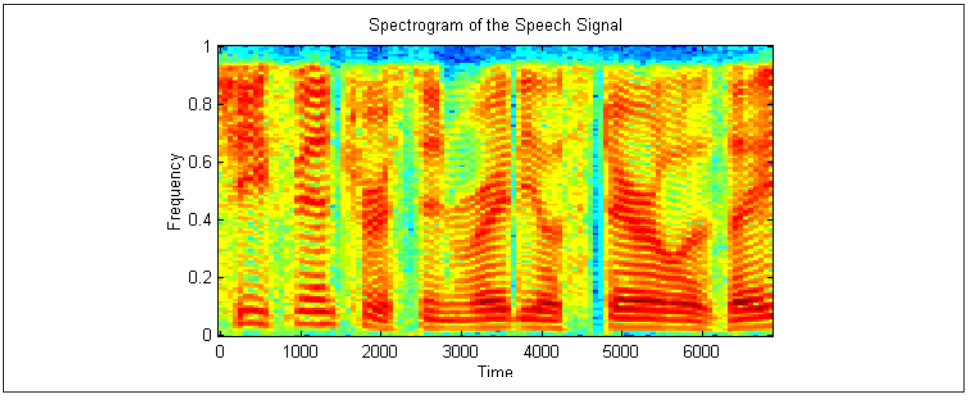

Figure 7-3. A speech spectrogram representing the frequencies found in a speech sample.

Recurrent Cells

Gradient Instability

Recurrent networks tend to degrade signal over time. Think of it as attenuating a signal by a multiplicative factor at each timestep. As a result, after 50 timesteps, the signal is quite degraded.

As a result of this instability, it has been challenging to train recurrent neural networks on longer time-series. A number of methods have arisen to combat this instability, which we will discuss in the remainder of this section.

There are a number of elaborations on the concept of a simple recurrent neural network that have proven significantly more successful in practical applications. In this section, we will briefly review some of these variations.

Long Short-Term Memory (LSTM)

Part of the challenge with the standard recurrent cell is that signals from the distant past attenuate rapidly. As a result, RNNs can fail to learn models of complex dependencies. This failure becomes particularly notable in applications such as language modeling, where words can have complex dependencies on earlier phrases.

One potential solution to this issue is to allow states from the past to pass through unmodified. The long short-term memory (LSTM) architecture proposes a mechanism to allow past state to pass through to the present with minimal modifications. Empirically using an LSTM "cell" (shown in Figure 7-4) seems to offer superior learning performance when compared to simple recurrent neural networks using fully connected layers.

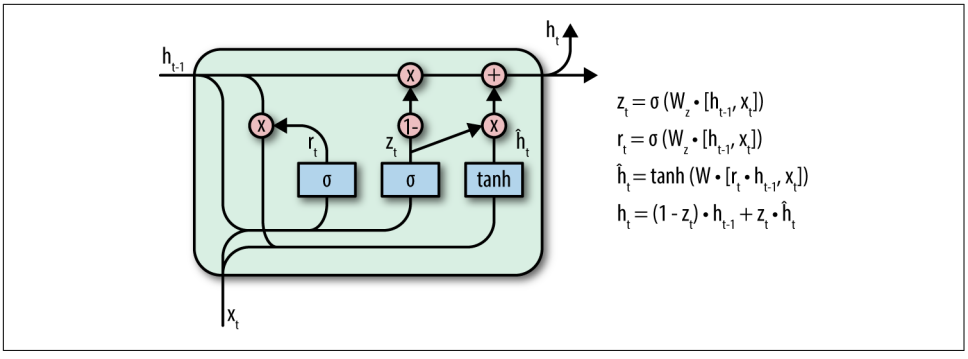

Figure 7-4. A long short-term memory (LSTM) cell. LSTMs perform better than standard recurrent neural networks at preserving long-range dependencies in inputs. As a result, LSTMs are often preferred for complex sequential data, such as natural language.

So Many Equations!

