TensorFlow Tutorial

MSBD 5001 Week 5

Machine Learning Frameworks

- Simplifies the implementation of machine learning models
  - Imagine you write hundreds of layers by yourself?
  - Implementing back-propagation?
  - Present and analysis your results?
- Tools like TensorFlow helps you by providing high-level APIs.

Outline

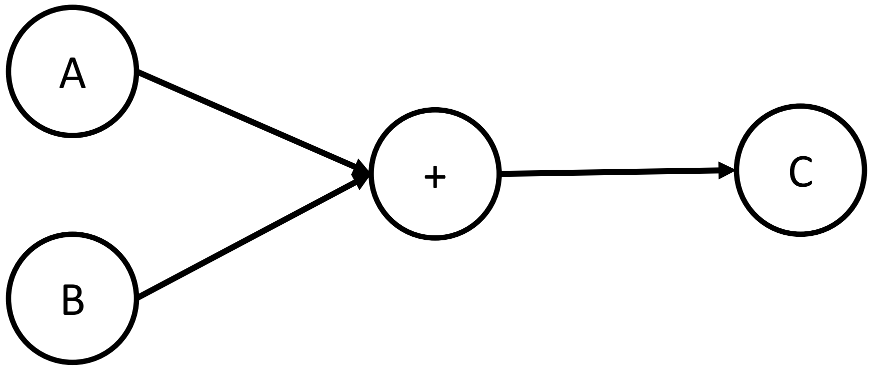
- Overview of TensorFlow computation pattern
- Building a deep learning model with TF
- Tensorboard: Result visualization
- Session saving and restoring
- Distributed TensorFlow

Computation Graphs

- TF represent computations in “Graphs”
- Nodes: Input data, output, intermediate results (“Tensor”) or computational operations, e.g., +, -
- Edges: Directions of data / intermediate results (”Flow”)

Example

- How do you represent A+B=C? A=2; B=3



- This is just how your code is interpreted, not how you write your code.
  - A=tf.Variable(2). Try to print C. What happens?

Lazy Evaluation and Session

- TensorFlow introduces “lazy evaluation”
  - Subgraph is only evaluated when a part is needed, not when you define it.
- Separate graph definition and execution
- Use of “tf.session”
  - session.run() actually tells you to run the graph
  - Initialize variables to run

Interactive Session & Eager

- Interactive Session: The concept of “current” session
- Try with the demo code to see the difference
  - In InteractiveSession(), passing the session to run the code is not required
- Eager execution: dynamic description of graph (not discussed today)

Graphs vs Neural Network

- Neural Network represents a sequence of operations on the input data
- It can be represented as a graph, as in TensorFlow

Evolution of Deep Learning Systems

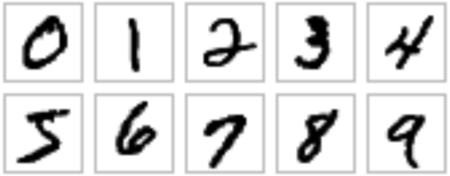
- In Google: DistBelief → TensorFlow
- Similarity: Both defined NN as a graph
- Difference:
  - TF used arithmetic operations as the minimal unit of nodes in graph
    - More flexibility for researchers to define new layers
    - DistBelief uses layers as nodes
  - TF provides more flexibility in refining (and defining) new training algorithms

More References

- If you want to read more about the design considerations of TensorFlow, you can refer to its OSDI'16 publication:
  - "TensorFlow: A system for large-scale machine learning"
  - https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf

Rest of today

- Task: Handwritten Digit classification
- Dataset: MNIST Dataset



- How do we analyze the performance?
- How do we store intermediate results?
- Introduce more TF concepts in the process.

