TensorFlow: a Framework for Scalable Machine Learning

ACM Learning Center, 2016

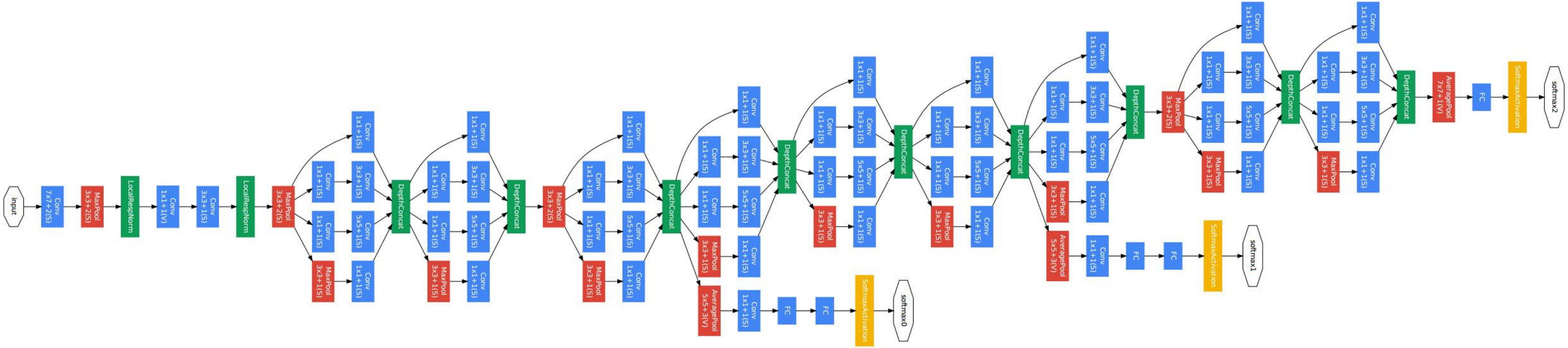
You probably want to know...

- What is TensorFlow?
- Why did we create TensorFlow?
- How does TensorFlow work?
- Code: Linear Regression
- Code: Convolution Deep Neural Network
- Advanced Topics: Queues and Devices

TensorFlow

- Fast, flexible, and scalable open-source machine learning library
- One system for research and production
- Runs on CPU, GPU, TPU, and Mobile
- Apache 2.0 license

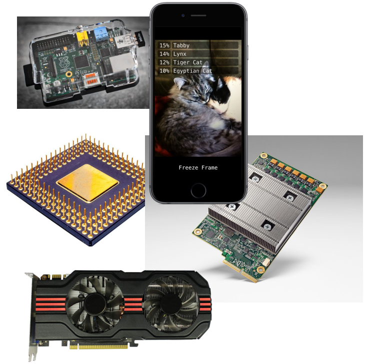
Machine learning gets complex quickly



Modeling complexity

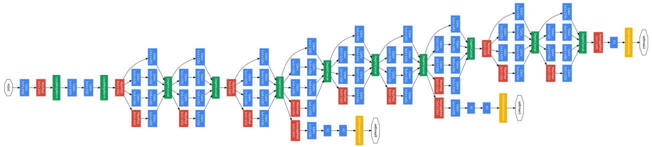
Machine learning gets complex quickly

Distributed
System



Heterogenous
System

TensorFlow Handles Complexity



Modeling complexity

Distributed System

Heterogenous System

Under the Hood

A multidimensional array.

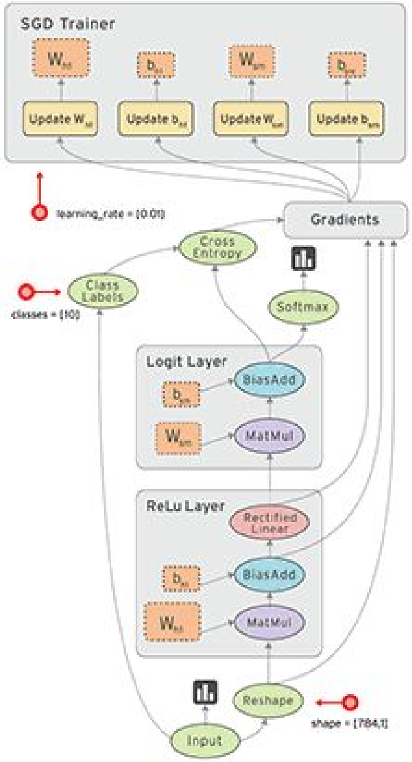
TensorFlow

A graph of operations.

