Kai Hwang : *Cloud Computing for Machine Learning and Cognitive Applications*
The MIT Press, June 2017

Lectures 21 and 22, 2017, USC EE 542

Chapter 9: TensorFlow, Keras, and DeepMind

DeepMind and Brain Projects at Google in the Past 5 Years

- In March 2016, Google AlphaGo program defeated a top Go player. This has opened up the debate between human vs. machine intelligence.
- Another reported advance is the Google Brain Project. In June 2012, tens of millions of random images from YouTube were recognized by a computer platform built over 16,000 CPU cores at Google.
- They use a training model over a deep neural network built with 1 billion of artificial neurons. This model system identified basic features of images and succeeded in recognize a cat out of thousands classes of images.
- The project leader Andrew Ng once said: "We directly put massive data (images) into the artificial system and the system learned (remembered) from the key features of those images, automatically."

Developer(s)	Google Brain Team[1]
Initial release	November 9, 2015; 21 months ago (2015-11-09)
Stable release	1.3.0[2] / August 17, 2017; 18 days ago (2017-08-17)
Repository	github.com/tensorflow/tensorflow
Written in	Python, C++, CUDA
Platform	Linux, macOS, Windows, Android
Type	Machine learning library
License	Apache 2.0 open source license
Website	www.tensorflow.org

- TensorFlow is Google Brain's second generation system. Version 1.0.0 was released on February 11, 2017.
- While the reference implementation runs on single devices, TensorFlow can run on multiple CPUs and GPUs (with optional CUDA extensions for general-purpose computing on graphics processing units).
- TensorFlow is available on 64-bit Linux, macOS, Windows, and mobile computing platforms including Android and iOS.
- TensorFlow computations are expressed as stateful dataflow graphs. The name TensorFlow derives from the operations that such neural networks perform on multidimensional data arrays. These arrays are referred to as "tensors".
- In June 2016, Dean stated that 1,500 repositories on GitHub mentioned TensorFlow, of which only 5 were from Google.

Tensor processing unit (TPU)

- In May 2016 Google announced its tensor processing unit (TPU), a custom ASIC built specifically for machine learning and tailored for TensorFlow.
- TPU is a programmable AI accelerator designed to provide high throughput of low-precision arithmetic (e.g., 8-bit), and oriented toward using or running models rather than training them.
- Google announced they had been running TPUs inside their data centers for more than a year, and have found them to deliver an order of magnitude better-optimized performance per watt for machine learning.
- In May 2017 Google announced the second-generation, as well as the availability of the TPUs in Google Compute Engine.
- The second-generation TPUs deliver up to 180 teraflops of performance, and when organized into clusters of 64 TPUs, provide up to 11.5 petaflops.

Some Reported TensorFlow Applications

Among the applications for which TensorFlow is the foundation, are automated image captioning software, such as DeepDream, AlphaGo, DeepMind, etc.

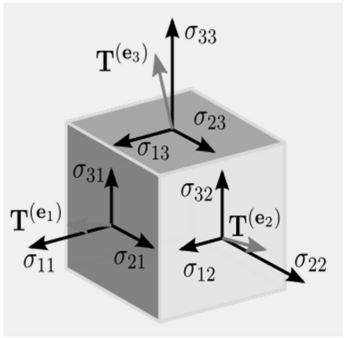
RankBrain now handles a substantial number of search queries, replacing and supplementing traditional static algorithm based search results.

Related Fields To TensorFlow Applications

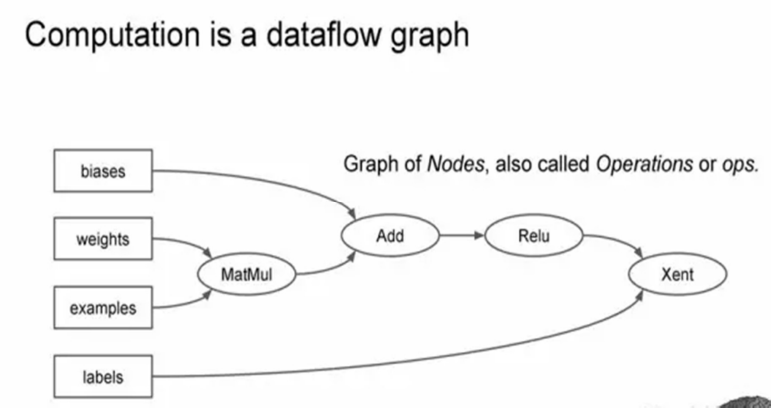
- Artificial neural network
- Comparison of deep learning software
- Convolutional neural network
- Deep learning
- Machine learning

