TensorFlow: A System for Large-Scale Machine Learning

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng.

Conor Perreault

Predecessors to TensorFlow

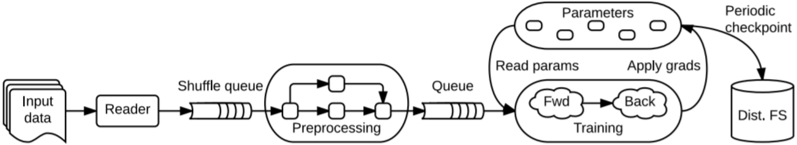
- **MapReduce:** Early batch data processing system, TensorFlow extends to more complex algorithms.
  - MapReduce used reactive backup workers, TensorFlow uses backup workers proactively.
- **Naiad:** Single optimized static data flow graph and some control flow (looping, branching) in TensorFlow are similar to Naiad.
- **DistBelief:** Google's earlier machine learning system.
  - Parameter server architecture from DistBelief is largely used again in TensorFlow.
  - TensorFlow is more flexible and faster on a variety of problems.

Major Accomplishments

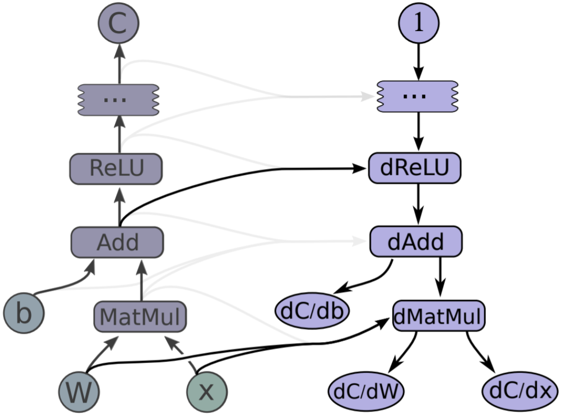
- Scalability across heterogeneous devices
  - Separating workload distribution from the implementation of computations allows the same node of data flow graph to be executed on a variety of different kernels/devices.
- Performance on a single device and distributed systems
  - Compared to DistBelief, TensorFlow scales *down* much better, while maintaining and improving large system performance
- Flexibility for new NN layers and optimization algorithms
  - Abstraction of computations allows new algorithms to be swapped in easily.

Data Flow Graph

- Nodes represent an operation
- Edges represent a tensor of data that will flow between nodes
- Variables can be used to store global data, e.g. current model parameters
- Queues also implement state, and help synchronize operations such as data fetching and computations
- Any subgraph can be executed by itself since data flow is separated



Data Flow Graph Generation



Distributed Workload

- **Client:** programmer interface that allows the graph, or a subgraph to be run.
- **Master:** main organizational process. Does not handle scheduling, this is implemented by blocking queues.
- **Worker Processes:** processes responsible for handling node computations on any of the hardware devices available.

Basic Data Flow

- With a single device, nodes are put in a ready queue when the number of unexecuted dependencies reaches 0.
- Using heuristics and a greedy algorithm, the algorithm simulates the graph execution and assigns nodes to different devices when multiple are available.
- On a single machine, TensorFlow has adequate training speed. One of the main challenges the authors faced was the ability to scale up and down.

Library	Training step time (ms)			
	AlexNet	Overfeat	OxfordNet	GoogleNet
Caffe [38]	324	823	1068	1935
Neon [58]	87	**211**	**320**	**270**
Torch [17]	**81**	268	529	470
TensorFlow	**81**	279	540	445

Distributed Data Flow

- Greedy algorithm incorporates time to send data between machines. Tensors are sent between machines only once even if multiple nodes use the same data.
- User-level checkpointing achieves fault tolerance which is necessary when training for a long time period on a cluster.
- Asynchronous training uses stale parameters, so TensorFlow begins to use synchronous training as well using blocking queues. Backup workers are required to mitigate the effect of lagging processes.
- Backup workers train on different data to take advantage of SGD batching. Reduces the effect of a lagging node, number of workers is optimized.

Major Benefits

- Much easier to experiment than previous systems.
- Abstracted computation kernels is a clever way to solve the heterogeneous device problem.
- Ability to scale up and scale down by balancing overhead computations is impressive.
- Backup processes are well optimized to minimize resource consumption and the effect of lagging workers.

Major Criticisms

- Static data flow graph makes reinforcement learning and recurrent neural networks difficult to implement.
- Authors stated the goal of making their ML platform more accessible, but TensorFlow still requires a lot of manual tuning.
- Too generic? TensorFlow works with a lot of languages and libraries, but doesn't fit in with any. Depending where a piece was implemented, the performance can be hard to predict. (C++ vs Python)
- Minimal comparison to other systems for large scale distributed machine learning.
- Is the balance between large and small scale abilities at the cost of being really good at either one?