The TensorFlow Graph

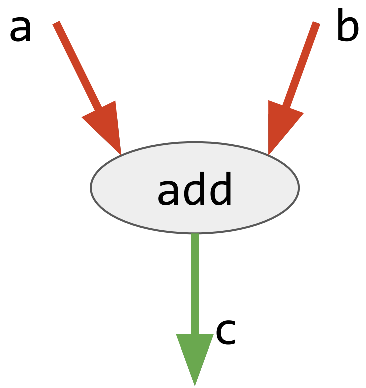
Computation is defined as a graph

- Graph is defined in high-level language (Python)
- Graph is compiled and optimized
- Graph is executed (in parts or fully) on available low level devices (CPU, GPU, TPU)
- Nodes represent computations and state
- Data (tensors) flow along edges

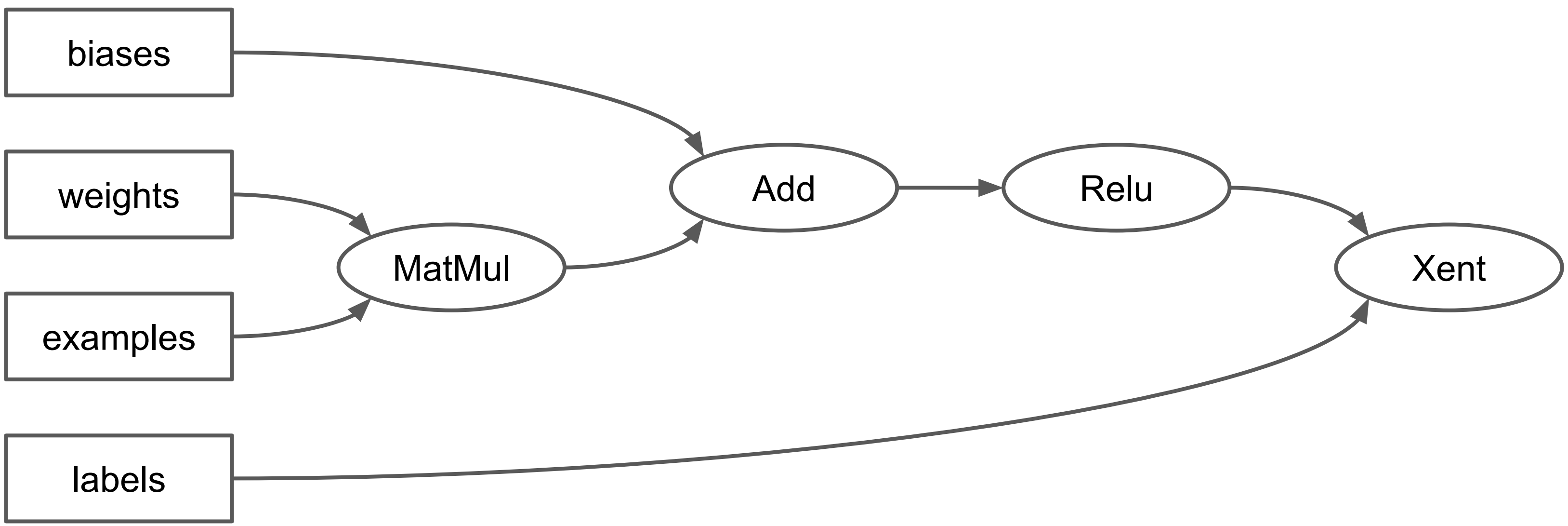


Build a graph; then run it.