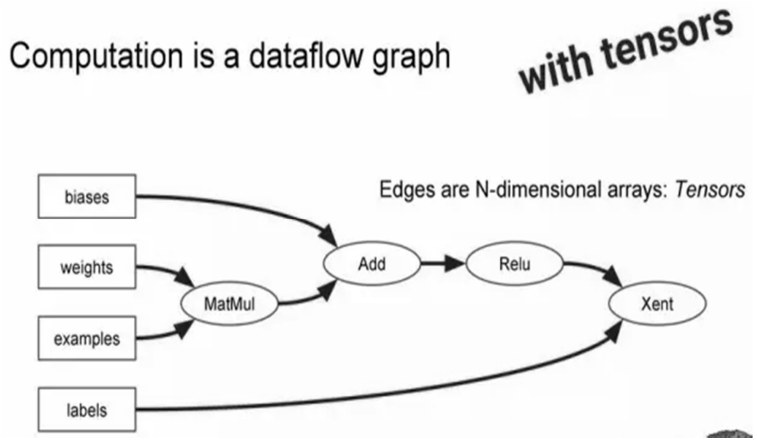
In mathematics tensors are geometric objects that describe linear relations between geometric vectors, scalars, and other tensors. Elementary examples of such relations include the dot product, the cross product, and linear maps. Geometric vectors, often used in physics and engineering applications, and scalars themselves are also tensors.

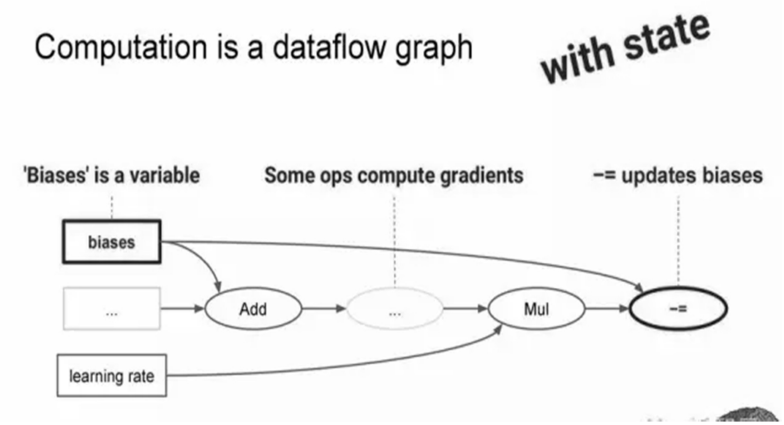
- Given a reference basis of vectors, a tensor can be represented as an organized multidimensional array of numerical values.
- The *order* (also *degree* or *rank*) of a tensor is the dimensionality of the array needed to represent it, or equivalently, the number of indices needed to label a component of that array.
- For example, a linear map is represented by a matrix (a 2-dimensional array) in a basis, and therefore is a 2nd-order tensor.
- A vector is represented as a 1-dimensional array in a basis, and is a 1st-order tensor. Scalars are single numbers and are thus 0th-order tensors. The collection of tensors on a vector space forms a tensor algebra.

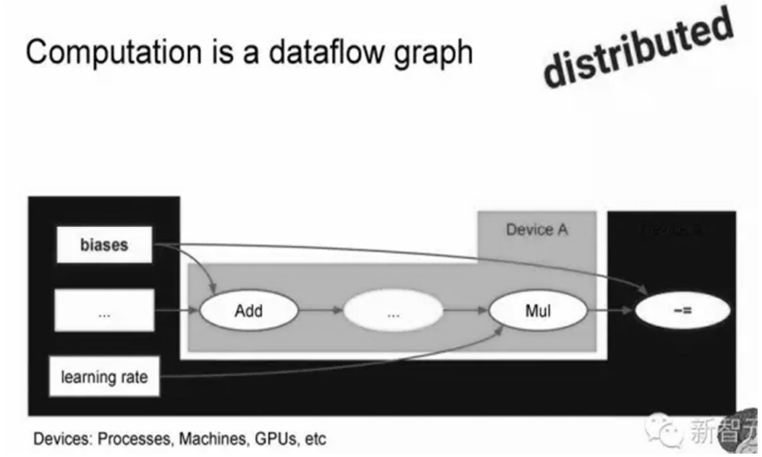


TensorFlow: The Software Framework for Deep Learning Applications









Operations and Kernels in TensorFlow:

An *operation* has a name and represents an abstract computation (e.g., "matrix multiply", or "add"). An operation can have *attributes*, and all attributes must be provided or inferred at graph-construction time in order to instantiate a node to perform the operation. One common use of attributes is to make operations polymorphic over different tensor element types (e.g., add of two tensors of type float versus add of two tensors of type int32). Google provides special cloud-based APIs for translate, speech, vision and text applications.

A *kernel* is a particular implementation of an operation that can be run on a particular type of device (e.g., CPU or GPU). A TensorFlow binary defines the sets of operations and kernels available via a registration mechanism. This set can be extended by linking in additional operation and/or kernel definitions/registrations. Table 9.1 shows operations types built into the core of TensorFlow library:

Tensors:

A tensor in our implementation is a typed, multidimensional array. We support a variety of tensor element types, including signed and unsigned integers ranging in size from 8 bits to 64 bits, IEEE float and double types, a complex number type, and a string type (an arbitrary byte array). Backing store of the appropriate size is managed by an allocator that is specific to the device on which the tensor resides. Tensor backing store buffers are reference counted and are deallocated when no references remain.

