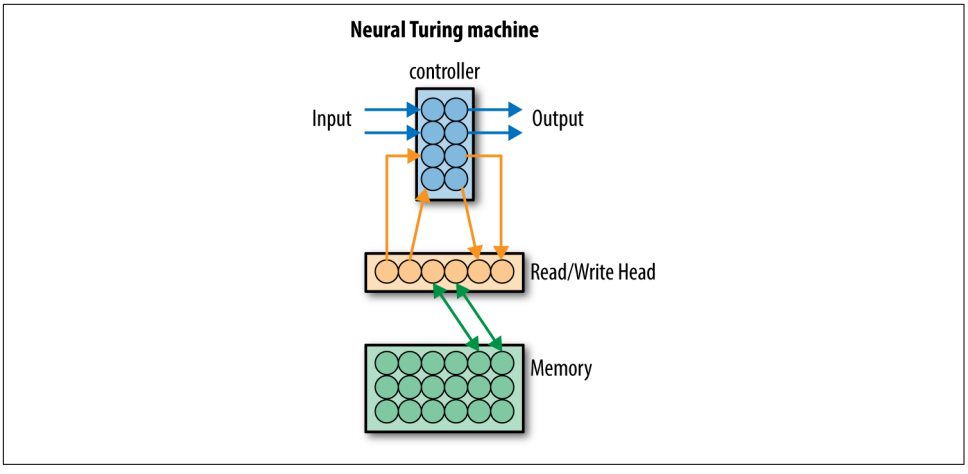

Figure 1-14. A conceptual depiction of a Neural Turing machine. It adds an external memory bank to which the deep architecture reads and writes.

Deep Learning Frameworks

Researchers have been implementing software packages to facilitate the construction of neural network (deep learning) architectures for decades. Until the last few years, these systems were mostly special purpose and only used within an academic group. This lack of standardized, industrial-strength software made it difficult for non-experts to use neural networks extensively.

This situation has changed dramatically over the last few years. Google implemented the DistBelief system in 2012 and made use of it to construct and deploy many simpler deep learning architectures. The advent of DistBelief, and similar packages such as Caffe, Theano, Torch, Keras, MxNet, and so on have widely spurred industry adoption.

TensorFlow draws upon this rich intellectual history, and builds upon some of these packages (Theano in particular) for design principles. TensorFlow (and Theano) in particular use the concept of tensors as the fundamental underlying primitive powering deep learning systems. This focus on tensors distinguishes these packages from systems such as DistBelief or Caffe, which don't allow the same flexibility for building sophisticated models.

While the rest of this book will focus on TensorFlow, understanding the underlying principles should enable you to take the lessons learned and apply them with little difficulty to alternative deep learning frameworks.

Limitations of TensorFlow

One of the major current weaknesses of TensorFlow is that constructing a new deep learning architecture is relatively slow (on the order of multiple seconds to initialize an architecture). As a result, it's not convenient in TensorFlow to construct some sophisticated deep architectures that change their structure dynamically. One such architecture is the TreeLSTM, which uses syntactic parse trees of English sentences to perform tasks that require understanding of natural language. Since each sentence has a different parse tree, each sentence requires a slightly different architecture. Figure 1-15 illustrates the TreeLSTM architecture.

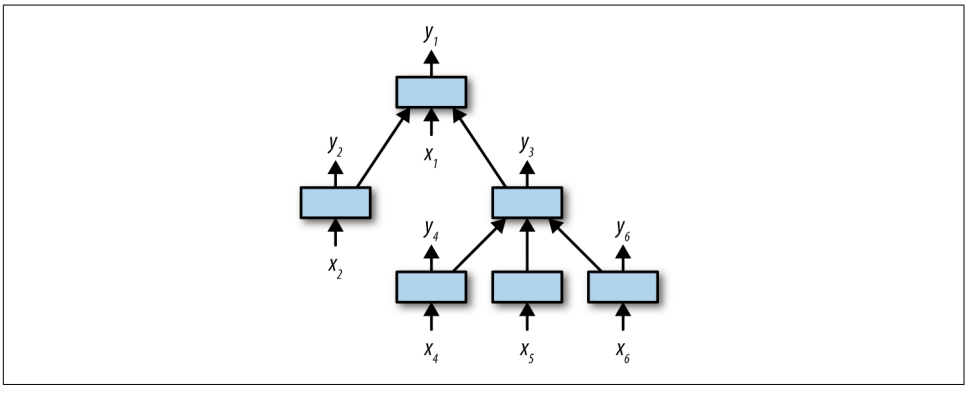

Figure 1-15. A conceptual depiction of a TreeLSTM architecture. The shape of the tree is different for each input datapoint, so a different computational graph must be constructed for each example.

While such models can be implemented in TensorFlow, doing so requires significant ingenuity due to the limitations of the current TensorFlow API. New frameworks such as Chainer, DyNet, and PyTorch promise to remove these barriers by making the construction of new architectures lightweight enough so that models like the TreeLSTM can be constructed easily. Luckily, TensorFlow developers are already working on extensions to the base TensorFlow API (such as TensorFlow Eager) that will enable easier construction of dynamic architectures.

One takeaway is that progress in deep learning frameworks is rapid, and today's novel system can be tomorrow's old news. However, the fundamental principles of the underlying tensor calculus date back centuries, and will stand readers in good stead regardless of future changes in programming models. This book will emphasize using TensorFlow as a vehicle for developing an intuitive knowledge of the underlying tensor calculus.