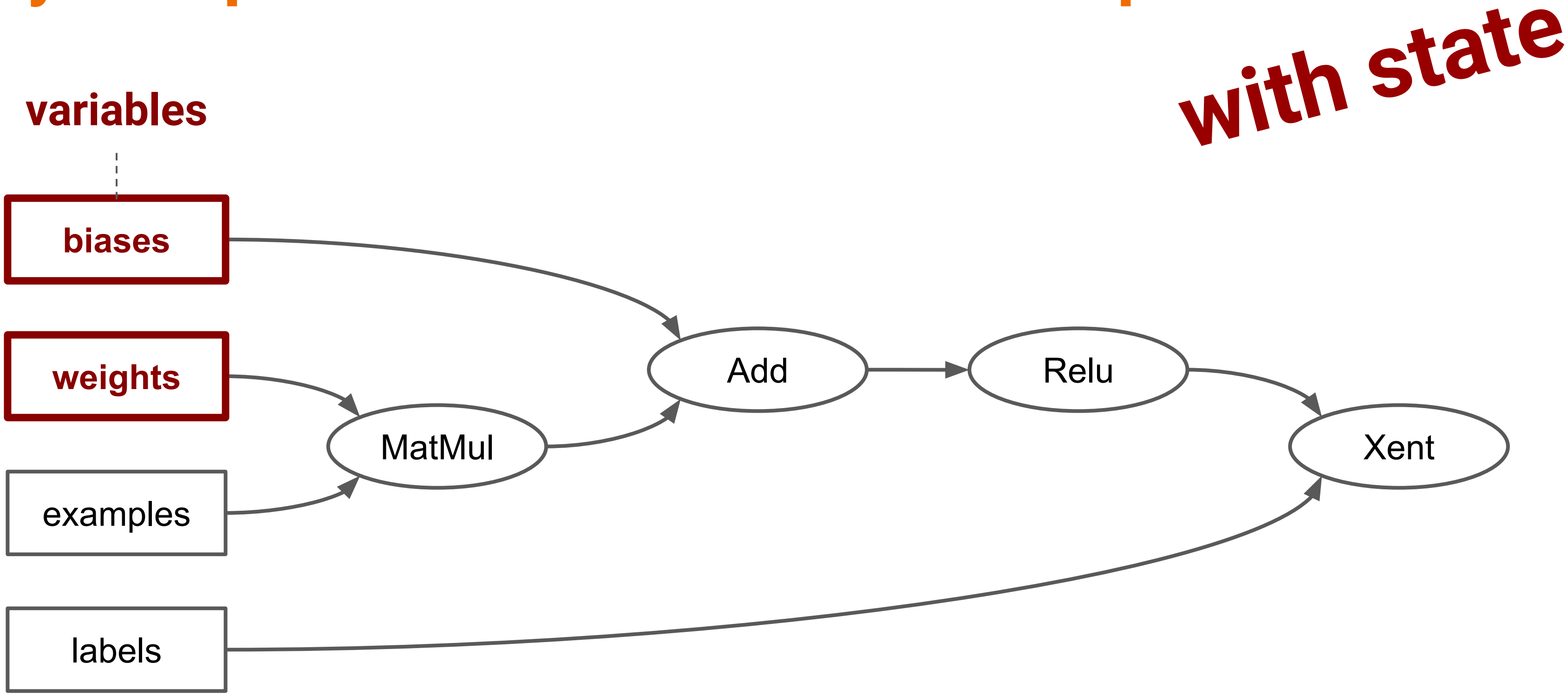
```
...
c = tf.add(a, b)

...

session = tf.Session()
value_of_c = session.run(c, {a=1, b=2})
```