Sessions:

Clients programs interact with the TensorFlow system by creating a *Session.* To create a computation graph, the Session interface supports an Extend method to augment the current graph managed by the session with additional nodes and edges. Assume the initial graph when a session is created is empty. The other primary operation supported by the session interface is *Run,* which takes a set of output names that need to be computed, as well as an optional set of tensors to be fed into the graph in place of certain outputs of nodes.


```
import tensorflow as tf
b = tf.Variable(tf.zeros([100]))      // 100-d vector, initialize to zeroes
W = tf.Variable(tf.random_uniform([784,100],-1,1))      // 784x100 matrix w/rnd vals
x = tf.placeholder(name="x")                  // Placeholder for input
relu = tf.nn.relu(tf.matmul(W, x) + b)        // Relu(Wx+b)
C = [...]                                     // Cost computed as a function of relu
s = tf.Session()
for step in xrange(0, 10):
input = ...construct 100-D input array ...  // Create 100-d vector for input
result = s.run(C, feed_dict={x: input})     // Fetch cost, feeding x=input
print step, result
```

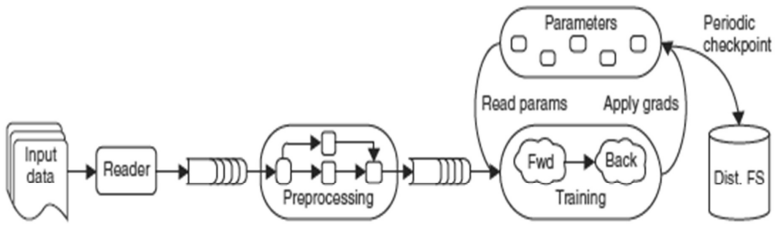
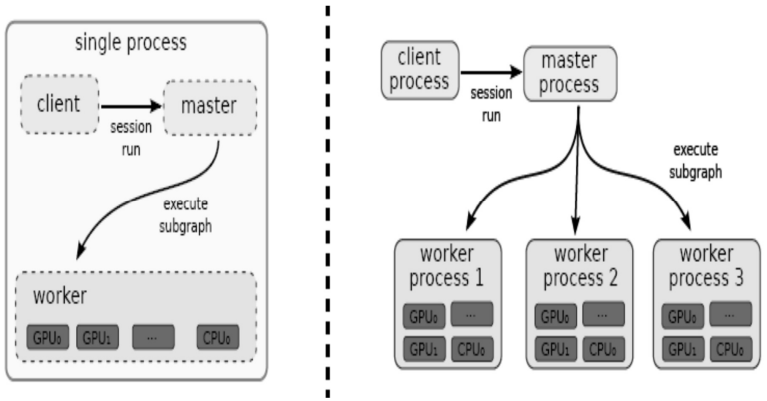



Figure 9.1
TensorFlow data flow graph for a training pipeline in DL applications. Reprinted with permission from M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," Google Brain Team White Paper, November 15, 2015.

Tensor Processing Unit (TPU)
Built by Google To Accelerate TensorFlow Execution



16

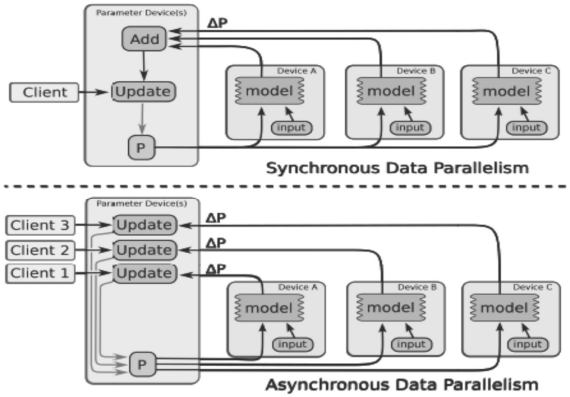


Figure 9.5 Synchronous and asynchronous data parallel training methods

Table 9.1
Examples of operation types built into TensorFlow core

Category	Examples
Element-wise mathematical operations	Add, Sub, Mul, Div, Exp, Log, Greater, Less, Equal, etc.
Array operations	Concat, SLice, Split, Constant, Rank, Shape, Shuffle, etc.
Matrix operations	MatMul, MatrixInverse, MatrixDeterminant, etc.
Stateful operations	Variable, Assign, AssignAdd, etc.
Neural-net building blocks	SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, etc.
Checkpointing operations	Save, Restore
Queue and synchronization operations	Enqueue, Dequeue, MutexAcquire, MutexRelease, etc.
Control flow operations	Merge, Switch, Enter, Leave, NextIteration

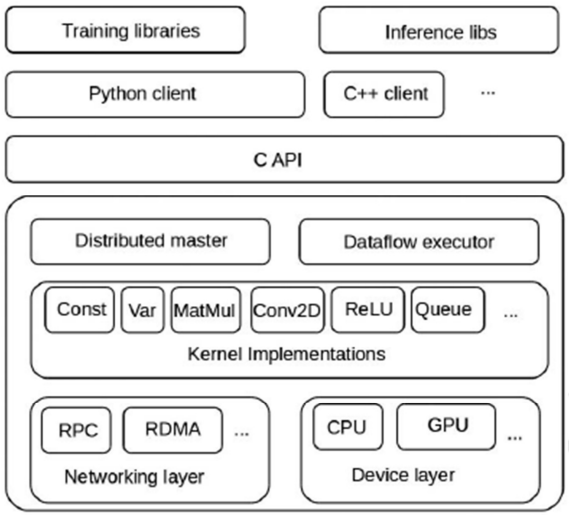
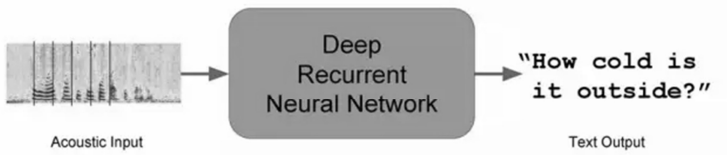


