A significant challenge of training the recurrent neural network is the vanishing and exploding gradient problem. When training a deep recurrent network for many layers of time instants, calculating the gradients of the weights of the neurons can become very volatile. When this happens, the value of the gradient can become extremely large tending to infinity, or they become tiny, all the way to zero. When this happens, the neurons become dead and cannot train or learn any new information further. This effect is called the exploding and vanishing gradient problem.

The exploding and vanishing gradient problem is most prevalent in recurrent neural networks because of the long-term dependencies or time instant of the unrolled recurrent neuron. A proposed alternative technique for mitigating this problem in recurrent networks (in addition to other discussed methods such as gradient clipping, batch normalization, and using a non-saturating activation function such as ReLu) is to discard early time instances or time instances in the distant past. This technique is called Truncated Backpropagation Through Time (truncated BPTT).

However, truncated BPTT suffers a significant drawback, and this is that some problems rely heavily on long-term dependencies to be able to make a prediction. A typical example is in language modeling where the long-term sequence of words in the past is vital in predicting the next word in the sequence.

The shortcoming of truncated BPTT and the need to deal with the problem of exploding and vanishing gradients led to the development of a memory cell called the Long Short-Term Memory or LSTM for short, which can store the long-term information of the problem in the memory cell of the recurrent network.

The Long Short-Term Memory (LSTM) Network

Long Short-Term Memory (LSTM) belongs to a class of RNN called gated recurrent unit. They are called *gated* because unlike the basic recurrent units, they contain extra components called gates that control the flow of information within the recurrent cell. This includes choosing what information to store in the cell and what information to discard or forget.

LSTM is very efficient for capturing the long-term dependencies across a large number of time instants. It does this by having a slightly more sophisticated cell than the basic recurrent units. The components of the LSTM are the

- Memory cell
- Input gate

- Forget gate
- Output gate

These extra components enable the RNN to remember and store important events from the distant past. The LSTM takes as input the previous cell state, c_{t-1}; the previous hidden state, h_{t-1}; and the current input, x_t. To keep in line with the simplicity of this book, we provide a high-level illustration of the LSTM cell showing how the extra components of the cell come together. In TensorFlow 2.0, LSTM layer is implemented in the method **'tf.keras.layers.LSTM()'**.

The illustration in Figure 36-13 is the LSTM memory cell. The components of the LSTM cell serve distinct functions in preserving long-term dependencies in sequence data. Let's go through them:

- The input gate: This gate is responsible for controlling what information gets stored in the long-term state or the memory cell, c. Working in tandem with the input gate is another gate that regulates the information flowing into the input gate. This gate analyzes the current input to the LSTM cell, x_t, and the previous short-term state, h_{t-1}.
- The forget gate: The role of this gate is to regulate how much of the information in the long-term state is persisted across time instants.
- The output gate: This gate controls how much information to output from the cell at a particular time instant. This gate controls the value of h_t (the short-term state) and y_t (the output at time t).

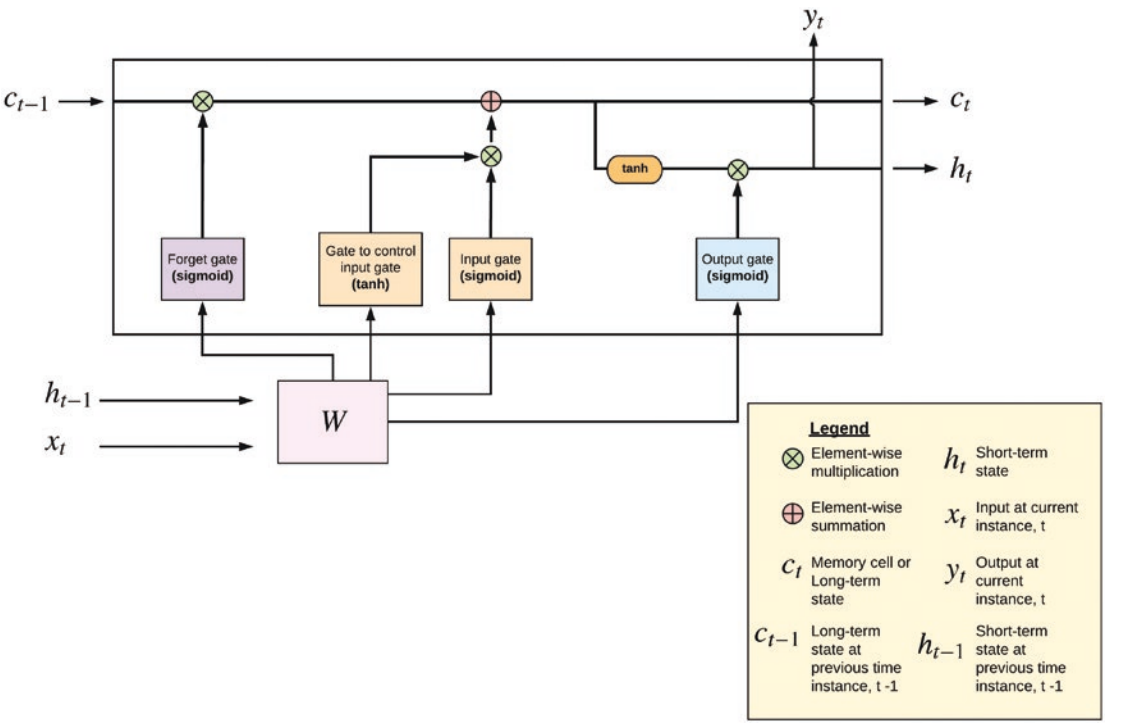

Figure 36-13. *LSTM cell*

It is important to note that the components of the LSTM cells are all fully connected neural networks. There exist other variants of recurrent networks with memory cells, two of such are the peephole connections and the gated recurrent units.

Peephole Connection

The peephole connection extends the LSTM network by also using information from the memory cell or long-term state of the previous time instant c_{t-1} as input to the LSTM gates. The goal of the peephole is to provide extra information into the LSTM unit by peeping at the stored long-term memory. This is further illustrated in Figure 36-14.
In TensorFlow 2.0, the implementation of peephole connections to an LSTM layer is provided by the method **'tf.keras.experimental.PeepholeLSTMCell()'**.