A Simple Model

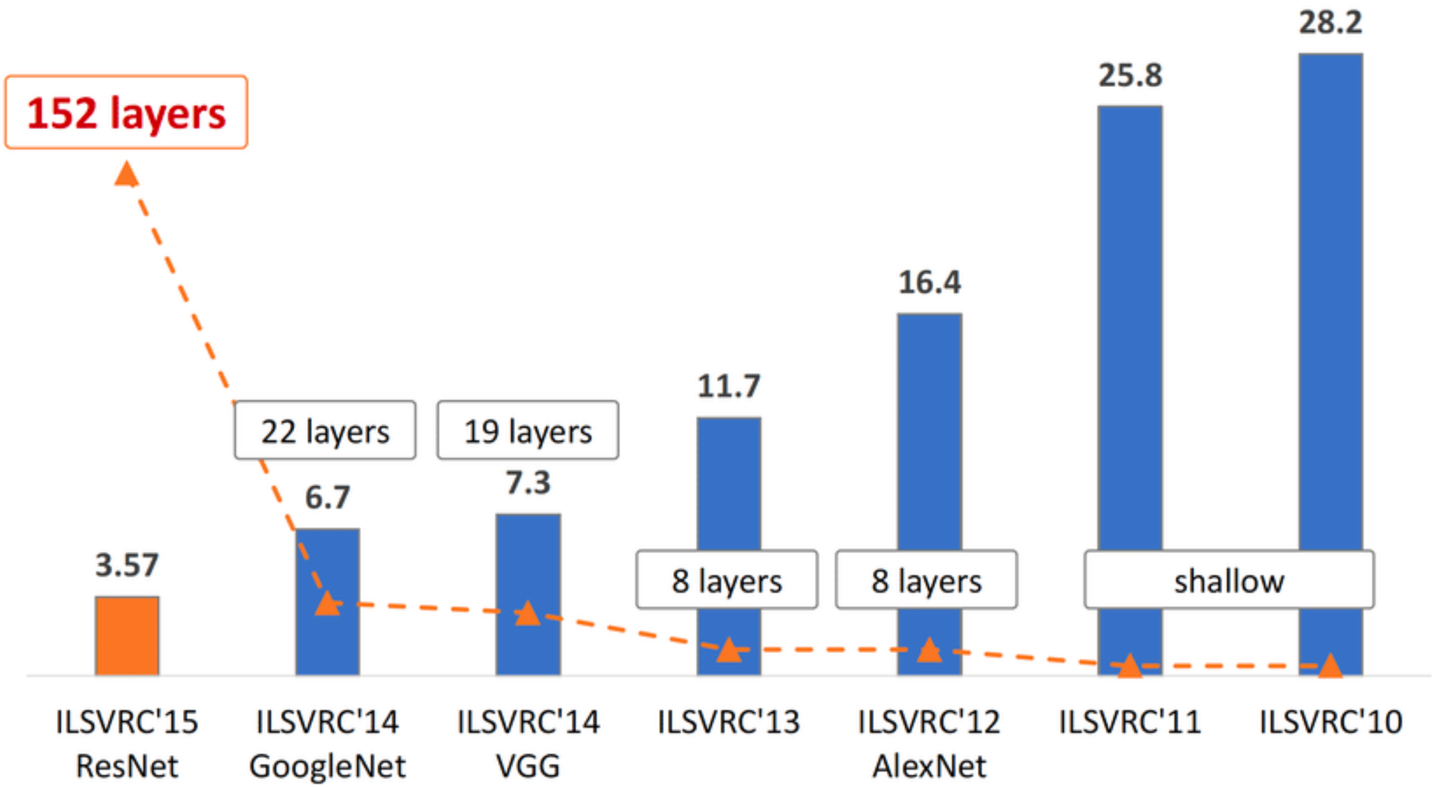
- MNIST datasets: black-white images, 28*28 size.
- In the dataset examples, the size can be represented as 1-D tensor of size 784=28*28.
- y = tf.nn.softmax(tf.matmul(x, W) + b)
  – Input (x) multiplied by weight matrix (W), and add bias vector (b), use a function called softmax to predict output probability.
- The loss is defined as cross-entropy loss (deviation from ideal probability, 1 vs all 0)
  – Please try the code yourself