Figure 9.6
The layered architecture of TensorFlow architecture. (Reprinted with permission from M. Abadi et al., "TensorFlow: A System for Large-Scale Machine Learning," Google Brain Team White Paper, November 15, 2015.)

Table 9.2
Some available TensorFlow models for machine learning applications

Model Name	Brief Description
Autoencoder	Various autoencoders
Inception	Deep convolutional networks for computer vision
Nameignizer	Recognize and generate names
Neural_GPU	Highly parallel neural computer
Privacy	Privacy-preserving student models from multiple teachers
Resnet	Deep and wide residual networks
Slim	Image classification models in TF-Slim
Swivel	The Swivel algorithm for generating word embeddings
Syntaxnet	Neural models of natural language syntax
Textsum	Sequence-to-sequence with attention model for text summarization
Transformer	Spatial transformer network allowing spatial data manipulation
Im2txt	Image-to-text neural network for image captioning

Speech Recognition



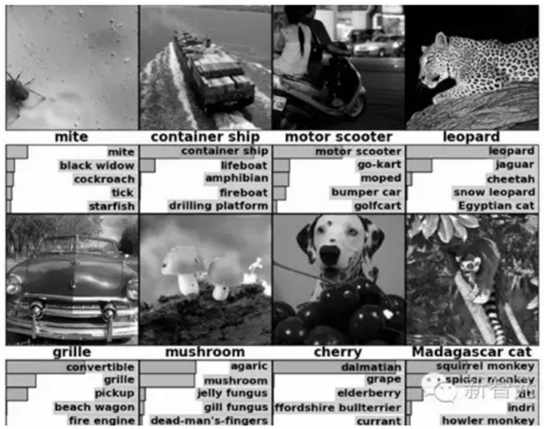
Reduced word errors by more than 30%

Google Research Blog - August 20[illegible] August 2015

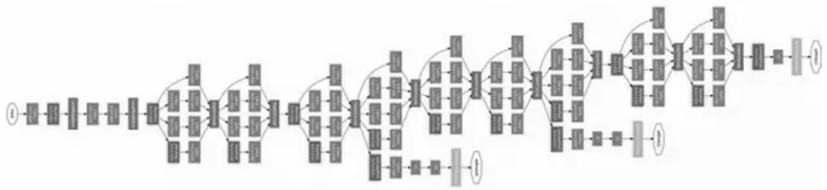
ImageNet Challenge

Given an image, predict one of 1000 different classes

Image credit: www.cs.toronto.edu/~fritz/absps/imagenet.pdf



The Inception Architecture (GoogLeNet, 2014)



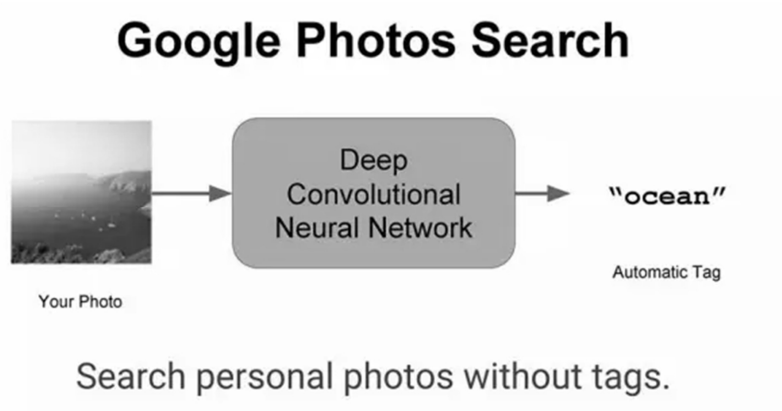
Going Deeper with Convolutions

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich

Neural Nets: Rapid Progress in Image Recognition

Team	Year	Place	Error (top-5)
XRCE (pre-neural-net explosion)	2011	1st	**25.8%**
Supervision (AlexNet)	2012	1st	16.4%
Clarifai	2013	1st	11.7%
GoogLeNet (Inception)	2014	1st	6.66%
Andrej Karpathy (**human**)	2014	N/A	5.1%
BN-Inception (Arxiv)	2015	N/A	4.9%
Inception-v3 (Arxiv)	2015	N/A	**3.46%**