The LSTM equations involve many sophisticated terms. If you are interested in understanding precisely the mathematical intuitions behind the LSTM, we encourage you to play with the equations with pencil and paper and trying to take derivatives of the LSTM cell.

However, for other readers who are primarily interested in using recurrent architectures to solve practical problems, we believe it isn't absolutely necessary to delve into the nitty-gritty details of how LSTMs work. Rather, keep the high-level intuition that past state is allowed to pass through, and work through the example code for this chapter in some depth.

Optimizing Recurrent Networks

Unlike fully connected networks or convolutional networks, LSTMs involve some sophisticated mathematical operations and control-flow operations. As a result, training large recurrent networks at scale has proven to be challenging, even with modern GPU hardware.

Significant effort has been put into optimizing RNN implementations to run quickly on GPU hardware. In particular, Nvidia has incorporated RNNs into their CuDNN library that provides specially optimized code for training deep networks on GPUs. Luckily for TensorFlow users, integration with libraries such as CuDNN is performed within TensorFlow itself so you don't need to worry too much about code optimization (unless of course, you're working on very large-scale datasets). We will discuss hardware needs for deep neural networks at greater depth in Chapter 9.

Gated Recurrent Units (GRU)

The complexity, both conceptual and computational, for LSTM cells has motivated a number of researchers to attempt to simplify the LSTM equations while retaining the performance gains and modeling capabilities of the original equations.

There are a number of contenders for LSTM replacement, but one of the frontrunners is the gated recurrent unit (GRU), shown in Figure 7-5. The GRU removes one of the subcomponents of the LSTM but empirically seems to achieve performance similar to that of the LSTM. The GRU might be a suitable replacement for LSTM cells on sequence modeling projects.

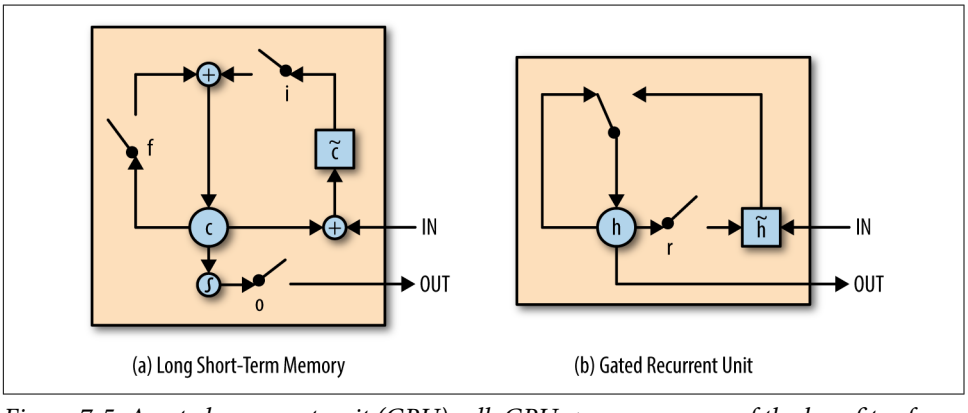

Figure 7-5. A gated recurrent unit (GRU) cell. GRUs preserve many of the benefits of LSTMs at lower computational cost.

Applications of Recurrent Models

While recurrent neural networks are useful tools for modeling time-series datasets, there are a host of other applications of recurrent networks. These include applications such as natural language modeling, machine translation, chemical retrosynthesis, and arbitrary computation with Neural Turing machines. In this section, we provide a brief tour of some of these exciting applications.

Sampling from Recurrent Networks

So far, we've taught you how recurrent networks can learn to model the time evolution of sequences of data. It stands to reason that if you understand the evolution rule for a set of sequences, you ought to be able to sample new sequences from the distribution of training sequences. And indeed, it turns out that that good sequences can be sampled from trained models. The most useful application thus far is in language modeling. Being able to generate realistic sentences is a very useful tool that underpins systems such as autocomplete and chatbots.