TENSORFLOW: LARGE-SCALE MACHINE LEARNING ON HETEROGENEOUS DISTRIBUTED SYSTEMS

Martin Abadi et al
Recap

Deep Learning
- Broader set of Machine Learning methods and Neural Network variants.
- Based on learning data representation.
- It can be supervised, unsupervised or semi-supervised.
- Examples Convolutional Neural Network, Recurrent Neural Network, Hybrid Networks

Deep Learning Applications

Computer Vision
- Image Recognition
- Object detection
- Face recognition
- Image Caption Generation

Deep Learning Applications

Exciting Applications

- Automatic hand writing generation
- Coloring Black & White images

Deep Learning Applications

Speech Analysis
- Speech Recognition
- Voice search and Voice activated assistants
- Adding sounds to silent movies
- Healthcare using Speech

Source: https://www.maorums.com/2017/10/18/apple-explains-how-hey-siri-works/
Deep Learning Applications

Natural Language Processing
- Language Translation
- Language Understanding
- Text Summarization
- Text Categorization
- Paraphrase detection
- And more

Deep Learning Applications

Cross Domain Applications

- Automatic Call Centers
- Virtual Assistants
- Restoring sound in videos
- Robotics
- Playing games

Source: https://www.w3livenews.com/News/ReadArticle
I HAVE DL APPLICATION IDEA

WHAT SHOULD I DO
THIS IS INSANE

Derivatives
- Image analysis
- Image feature extraction methods
- Speech Analysis
- Speech Feature extraction methods
- Natural Language Processing

Probabilities
- Deep Learning models
- Training Algorithms
- Optimization Methods
- Feature Engineering
- Distributed Computation

Linear Algebra
GPU programming

Source: imgflip.com
Deep Learning frameworks

Deep Learning frameworks are used to build, train, and test deep learning frameworks.

- DL frameworks use high-level interfaces.
- Provide acceleration using GPU, TPU-like devices.
- Abstract complex mathematics.
- Inbuilt standard feature extraction methods.
- Support to design new layers.
Deep Learning Frameworks 2017

Source: https://towardsdatascience.com/battle-of-the-deep-learning-frameworks-part-one-3841750
GAME OF FRAMEWORKS

THEANO
GAME OF FRAMEWORKS

THEANO

PYTORCH

TF

KERAS

CAFFE

CNTK

Source: imgflip.com
TensorFlow

“TensorFlow is an interface for expressing machine learning algorithms and an implementation for executing such algorithms.”

TensorFlow is supported by Google

TensorFlow has interface for all major programming languages

- Provides stable Python and C API
- C++, Java, Go and Swift API without backward compatibility
- Third party interface for C#, Julia, R, Scala, Haskell, Rust and OCaml

Source: https://en.wikipedia.org/wiki/TensorFlow
Key Features

- Efficiently works with mathematical expressions involving multi-dimensional arrays
- Good support of deep neural networks and machine learning concepts
- GPU/CPU computing where the same code can be executed on both architectures
- High scalability of computation across machines and huge data sets
- TensorBoard for the effective data visualization of network modeling and performance.

Source: https://www.toptal.com/machine-learning/tensorflow-machine-learning-tutorial
Google Brain project started in 2011 to explore the usage of very large scale deep neural networks.

DistBelief was built, a scalable distributed training and interface system.

DistBelief was used by various research teams for their work in language representation, unsupervised learning, image classification and object detection etc.

Using the experiences of DistBelief and its working was used to build second generation Deep Learning Framework, TensorFlow.
Basic Concepts

- Data Flow Graphs
- Nodes
- Tensors
- Operations and Kernels
- Sessions
- Devices
Dataflow Graphs

Dataflow is a common programming model for parallel computing. In a dataflow graph, the nodes represent units of computation, and the edges represent the data consumed or produced by a computation.

Why???

- Parallelism
- Distributed execution
- Compilation
- Portability

Source: https://www.tensorflow.org/guide/graphs
Nodes

- Nodes represent a single unit of computation in data flow graphs.
- Each node has zero or more inputs and outputs.
- It represents one instant of operation.