ImageNet challenge classification task



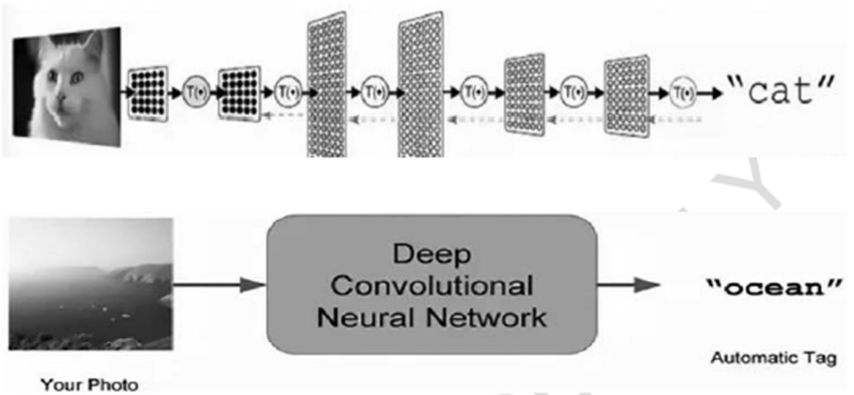


Figure 9.9
Using a deep convolution neural network to recognize a particular image out of millions of photos belonging to different or similar classes. (a) Reconizing a cat (not a dog), (b) Distinguishing an ocean view from many other views. (Courtesy of Jeff Dean, Google Brain Team, 2016.)

Example 9.7 The Use of Google's ImageNet for Image Understanding

ANN models, especially DCNNs, are undergoing a reincarnation. They offer a collection of simple, trainable mathematical functions that are compatible with many variants of ML. Figure 9.9 shows the idea of using a DCNN to recognize a "cat" or an "ocean" from a thousand classes over millions of photo images. Image captioning has been in high demand in Google search requests.

Searching for a personal photo without tags is equivalent to the task of identifying one image out of 1,000 different classes. The work was carried out at Google using ImageNet. Another project using GoogLeNet also emphasizes a deeper convolution approach in the inception area. Neural networks have made rapid progress in image recognition. The ImageNet project challenges many classification tasks. The Inception team using GoogLeNet reduced the error rate to 6.66% in 2014. ■

(4) Develop your own machine learning models

https://www.tensorflow.org/versions/master/get_started/basic_usage.html

TensorFlow ™ GET STARTED

Overview

TensorFlow is a programming system in which you represent computations as graphs. Nodes in the graph are called ops (short for operations). An op takes zero or more **Tensors**, performs some computation, and produces zero or more **Tensors**. A **Tensor** is a typed multi-dimensional array. For example, you can represent a mini-batch of images as a 4-D array of floating point numbers with dimensions `[batch, height, width, channels]`.

A TensorFlow graph is a description of computations. To compute anything, a graph must be launched in a **Session**. A **Session** places the graph ops onto **Devices**, such as CPUs or GPUs, and provides methods to execute them. These methods return tensors produced by ops as numpy `ndarray` objects in Python, and as `tensorflow::Tensor` instances in C and C++.

The computation graph

TensorFlow programs are usually structured into a construction phase, that assembles a graph, and an execution phase that uses a session to execute ops in the graph.

Deep neural networks are making significant strides in understanding: In speech, vision, language, search, …

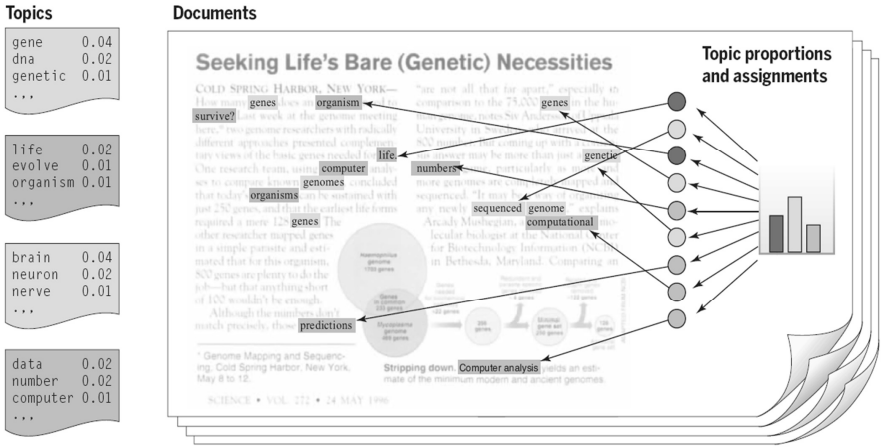
If you're not considering how to use deep neural nets to solve your vision or understanding problems, **you almost certainly should be**

Pre-trained models or pre-trained APIs are a low overhead way of starting to explore

TensorFlow makes it easy for everyone to experiment with these techniques

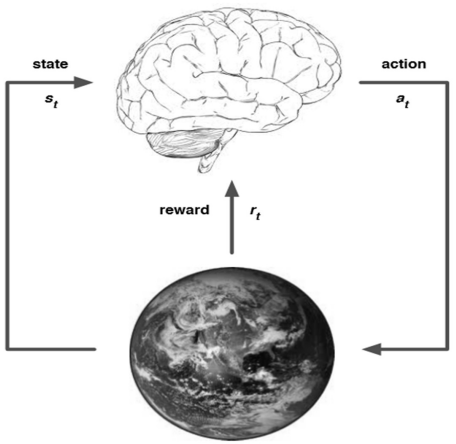
- Highly scalable design allows faster experiments, accelerates research
- Easy to share models and to publish code to give reproducible results
- Ability to go from research to production within same system

Topic Model for Analyzing Documents



Topic modeling is a methodology for analyzing documents, where a document is viewed as a collection of words, and the words are viewed as being generated by an underlying set of topics (denoted by the colors). Topics are probability distributions across words (leftmost column), and each document is characterized by a probability distribution across topics (histogram).