Outline

- Overview of TensorFlow Computation Pattern
- Building a deep learning model with TF
- Tensorboard: Result visualization
- Session saving and restoring
- Distributed TensorFlow

The Power of Deep Models

- Typically deep models can attain higher accuracy
  - Example: ImageNet Classification winners

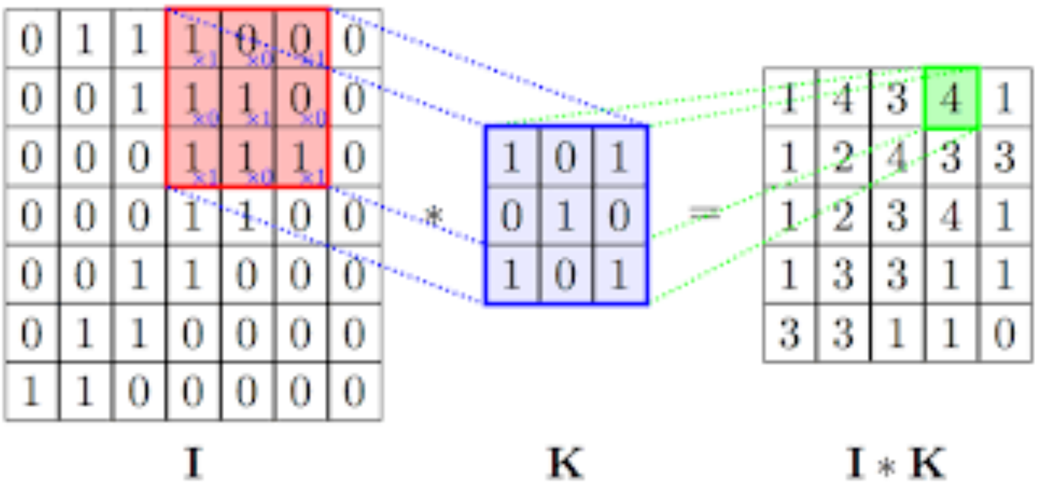


Why Deep Learning Becomes Popular?

- Availability of big data
- Computation power of training large networks (especially GPUs)
- Success in computer vision, speech recognition, and NLP
- Develop of algorithms/techniques dealing with training issues
- Etc.

