TensorFlow: A System for Large-Scale Machine Learning

Presented by Raymond Xu

Goals

- Large-scale machine learning
- Experimenting, training, and productionisation
- Distributed computing and diverse machines
- Enable developers and researchers

Predecessor

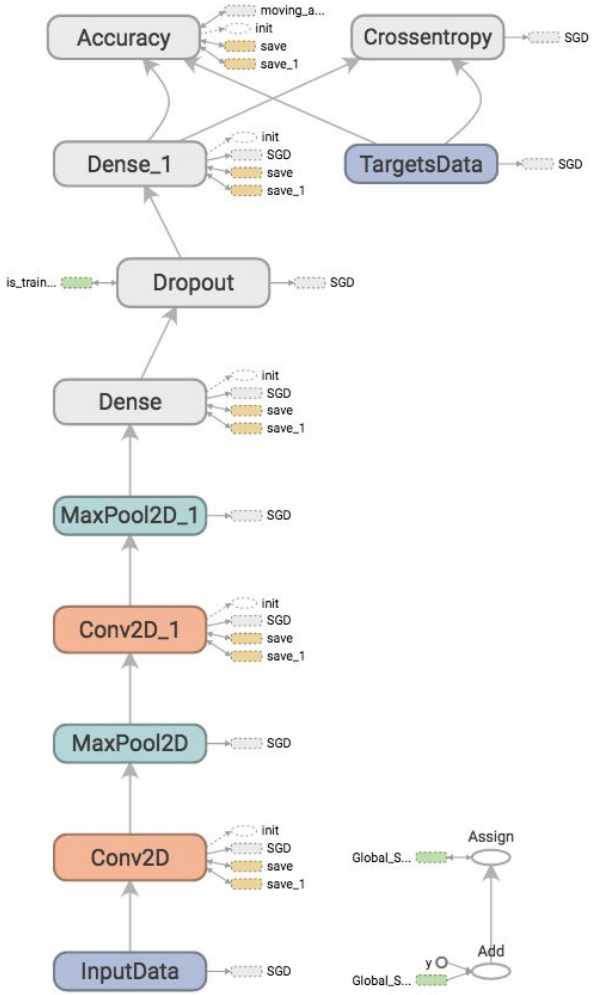
- Google’s previous system: DistBelief
- Parameter server architecture
  - Worker processes do computation
  - Parameter server maintains current version of model parameters
  - Workers write back delta updates to each parameter server
- Limitations
  - Defining new layers
    - Must do in C++
  - Refining training algorithms
    - Requires changing the parameter server implementation
  - Defining new training algorithms
    - Hardcoded feed-forward execution pattern

Tensorflow

- Dataflow graphs of primitive operators
  - vs DistBelief's few complex layers
- Deferred execution
  - Optimize execution
- Common abstraction for accelerators
  - CPU, GPU, TPU

Tensorflow Execution Model

- One dataflow graph represents all computation and state
  - Vertices: operations
  - Edges: values
- All data are tensors (n-dimensional arrays)
- Operations take m >= 0 tensors as input and produce n >= 0 tensors as output
- Variables are stateful operations with mutable buffers
- Queues allow you to stream computation between graphs


Accuracy
moving_a...
init
save
save_1
Crossentropy
SGD
Dense_1
init
SGD
save
save_1
TargetsData
SGD
is_train...
Dropout
SGD
Dense
init
SGD
save
save_1
MaxPool2D_1
SGD
Conv2D_1
init
SGD
save
save_1
MaxPool2D
SGD
Conv2D
init
SGD
save
save_1
InputData
SGD
Assign
Global_S...
Add
y
Global_S...

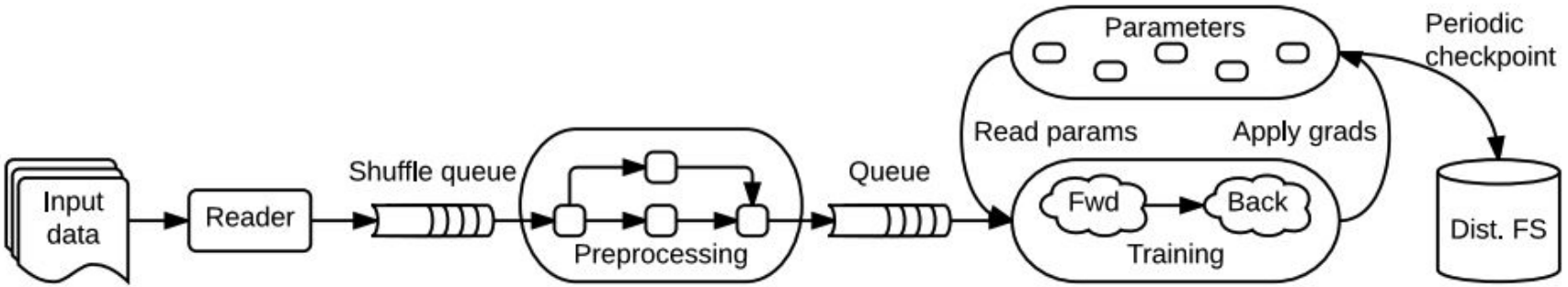


Figure 2: A schematic TensorFlow dataflow graph for a training pipeline, containing subgraphs for reading input data, preprocessing, training, and checkpointing state.

