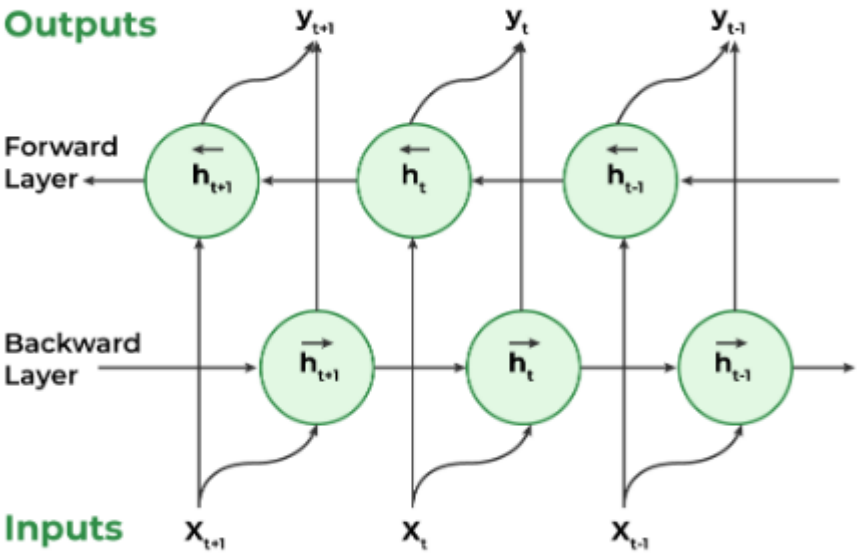
Bi-directional Recurrent Neural Network

An architecture of a neural network called a bidirectional recurrent neural network (BRNN) is made to process sequential data. In order for the network to use information from both the past and future context in its predictions, BRNNs process input sequences in both the forward and backward directions. This is the main distinction between BRNNs and conventional recurrent neural networks.

A BRNN has two distinct recurrent hidden layers, one of which processes the input sequence forward and the other of which processes it backward. After that, the results from these hidden layers are collected and input into a prediction-making final layer. Any recurrent neural network cell, such as Long Short-Term Memory (LSTM) or Gated Recurrent Unit, can be used to create the recurrent hidden layers.

The BRNN functions similarly to conventional recurrent neural networks in the forward direction, updating the hidden state depending on the current input and the prior hidden state at each time step. The backward hidden layer, on the other hand, analyses the input sequence in the opposite manner, updating the hidden state based on the current input and the hidden state of the next time step.

The input sequence is processed by the BRNN in a single forward pass at inference time, and predictions are made based on the combined outputs of the two hidden layers. layers.



Working of Bidirectional Recurrent Neural Network

1. **Inputting a sequence:** A sequence of data points, each represented as a vector with the same dimensionality, are fed into a BRNN. The sequence might have different lengths.

2. **Dual Processing:** Both the forward and backward directions are used to process the data. On the basis of the input at that step and the hidden state at step t-1, the hidden state at time step t is determined in the forward direction. The input at step t and the hidden state at step t+1 are used to calculate the hidden state at step t in a reverse way.
3. **Computing the hidden state:** A non-linear activation function on the weighted sum of the input and previous hidden state is used to calculate the hidden state at each step. This creates a memory mechanism that enables the network to remember data from earlier steps in the process.
4. **Determining the output:** A non-linear activation function is used to determine the output at each step from the weighted sum of the hidden state and a number of output weights. This output has two options: it can be the final output or input for another layer in the network.
5. **Training:** The network is trained through a supervised learning approach where the goal is to minimize the discrepancy between the predicted output and the actual output. The network adjusts its weights in the input-to-hidden and hidden-to-output connections during training through backpropagation.

To calculate the output from an RNN unit, we use the following formula:

$$H_t \text{ (Forward)} = A(X_t * W_{XH} \text{ (forward)} + H_{t-1} \text{ (Forward)} * W_{HH} \text{ (Forward)} + b_H \text{ (Forward)}$$

$$H_t \text{ (Backward)} = A(X_t * W_{XH} \text{ (Backward)} + H_{t+1} \text{ (Backward)} * W_{HH} \text{ (Backward)} + b_H \text{ (Backward)}$$

where,
A = activation function,
W = weight matrix
b = bias

The hidden state at time t is given by a combination of H_t (Forward) and H_t (Backward). The output at any given hidden state is :

$$Y_t = H_t * W_{AY} + b_y$$

The training of a BRNN is similar to backpropagation through a time algorithm. BPTT algorithm works as follows:

- Roll out the network and calculate errors at each iteration
- Update weights and roll up the network.

Applications of Bidirectional Recurrent Neural Network

Bi-RNNs have been applied to various natural language processing (NLP) tasks, including:

1. **Sentiment Analysis:** By taking into account both the prior and subsequent context, BRNNs can be utilized to categorize the sentiment of a particular sentence.
2. **Named Entity Recognition:** By considering the context both before and after the stated thing, BRNNs can be utilized to identify those entities in a sentence.

3. **Part-of-Speech Tagging:** The classification of words in a phrase into their corresponding parts of speech, such as nouns, verbs, adjectives, etc., can be done using BRNNs.
4. **Machine Translation:** BRNNs can be used in encoder-decoder models for machine translation, where the decoder creates the target sentence and the encoder analyses the source sentence in both directions to capture its context.
5. **Speech Recognition:** When the input voice signal is processed in both directions to capture the contextual information, BRNNs can be used in automatic speech recognition systems.