**[TensorFlow: A system for large-scale machine learning]**

1. Introduction

In recent years, machine learning has led advances in various fields, and for this success, Google developed the TensorFlow system to experiment with new machine learning models and train large-scale models. With its experience with DistDelief, a first-generation system, Google was able to create TensorFlow that supports large-scale training and inference.

2. Background & motivation

a. TensorFlow is the successor to DistBelief, which uses a parameter server architecture to process neural network training, where users define neural networks with directional acyclic graphs and the learning algorithms are updated to minimize the loss function. Compared to previous systems, TensorFlow provides a single programming model and runtime system that can define new training algorithms or make efficient updates easier.

b. TensorFlow is designed to be much more flexible than DistBelief by providing simple data flow-based programming abstractions while meeting Google's production machine learning workloads. By representing mathematical operators as nodes in a data flow graph, various model structures and optimization algorithms can be experimented with. And by delaying execution until the entire program is ready, it can be optimized based on global information about the entire computation, and it provides a common abstraction for heterogeneous accelerators.

c. Various machine learning frameworks include Caffe, Theano, and Torch, each with its pros and cons. Batch DataFlow systems are used in many machine learning algorithms, but can take a long time to update. TensorFlow provides a high-level programming model to overcome the limitations of the parameter server architecture, allowing users to customize the code in all parts.

3. TensorFlow execution model

(3.1) Dataflow graph elements

Each vertex in the graph represents a local computational unit, and each edge represents the output or input of the vertex.

* Tensors: In TensorFlow, all data are modeled as tensors (multidimensional arrays). Tensors have several basic types of elements, such as int32, float32, or string.
* Operations: The operation takes m ≥ 0 input tensors and produces n ≥ 0 output tensors. The operation has a "type" (e.g., Const, MatMul, or Assign), and has a compilation time property that determines the operation.
* Stateful operations: variables: The operation owns a variable buffer that stores the shared parameters of the model during training.
* Stateful operations: queues: TensorFlow includes several queue implementations, such as FIFO Queues, which support more advanced tuning.

(3.2) Partial and concurrent execution

TensorFlow supports partial, simultaneous execution, allowing multiple graphs to run simultaneously, share status, and adjust. This provides TensorFlow flexibility and simplicity in implementing algorithms.

(3.3) Distributed execution

TensorFlow makes distributed execution simple, clear, and device-to-device communication easier through DataFlow. Operations are assigned to specific devices, and users can manually or automatically optimize operations. Once operations are deployed, TensorFlow improves execution speed by dividing them into subgraphs for each device.

(3.4) Dynamic control flow

TensorFlow supports conditional and iterative control flows, enabling the implementation of advanced algorithms such as RNNs. It uses basic operations such as Switch and Merge to handle conditional branches and iterative loops, and dynamic control flows can handle variable-length sequences.

4. Extensibility case studies

a. TensorFlow provides a library that automatically computes various optimization algorithms such as SGD, allowing users to optimize their learning by leveraging techniques such as batch normalization and gradient clipping. Furthermore, TensorFlow can experimentally use several optimization algorithms such as Momentum, AdaGrad, RMSProp, Adam, etc.

b. Variance representations are commonly used to train models with high-dimensional data. In TensorFlow, sparse vectors are multiplied by embedding matrices to transform them into dense matrix representations, and optimization algorithms modify only the rows of embedding matrices read from sparse multiplication. Instead of manually constructing graphs, users can use TensorFlow's library to generate appropriate graphs according to variance order.

c. Because model training is time consuming and can be disruptive, TensorFlow manages it with user-level checkpointing, which allows users to periodically store and restore variables in the model to prevent disruptions to the training process.

d. TensorFlow allows SGD to be trained asynchronously and synchronously, leveraging backup workers to mitigate delays.

5. Implementation

TensorFlow runtime uses C API to separate user code and runs on various platforms. Core libraries are implemented in C++ and run on a variety of devices. There are distributed masters and dataflow practitioners that generate and coordinate dataflow graphs, and it supports more than 200 standard operations such as mathematical, array manipulation, control flow, and state management. TensorFlow is used to build neural network layers, optimization algorithms, and more. It also provides tools such as serving infrastructure for inference, visualization dashboards, graph visualization tools, and distributed profilers.

6. Evaluation

This section evaluates the performance of TensorFlow on a number of synthetic and realistic workloads.

a. On single-machine benchmark, TensorFlow achieves a shorter step-by-step time than Caffe, comparable to the latest version of Torch by less than 6%.

b. In small models, synchronous training works well, but in large models, the load increases.

c. Tensorflow performs slightly better than MXNet.

d. Two implementations of softmax are used to measure the learning throughput.

7. Conclusions

TensorFlow has been widely used and applied in the field of machine learning since it was released as open software, and is steadily developing. Recently, even when the computational structure is dynamically deployed, it is facing a problem of providing a system that transparently and efficiently uses distributed resources.