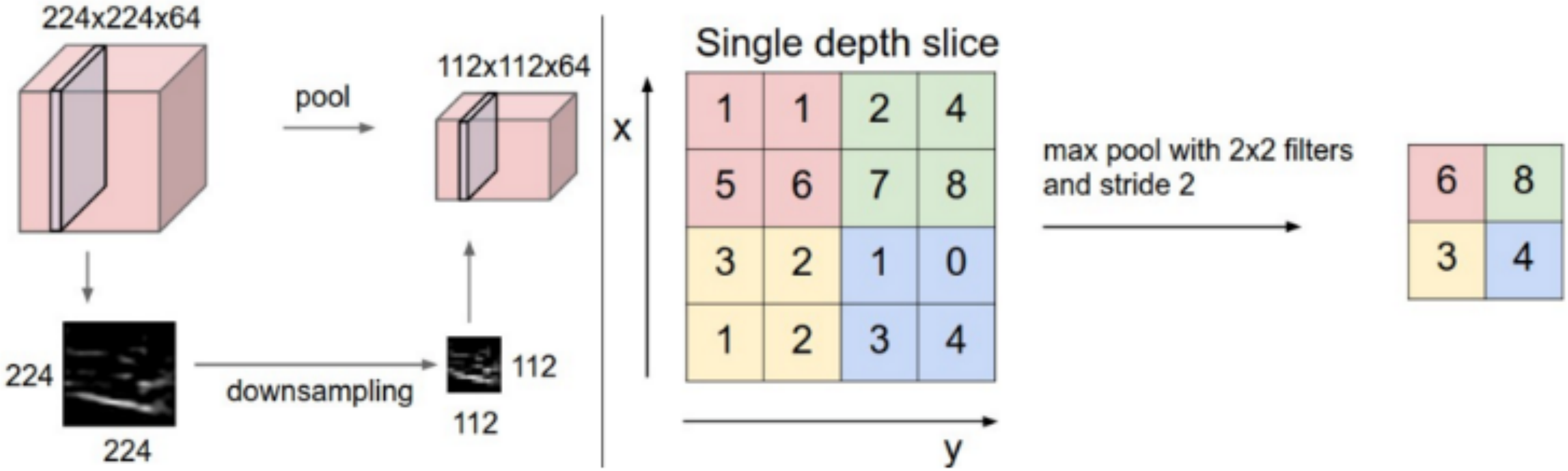
Convolution Layers

- Extracts ”Local Features”
- More time-consuming, higher accuracy.
- Output is sum of element-wise product between a input region and kernel.