The Concept of Reinforcement Learning

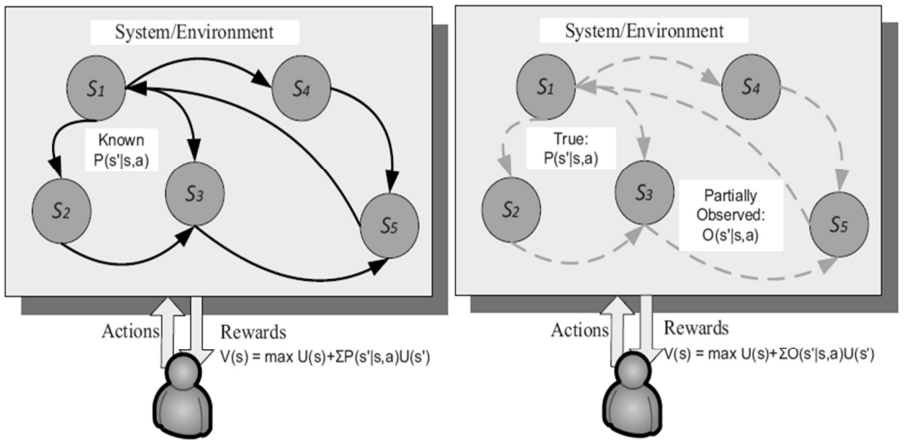


- At each step t the agent:
  - Receives state s_t
  - Receives scalar reward r_t
  - Executes action a_t
- The environment:
  - Receives action a_t
  - Emits state s_t
  - Emits scalar reward r_t

DeepMind AlphaGo Program

- AlphaGo program defeated the world Go Champaign in March 2016. The Go is the most complicated chase-board game played on a 19 × 19 grid by two players placing black and white stones on the board alternatively.
- There are 361 board points that can be placed. The whole game may end up with 10^{170} possible choices for the player to consider.
- The AlphaGo AI program was developed by DeepMind, an AI subsidiary under the Alphabet Company. This milestone achievement marked the era that machine intelligence can beat human players in some selected areas.

Reinforcement Learning



(a) Markov decision process. (b) Partially observed Markov decision process.

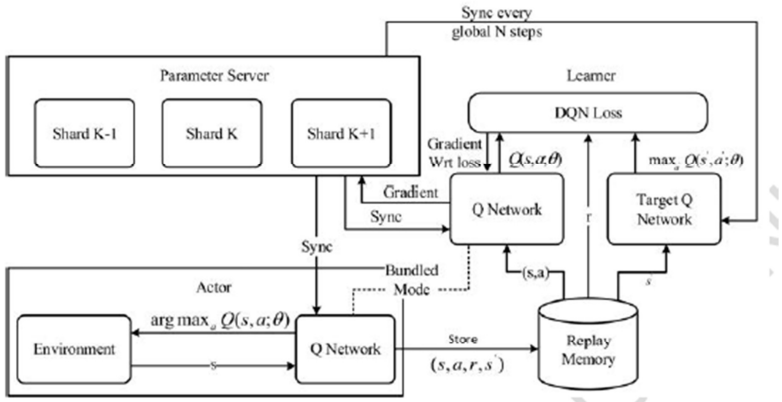


Figure 9.14
The Gorila architecture for implementing the Google reinforcement learning system. (Courtesy of David Silver, "Deep Reinforcement Learning," http://www0.cs.ucl.ac.uk/staff/d.silver/web/Resources_files/deep_rl.pdf, May 8, 2015.)

