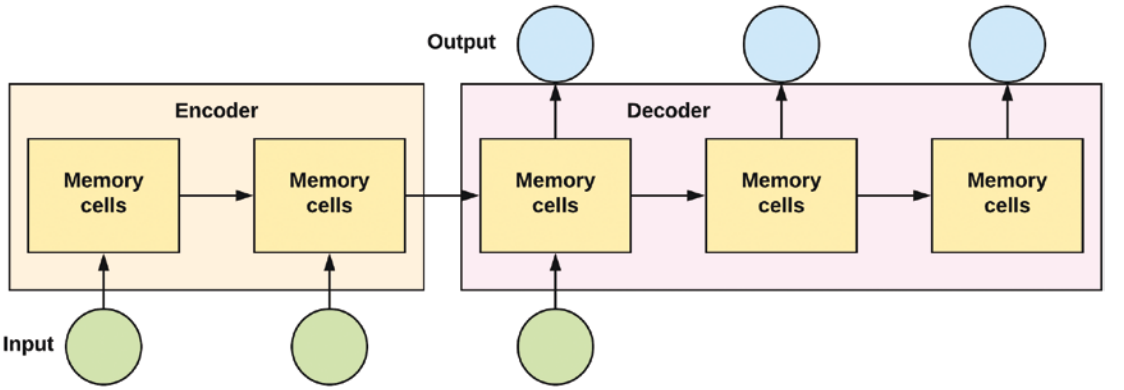

Figure 36-10. *Sequence input to sequence output*

4. Synced sequence input to output. This sort of framework is ideal for video classification in the event we want to label each video frame. See Figure 36-11.

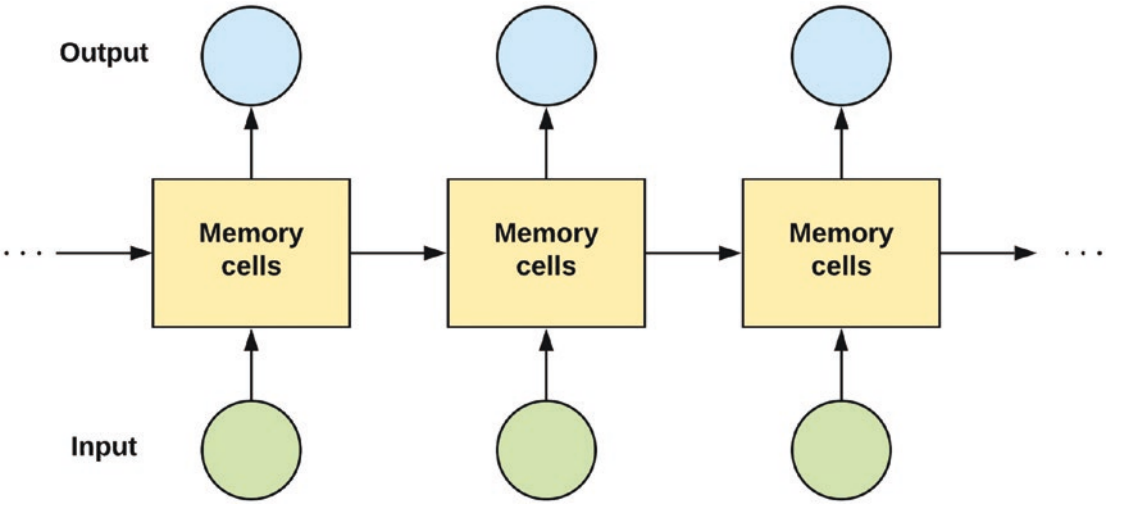

Figure 36-11. *Synced sequence input to output*

In the schemes illustrated in this sub-section, information flows from the hidden unit (or memory cell) of the recurrent layer at time instant $t-1$ to the hidden unit at time instant t. As discussed earlier, this is because the transferred information is more feature-rich and contains more information from the past.

Training the Recurrent Network: Backpropagation Through Time

The recurrent neural network is trained in much the same way as other traditional neural networks by using the backpropagation algorithm. However, the backpropagation algorithm is modified into what is called backpropagation through time (BPTT).

Due to the architectural loop or recurrent structure of the recurrent network, vanilla backpropagation as is cannot work. Training a network using backpropagation involves calculating the error gradient, moving backward from the output layer through the hidden layers of the network and adjusting the network weights. However, this operation cannot work in the recurrent neuron because we have just one neural cell with recurrent connections to itself.

So, in order to train the recurrent network using backpropagation, we unroll the recurrent neuron across the time instants and apply backpropagation to the unrolled neurons at each time layer the same way it is done for a traditional feedforward neural network. This operation is further illustrated in Figure 36-12.

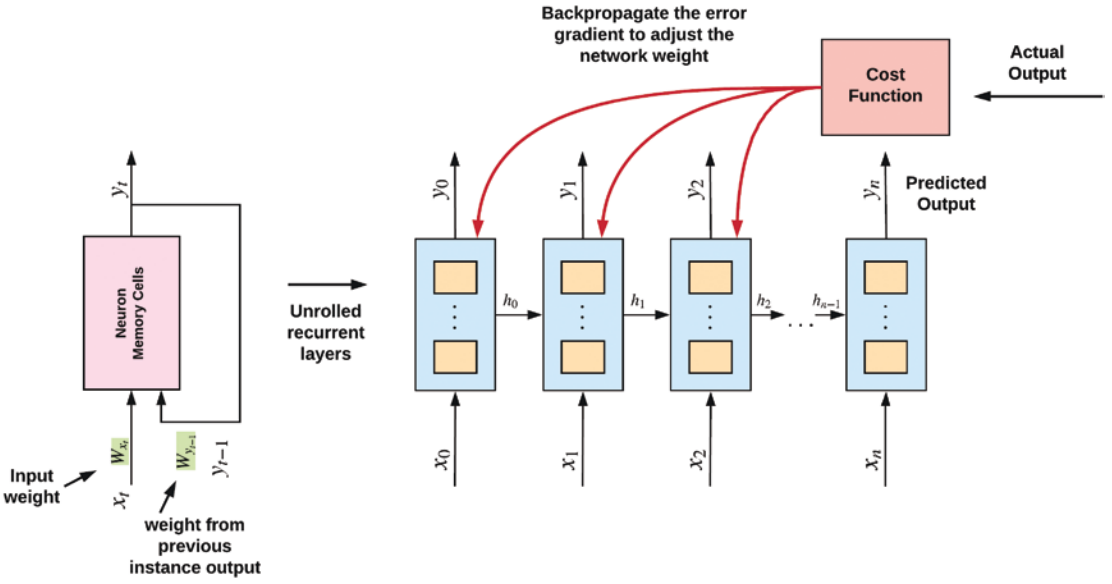

Figure 36-12. *Backpropagation through time*