Pooling Layer

- Downsampling



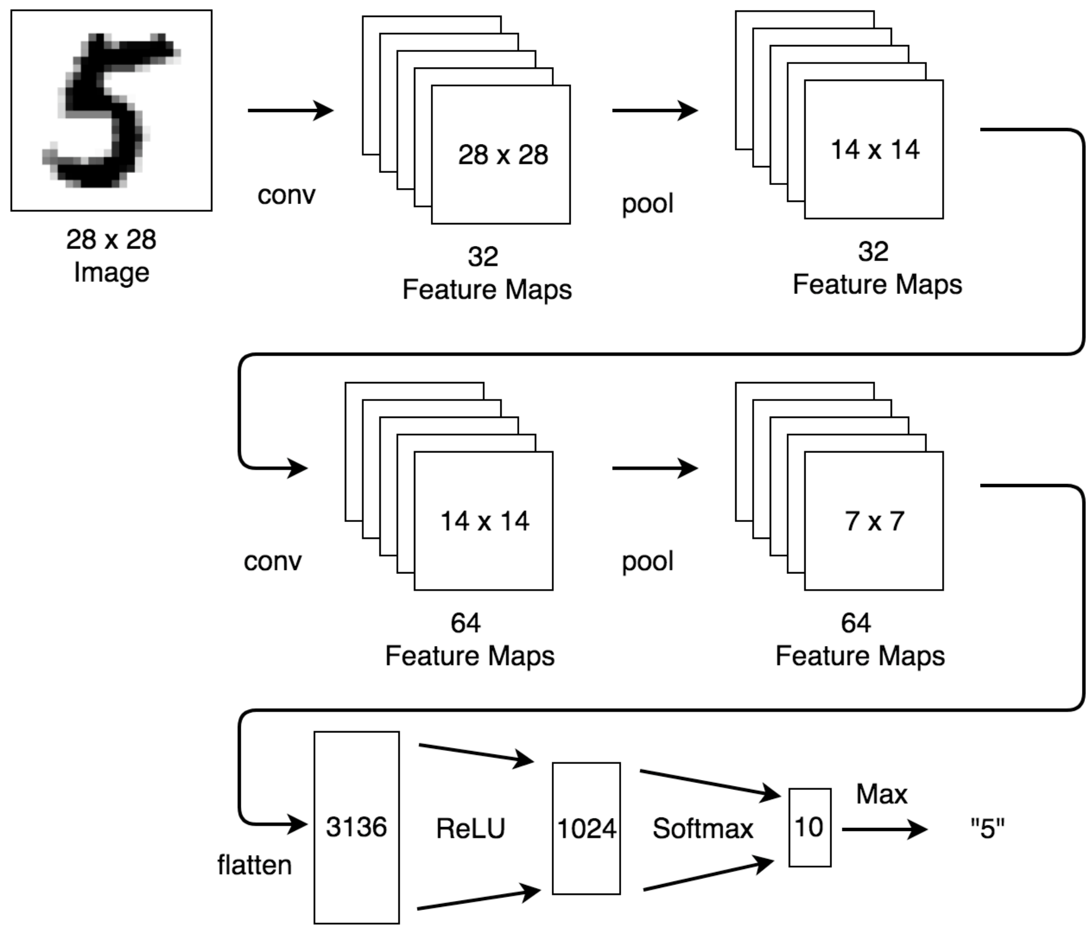
Other Layers

- Flatten Layer: Transforms high dimension tensor to 1D tensor
- Fully-connected Layer: Represents a linear transformation

Build a CNN for MNIST

- We suggest you to use the following structure:
  - Convolution 1: kernel size 5*5, input channel 1, output channel 32 + Max pooling 2*2, padding = “same”
  - Convolution 2: kernel size 5*5, input channel 32, output channel 64 + Max pooling 2*2, padding = “same”
  - Flatten to 1D + Fully connected layer 3136 (dropout) → 1024 → 10
- We first try implementing every step, then use tf.layers API.