Tensors

- Values that flow along normal edges in the graph are known as tensors.
- Tensors are typed, multi-dimensional arrays.
- Supported types are signed, unsigned integers ranging from size 8 bits to 64 bits, IEEE float, double, complex numbers, and string type.
Operations and Kernels

- Operations has a name and represents an abstract computation (e.g. Matrix multiplication or add).
- Operations can have attributes that can given or must be inferred at computation time.
- Kernel is the particular implementation of an operation that can be run on the particular device (CPU or GPU).
- Several sets of operations are defined in the TensorFlow library.
<table>
<thead>
<tr>
<th>Category</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>Element-wise mathematical operation</td>
<td>Add, Sub, Mul, Div, Exp, Log, Greater, Less, ...</td>
</tr>
<tr>
<td>Array operations</td>
<td>Concat, Slice, Split, Constant, Rank, ...</td>
</tr>
<tr>
<td>Matrix operations</td>
<td>MatMul, MatrixInverse, MatrixDeterminant, ...</td>
</tr>
<tr>
<td>Neural-net building blocks</td>
<td>SoftMax, Sigmoid, ReLU, Convolution2D, MaxPool, ...</td>
</tr>
<tr>
<td>Checkpointing operations</td>
<td>Save, Restore</td>
</tr>
</tbody>
</table>
Sessions

- Clients programs interacts with TensorFlow system by creating a Session.
- Computational Graphs can be creates using Tf.Graph() interface or can be expanded later.
- Session interface provides run function to execute the graph.
- Input and output nodes are provided in run function.
- TensorFlow implementation finds which nodes should be executed in order to compute output.
- In most uses session is created once, graph is executed multiple times using run function.
Devices

- Devices are computational heart of TensorFlow.
- Each device has device ID and device name.
- TensorFlow provides implementation for the CPU and GPU device types.
- New device implementation can be provided by using registration.
- Each device is responsible for the managing memory like allocation and deallocation, execution of kernels.
import tensorflow as tf

b = tf.Variable(tf.zeros([[100]]))  # 100-d vector, init 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 = [...]

s = tf.Session()
for step in xrange(0, 10):
    input = ...construct 100-D input array ...
    result = s.run(C, feed_dict={x: input})  # Create 100-d vector for input
    print step, result  # Fetch cost, feeding x=input
```python
import tensorflow as tf

b = tf.Variable(tf.zeros([100]))  # 100-d vector, init 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 ...
    result = s.run(C, feed_dict={x: input})  # Create 100-d vector for input
    print step, result  # Fetch cost, feeding x=input
```
Implementation

In TensorFlow system, client uses Session interface to communicate with master and worker processes.

Each Worker is responsible for one or more devices.

TensorFlow provides two types of execution:

- Single Device Execution
- Multi-Device Execution
Single Device Execution

- This is simplest execution scenario: a single worker process with single device
- Nodes are executed according to their dependencies
- Count is maintained with respect to each node that denotes how many dependencies are left
- Once dependencies drops to zero, node added to the queue for the execution.
- When node is executed, counters of all nodes dependent on executed node is decreased by 1.
Multi Device Execution

There are two main complications when running TensorFlow on the multi devices.

- Deciding devices for computing each nodes.
- Communication of data across devices implied by the placement decisions.
Node Placement

- Mapping of computation on the available set of devices.
- Cost model is input to the node placement algorithm.
- Cost model contains estimates of the input, output tensors and computational time required by the each node.
- These cost are estimated statistically or measured based on an actual set of placement decision for earlier execution.
- Placement algorithm first runs a simulated execution of the graph.
Dataflow graph is given

For given node find the feasible devices

If multiple feasible device, find the cost of each feasible device

Select device based on the greedy heuristic.

Move to next node and repeat the process.
Cross Device Communication

- Graph is partitioned into a set of subgraphs one per device.
- Edges between the partitions are removed.
- Edges are replaced by send and receive nodes.
Extension

These are some advance features which are available in TensorFlow:
- Automatic gradient computation
- Partial Execution
- Device Constraints
- Control flow
- Input operations
- Queues
- Containers
Optimizations

Some of the optimizations implemented in the TensorFlow:

- Common Subexpression Elimination
- Controlling Data Communication and Memory Usage
- Asynchronous Kernels
- Optimized Libraries for Kernel Implementations
- Lossy Compressions
## Status and Experience