Distributed Execution

- Each operation resides on a device in a task
- Tensorflow uses a placement algorithm to place operations on devices
  - Can also manually optimize
- Per-device subgraphs contain all operations for a device
  - Edges between devices are replaced with *Send* and *Recv* operations

Fault Tolerance

- Training can take days so failure is especially bad, although unlikely
- Saving every write is wasteful
  - Data can be recomputed
  - Strong consistency is unnecessary
- Solution: User-level checkpointing
  - *Save* and *Restore* tensors from checkpoint files

Synchrony

- Tensorflow supports synchronous replication
  - Use a blocking queue
  - Use backup workers to mitigate slow workers that limit throughput
    - E.g. in SGD training data is randomly sampled so it’s trivially parallelizable
    - Improves throughput by up to 10%

Implementation

- C++ core, C API, many clients
- Distributed master takes user requests and coordinates graph partitioning and device placement of tasks
- Dataflow executor in each task handles requests from the master and schedules the execution of the kernels

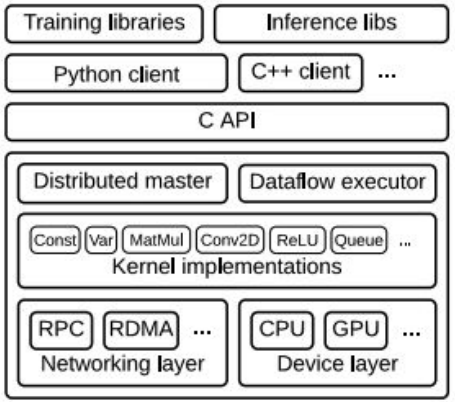


Figure 6: The layered TensorFlow architecture.

Sample Code

```
x = tf.constant([[1], [2], [3], [4]], dtype=tf.float32)
y_true = tf.constant([[0], [-1], [-2], [-3]], dtype=tf.float32)

linear_model = tf.layers.Dense(units=1)

y_pred = linear_model(x)
loss = tf.losses.mean_squared_error(labels=y_true, predictions=y_pred)

optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

init = tf.global_variables_initializer()

sess = tf.Session()
sess.run(init)
for i in range(100):
  _, loss_value = sess.run((train, loss))
  print(loss_value)

print(sess.run(y_pred))
```

TensorFlow: A System for Large-Scale Machine Learning

Presented by Raymond Xu

TFX: A TensorFlow-Based Production-Scale Machine Learning Platform

Presented by Raymond Xu

What’s wrong with just Tensorflow?

- Orchestration of components not included, often messy
- Lacking validation of data and models
- Lacking production infrastructure for serving models

Concepts

- Learner: takes a dataset and emits a learned model
  - Training phase: takes features as input and emits predictions
  - How to deploy to production?
- Machine Learning Platform: learner, model, and components for productionisation
- Challenges
  - Many different learning tasks
  - Continuous training and serving
  - Human-in-the-loop
  - Production-level reliability and scalability

Platform

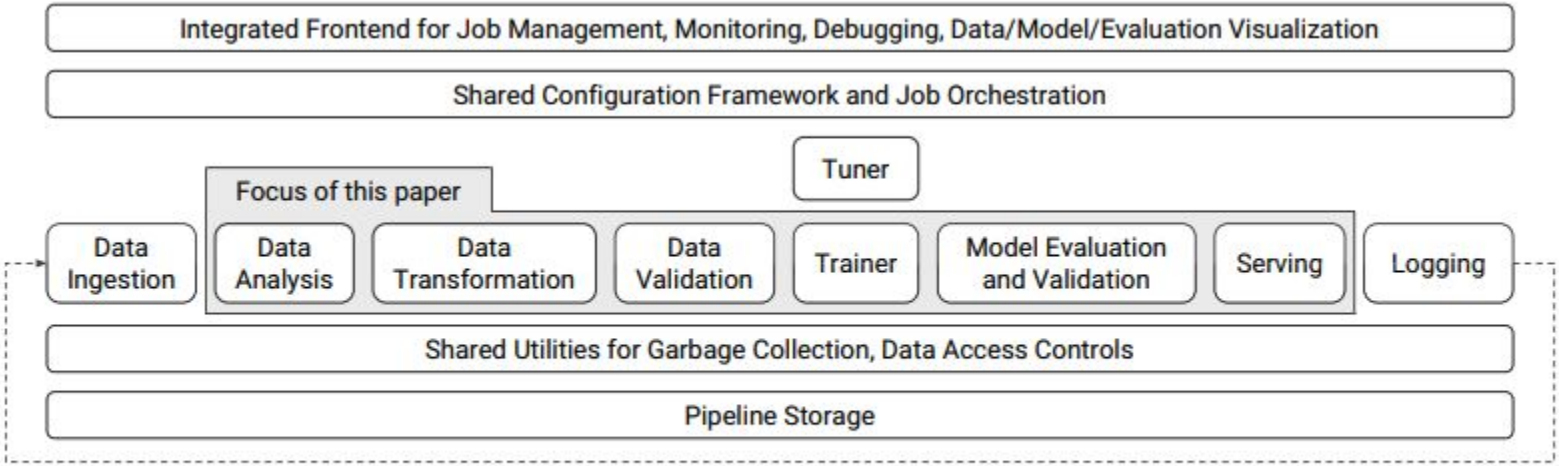


Figure 1: High-level component overview of a machine learning platform.

Data Analysis

- Collect statistics about data
  - Distributions
  - Counts
  - Histograms
  - Mean/Stdev

Data Validation

- A schema specifies expected properties of the data
- Anomalies are surfaced to the user with suggestions on how to fix them

Model Evaluation

- Evaluate models offline on held-out data to determine whether they are promising enough to start an online A/B experiment

Model Validation

1. Safe to serve: use a canary process
2. Prediction quality: compare against a fixed threshold and baseline model

Model Serving

1. To handle multitenancy, they implemented a threadpool for model-loading operations
2. To increase efficiency in handling data, they implemented a specialized protocol buffer parser

TFX: A TensorFlow-Based Production-Scale Machine Learning Platform

Presented by Raymond Xu