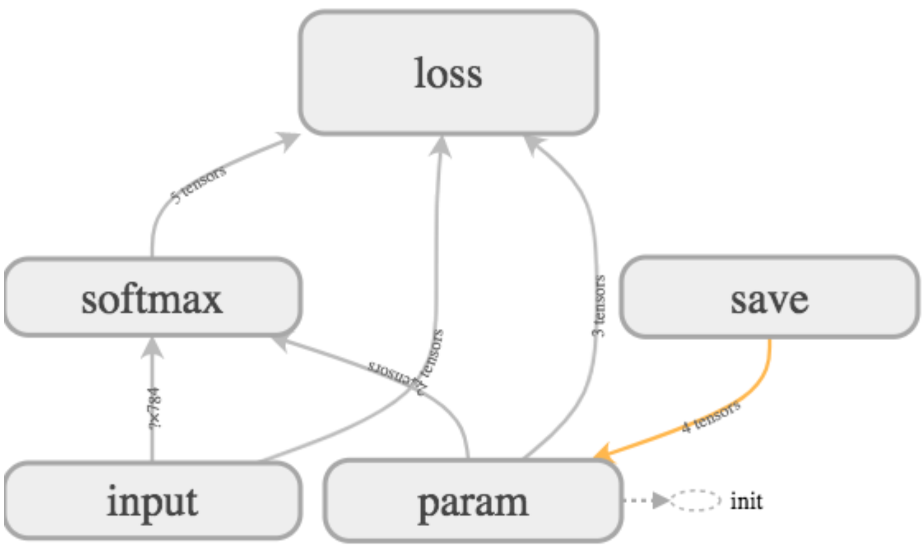
Visualized structures



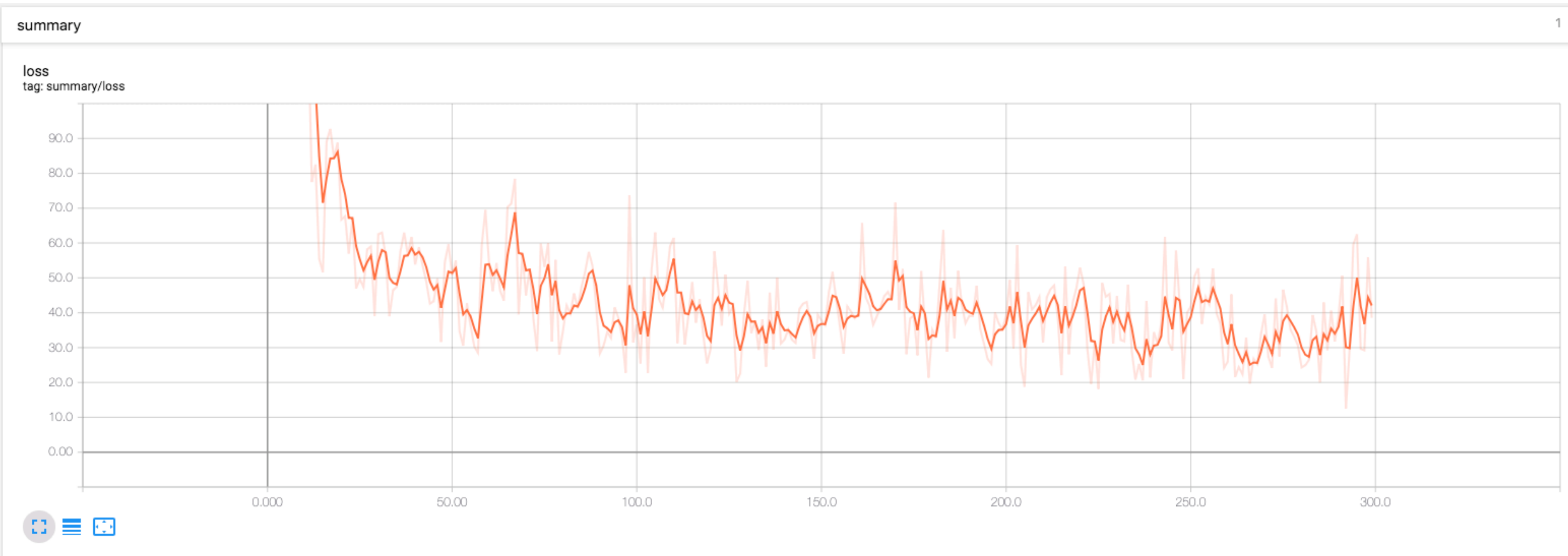
Outline

- Overview of TensorFlow Computation Pattern
- Building a deep learning model with TF
- Tensorboard: Result visualization
- Session saving and restoring
- Distributed TensorFlow

Visualize the Graph



Visualize Certain Scalar



Add Summaries

- Use command tf.summary.<summary to add>()
  - Scalars (with global step size)
  - Histograms

- Initialize a writer, write the summaries to the result.
- Exercise: Visualize your results using TensorBoard with following learning rate:
  - 0.1,0.01, 0.001

Define Your Nodes in Graph

- Use namescope command
  - Merge multiple nodes into one node
  - If you click the node, you can still view its internal structure
- We will show on TensorBoard what's the difference

Outline

- Overview of TensorFlow Computation Pattern
- Building a deep learning model with TF
- Tensorboard: Result visualization
- Session saving and restoring
- Distributed TensorFlow

Saving and Reusing Sessions

- You may want to save your session occasionally
  - Backup for fault-tolerance
  - Save result for later uses, no need to start to train again.
- At its core, saving session is saving the variable values on the graph.
- Try save() and restore() methods
- Exercise: First save your model, and predict directly using a previous result

Outline

- Overview of TensorFlow Computation Pattern
- Building a deep learning model with TF
- Tensorboard: Result visualization
- Session saving and restoring
- Distributed TensorFlow

Distributed Learning

- Additional topic today: No more coding
- Problem: Learning a model on a local machine can be time-consuming sometimes.
  - Hours to days
- Intuitive solution: Use multiple machines to learn together
- Challenge: How to sync the results?