<table>
<thead>
<tr>
<th>Build</th>
<th>Build tools to gain insight into the exact number of parameters.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start</td>
<td>Start small and scale up.</td>
</tr>
<tr>
<td>Ensure</td>
<td>Always ensure that the objective function matches between machine learning systems.</td>
</tr>
<tr>
<td>Make</td>
<td>Make a single machine implementations match before debugging distributed implementations.</td>
</tr>
<tr>
<td>Guard</td>
<td>Guard against numerical errors.</td>
</tr>
<tr>
<td>Analyze</td>
<td>Analyze piece of a network and understand magnitude of error.</td>
</tr>
</tbody>
</table>
DATA PARALLEL TRAINING

Computation of the gradient for a mini-batch across mini-batch elements.
Model Parallel Training

Different portions of the model computation are done on different computational devices simultaneously.
TensorBoard (model visualization)
TensorBoard (Data visualization)
ARE YOU SERIOUS

TENSORFLOW IS NOT EASY
DON'T WORRY

KERAS IS HERE
Keras

- Keras is a high level neural network library written in Python.
- Capable of running top on TensorFlow, Theano or CNTK.
- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Source: https://keras.io/
from keras.models import Sequential
model = Sequential()

from keras.layers import Dense
model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))

model.compile(loss='categorical_crossentropy',
              optimizer='sgd',
              metrics=['accuracy'])

# x_train and y_train are Numpy arrays -- just like in the Scikit-Learn API.
model.fit(x_train, y_train, epochs=5, batch_size=32)

Source: https://keras.io/
DISTRIBUTED TENSORFLOW WITH MPI

A. Vishnu, C. Siegel, and J. Daily
Message Passing Interface (MPI)

- MPI provides rich set of abstractions for inter-process communication.
- MPI supports pair wise and group communications.
- The primary reason for the MPI success is its wide availability.
- All to all reduce function of the MPI is used to disseminates the final result across all nodes.
- Point to point operations for data distribution.
- Spark is used as programming model with MPI.
Proposed Design and Implementation

TensorFlow backend is used as a Blackbox and its primitives are leveraged to support distributed memory execution.

Following ideas are considered:
- Work Distribution
- Data Parallelism
- Synchronous/Asynchronous updates
Work Distribution

- Firstly samples are distributed across all TensorFlow devices equally.
- It is possible to split samples unequally but in this implementation all devices are considered equal.
- Default process reads the samples from the disk and slits them.
- Current implementation is not optimized for parallel reading.
- This is not a issue as most of the time is spent in the training.
Data Parallelism

Few possibilities are
- Matrices belonging to each layer being distributed across multiple compute nodes.
- Each worker can update weights on a parameter server asynchronously.
- Splitting the data graph across multiple compute nodes.
- Model is replicated on each device.
Synchronous/ Asynchronous updates

- Synchronous weight updates is implemented.
- Asynchronous weight updates has advantages but it is hard to verify training algorithm correctness.
- Synchronous weight updates scales with MPI and high performance interconnects.
- Averaging operation for the synchronized weight update is highly optimized in the MPI library.
Experimental Evaluation

- Each machine in the system consists of a multi core Intel Haswell CPU and 64GB RAM.
- Machines are interconnected using infiband.
- OpenMPI 1.8.3 is used for performance evaluations.

<table>
<thead>
<tr>
<th>Data Set</th>
<th>ALGO</th>
<th>Network Architecture</th>
</tr>
</thead>
<tbody>
<tr>
<td>ADULT</td>
<td>DNN</td>
<td>123-200-100-2</td>
</tr>
<tr>
<td>ACOUSTIC</td>
<td>DNN</td>
<td>50-200-100-3</td>
</tr>
<tr>
<td>MNIST</td>
<td>DNN</td>
<td>784-200-100-10</td>
</tr>
<tr>
<td>MNIST</td>
<td>CNN</td>
<td>32,64 (CONV), 1024 (FULL)</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>DNN</td>
<td>3072-200-100-10</td>
</tr>
<tr>
<td>CIFAR10</td>
<td>CNN</td>
<td>32,64 (CONV), 1024 (FULL)</td>
</tr>
<tr>
<td>HIGGS</td>
<td>DNN</td>
<td>28-1024-2</td>
</tr>
</tbody>
</table>
MNIST

Relative speedup to 1-core on MNIST-DNN using up to 32 cores

Relative speedup to 16-core on MNIST-CNN using up to 64 cores
CIFAR10

Relative speedup to 4-core on CIFAR10-CNN using up to 64 cores
Relative speedup to 5-core on Adult-DNN using up to 40 cores
Acoustic

Relative speedup to 1-core on Acoustic-DNN using up to 40 cores
Questions
Thank You