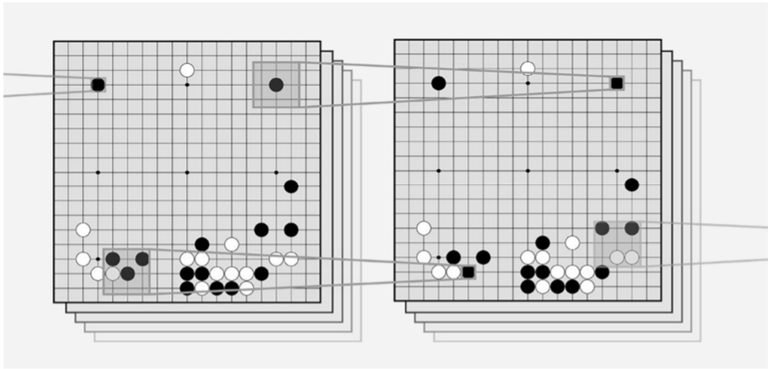
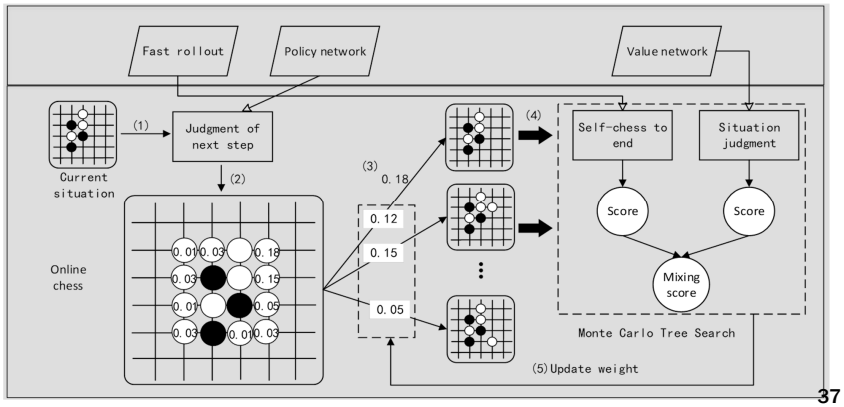


Figure 9.13: Convolutional neural network for processing the Go playing board.

DeepMind and Brain Projects at Google

- **In 2016, Google AlphaGo program defeated the top Go player. This opened up the debate between human intelligence vs. machine intelligence.**
- **The Google Brain Team has used large CPU/GPU/TPU clusters to recognize 2000 classes of photo images trained from billions of YouTube images.**



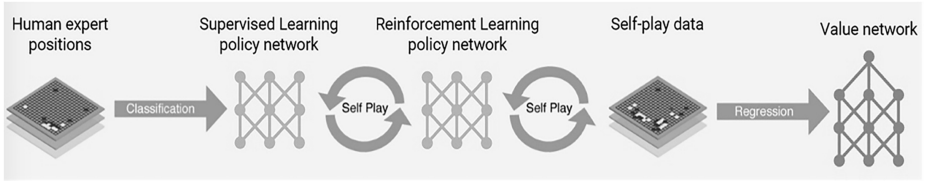


Figure 9.15 The self-play training pipeline between policy network and value networks by human experts.

Program	Accuracy
Human 6-dan	~ 52%
12-Layer ConvNet	55%
8-Layer ConvNet*	44%
Prior state-of-the-art	31-39%

Program	Winning rate
GnuGo	97%
MoGo (100k)	46%
Pachi (10k)	47%
Pachi (100k)	11%

Figure 9.17 Performance of different AlphaGo programs.

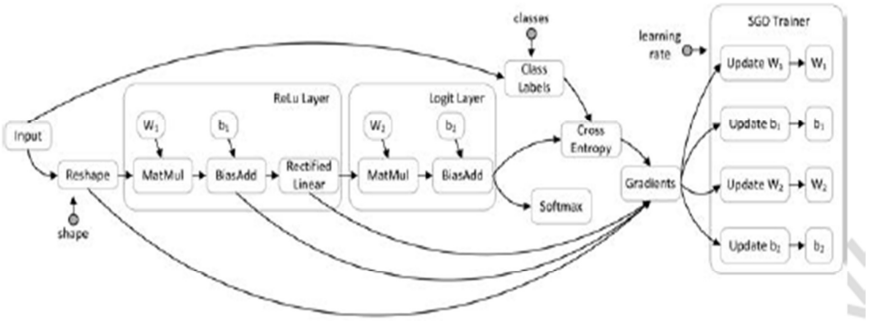


Figure 9.10
The image recognition model expressed as a TensorFlow graph. (Courtesy of Jeff Dean, Google Brain Team.)

Figure 9.11
Sample image tested through the TensorFlow image recognition system at Google.

Predictive Analytics Methods

- Linear Regression
- Logistic Regression
- Decision Trees
- Artificial Neural Networks (ANN)
- Support Vector Machines (SVM)
- Multiclass Classification

Example Analytics Applications:

- Credit Risk Modeling
- Fraud Detection
- Net Lift Response Modeling
- Churn Prediction
- Recommender Systems
- Web Analytics
- Social Media Analytics
- Business Process Analytics

Table 9.3
Top five commercial predictive analytics software systems

Software Name	Functionality and Application Domains
IBM Predictive Analytics	Predictive analytics portfolio from IBM includes SPSS Modeler, Analytical Decision Management, Social Media Analytics, SPSS Data Collection, Statistics, Analytic Server, and Analytic Answers.
SAS Predictive Analytics	SAS supports predictive, descriptive modeling, data mining, text analytics, forecasting, optimization, simulation, and experimental design.
SAP Predictive Analytics	SAP Predictive Analytics software works with the existing data environment as well as with the SAP BusinessObjects BI platform to mine and analyze the business data, anticipate business changes, and drive smarter and more strategic decision making.
GraphLab Create	A machine learning platform from Dato that enables data scientists and app developers to easily create intelligent apps at scale.
Predixion	The first cloud-based predictive modeling platform released in 2010. It supports end-to-end predictive analytics capabilities, from data shaping to deployment. Models evolved from machine learning libraries by Microsoft SQL Server Analysis Services, R, and Apache Mahout.