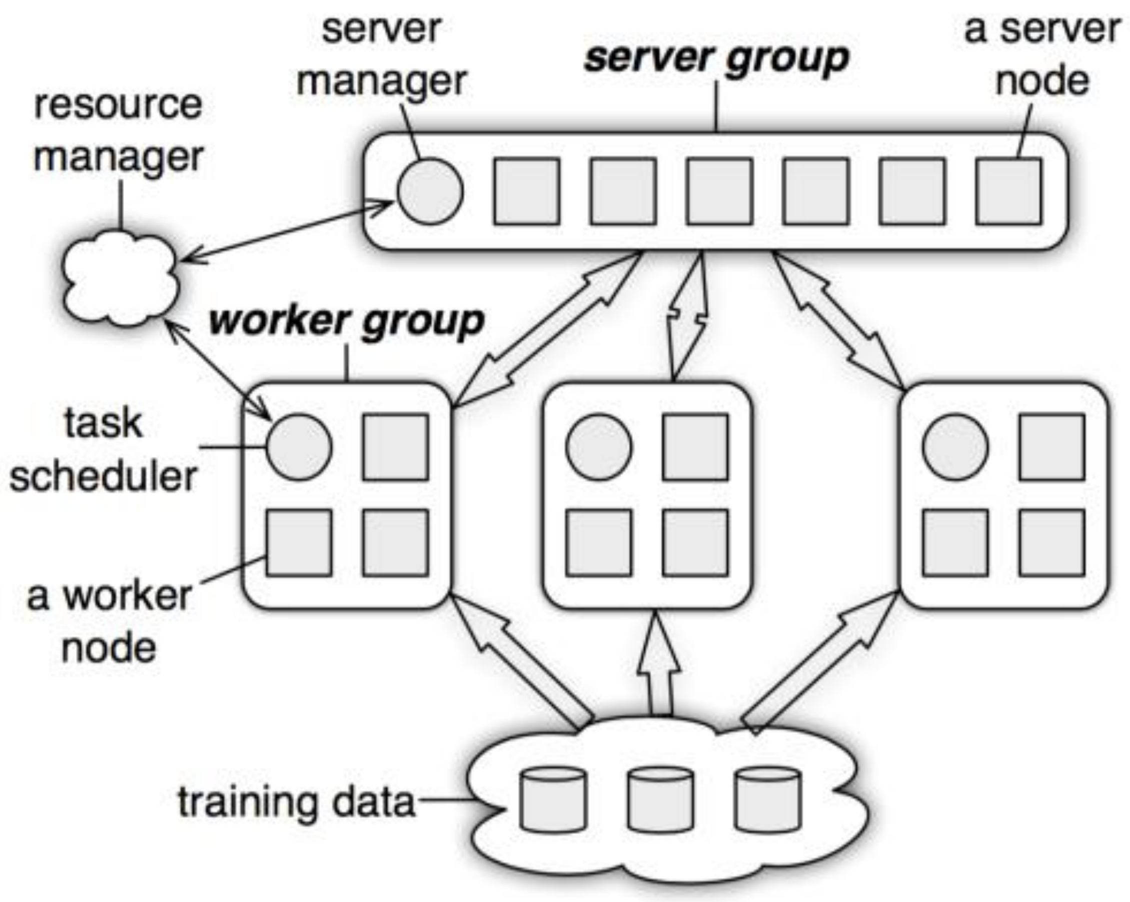
Review: Gradient Descent

- We only discuss the gradient-descent-based methods
- Intuitive description of these methods:
  - Read a mini batch of input data
  - Calculate the loss of current version of model
  - Compute the model updates (e.g., gradients)
  - Apply the updates to the model

Gradient Descent Distributed

- We only discuss the gradient-descent-based methods
- Intuitive description of these methods:
  - Read a mini batch of input data
  - Grabs a global copy of model
    - Everyone needs to train something similar
  - Calculate the loss of current version of model
  - Compute the model updates (e.g., gradients)
  - Apply the updates to the global model
    - In gradient descent: average gradients from everyone

Parameter Server



Distributed Training in TensorFlow

- Today: data-parallel training
  - As the graph shown above, all workers train the same model, but the data is not necessarily the same.
  - Define clusters structure, to let the graph know which is PS and which is worker.
  - We will post an example on canvas and GitHub, not discussed today.