Equation 14-3 summarizes how to compute the cell's long-term state, its short-term state, and its output at each time step for a single instance (the equations for a whole mini-batch are very similar).

*Equation 14-3. LSTM computations*

$$
\begin{aligned}
\mathbf{i}_{(t)} &= \sigma\left(\mathbf{W}_{xi}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hi}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_i\right) \\
\mathbf{f}_{(t)} &= \sigma\left(\mathbf{W}_{xf}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hf}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_f\right) \\
\mathbf{o}_{(t)} &= \sigma\left(\mathbf{W}_{xo}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{ho}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_o\right) \\
\mathbf{g}_{(t)} &= \tanh\left(\mathbf{W}_{xg}{}^T \cdot \mathbf{x}_{(t)} + \mathbf{W}_{hg}{}^T \cdot \mathbf{h}_{(t-1)} + \mathbf{b}_g\right) \\
\mathbf{c}_{(t)} &= \mathbf{f}_{(t)} \otimes \mathbf{c}_{(t-1)} + \mathbf{i}_{(t)} \otimes \mathbf{g}_{(t)} \\
\mathbf{y}_{(t)} &= \mathbf{h}_{(t)} = \mathbf{o}_{(t)} \otimes \tanh\left(\mathbf{c}_{(t)}\right)
\end{aligned}
$$

- $\mathbf{W}_{xi}$, $\mathbf{W}_{xf}$, $\mathbf{W}_{xo}$, $\mathbf{W}_{xg}$ are the weight matrices of each of the four layers for their connection to the input vector $\mathbf{x}_{(t)}$.
- $\mathbf{W}_{hi}$, $\mathbf{W}_{hf}$, $\mathbf{W}_{ho}$, and $\mathbf{W}_{hg}$ are the weight matrices of each of the four layers for their connection to the previous short-term state $\mathbf{h}_{(t-1)}$.
- $\mathbf{b}_i$, $\mathbf{b}_f$, $\mathbf{b}_o$, and $\mathbf{b}_g$ are the bias terms for each of the four layers. Note that TensorFlow initializes $\mathbf{b}_f$ to a vector full of 1s instead of 0s. This prevents forgetting everything at the beginning of training.

## Peephole Connections

In a basic LSTM cell, the gate controllers can look only at the input $\mathbf{x}_{(t)}$ and the previous short-term state $\mathbf{h}_{(t-1)}$. It may be a good idea to give them a bit more context by letting them peek at the long-term state as well. This idea was proposed by Felix Gers and Jürgen Schmidhuber in 2000.[6] They proposed an LSTM variant with extra connections called *peephole connections*: the previous long-term state $\mathbf{c}_{(t-1)}$ is added as an input to the controllers of the forget gate and the input gate, and the current long-term state $\mathbf{c}_{(t)}$ is added as input to the controller of the output gate.

To implement peephole connections in TensorFlow, you must use the `LSTMCell` instead of the `BasicLSTMCell` and set `use_peepholes=True`:

```
lstm_cell = tf.contrib.rnn.LSTMCell(num_units=n_neurons, use_peepholes=True)
```

6 "Recurrent Nets that Time and Count," F. Gers and J. Schmidhuber (2000).

There are many other variants of the LSTM cell. One particularly popular variant is the GRU cell, which we will look at now.

## GRU Cell

The *Gated Recurrent Unit* (GRU) cell (see Figure 14-14) was proposed by Kyunghyun Cho et al. in a 2014 paper[7] that also introduced the Encoder–Decoder network we mentioned earlier.

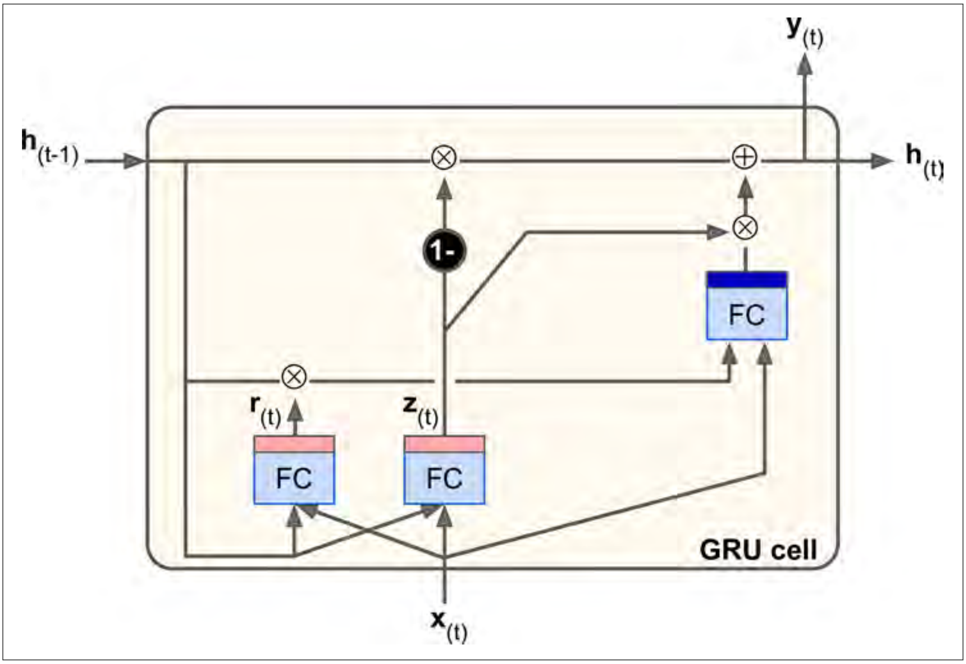


*Figure 14-14. GRU cell*

The GRU cell is a simplified version of the LSTM cell, and it seems to perform just as well[8] (which explains its growing popularity). The main simplifications are:

- Both state vectors are merged into a single vector $\mathbf{h}_{(t)}$.
- A single gate controller controls both the forget gate and the input gate. If the gate controller outputs a 1, the input gate is open and the forget gate is closed. If

7 "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," K. Cho et al. (2014).

8 A 2015 paper by Klaus Greff et al., "LSTM: A Search Space Odyssey," seems to show that all LSTM variants perform roughly the same.