Source: https://www.predictiveanalyticstoday.com/what-is-predictive-analytics/

- ***Keras : A High-Level Deep Learning Library:***
- **Keras is a high-level neural networks library, written in Python. The system is capable of running on top of either TensorFlow or Theano. The Keras developer focuses on enabling fast experimentation in deep learning with various types of deep neural networks.**
- **The library allows for easy and fast prototyping through total modularity, minimalism, and extensibility. Specifically, it supports both convolutional networks and recurrent networks, as well as combinations of the two most imporatnt neural networks.**
- **Arbitrary connectivity schemes are supported, including multi-input and multi-output training. The system runs seamlessly on CPU and GPU. Listed below are guiding principles in using the Keras library**
- **Modularity. A model is understood as a sequence or a graph of standalone, fully-configurable modules that can be plugged together with as little restrictions as possible. In particular, neural layers, cost functions, optimizers, initialization schemes, activation functions, regularization schemes are all standalone modules that you can combine to create new models.**
- **Minimalism. Each module should be kept short and simple. Every piece of code should be transparent upon first reading. No black magic: it hurts iteration speed and ability to innovate.**
- **Easy extensibility. New modules are dead simple to add (as new classes and functions), and existing modules provide ample examples. To be able to easily create new modules allows for total expressiveness, making Keras suitable for advanced research.**
- **Work with Python. No separate models configuration files in a declarative format. Models are described in Python code, which is compact, easier to debug, and allows for ease of extensibility.**

•To Create a TensorFlow Cluster: **A TensorFlow "cluster" is a set of "tasks" that participate in the distributed execution of a TensorFlow graph. Each task is associated with a TensorFlow "server", which contains a "master" that can be used to create sessions, and a "worker" that executes operations in the graph.**

•A cluster can also be divided into one or more "jobs", where each job contains one or more tasks. To create a cluster, you start one TensorFlow server per task in the cluster. Each task typically runs on a different machine, but you can run multiple tasks on the same machine (e.g. to control different GPU devices). The following actions are taking place in a clutster.

- **Create a tf.train.ClusterSpec that describes all of the tasks in the cluster. This should be the same for each task.**
- **Create a tf.train.Server, passing the tf.train.ClusterSpec to the constructor, and identifying the local task with a job name and task index.**
- **Create a tf.train.Server instance in each task. This instance contains local devices, connections to other tasks in its tf.train.ClusterSpec, and a session target to perform the distributed computation.**
- **Each server is a member of a specific job and has a task index within that job. All servers can communicate with each other in the cluster.**

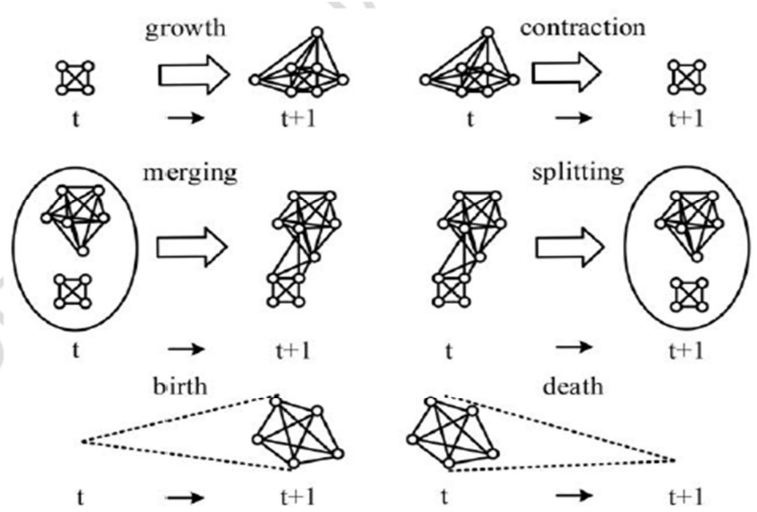


Figure 9.22
Community graph operations: birth, death, growth, merging, splitting, and contraction.

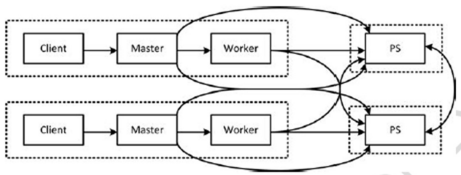


Figure 9.7
Between-graph replication for distributed Tensor execution.

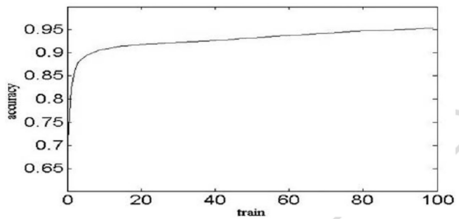


Figure 9.8
Results of TensorFlow based on programming an artificial neural network.

Final Remarks :

- **Deep Learning is now supported by both hardware chips, and devices, plust software platforms for both cloud and edge computing applications**
- **We must use clouds and big-data analytics in storing, processing, and mining of big data, which changes rapidly in time and space.**
- **Machine intelligence , clouds, IoT and social networks are being integrated together to promote global economy, public healthcare, smart cities and environments.**