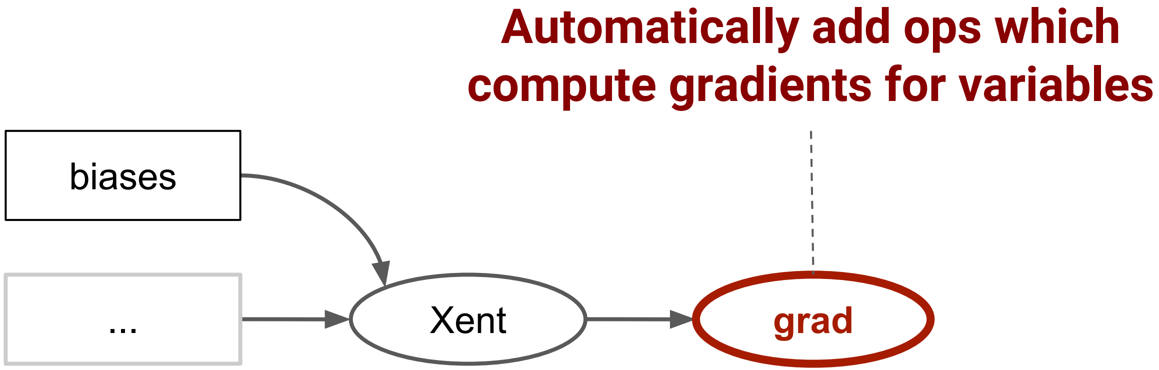
Any Computation is a TensorFlow Graph



Any Computation is a TensorFlow Graph



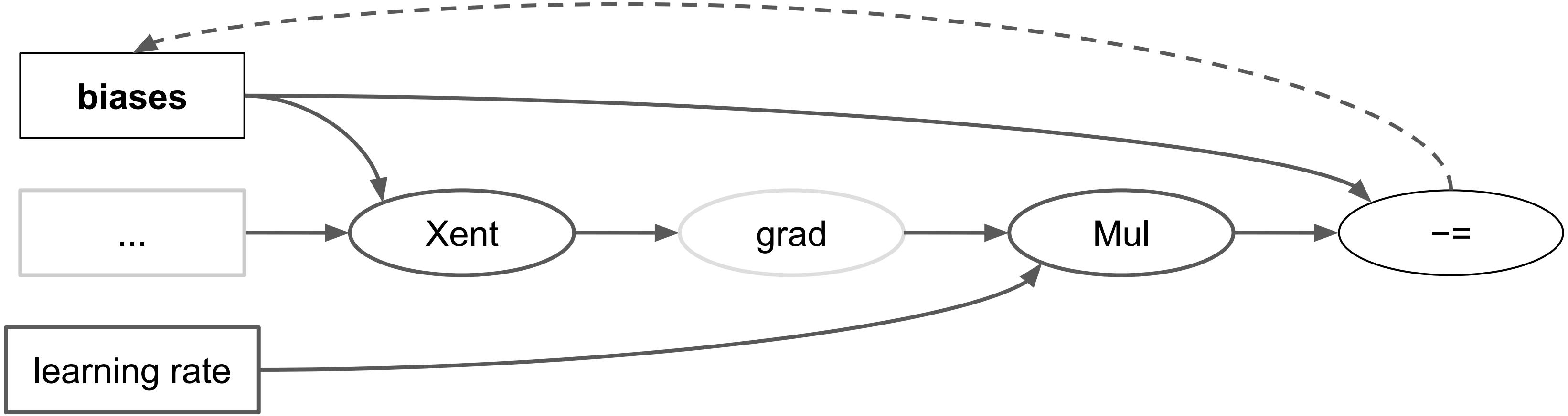
Automatic Differentiation



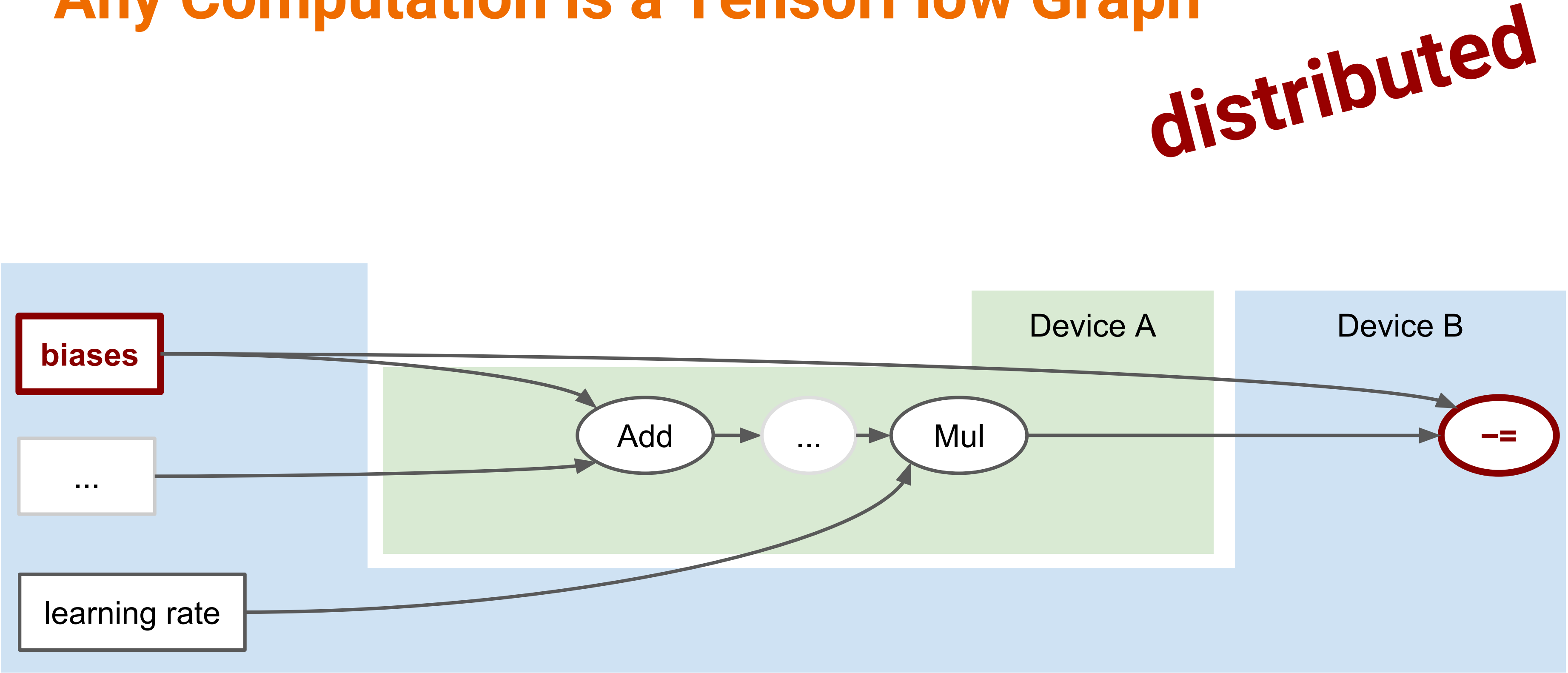
Any Computation is a TensorFlow Graph

Simple gradient descent:

with state

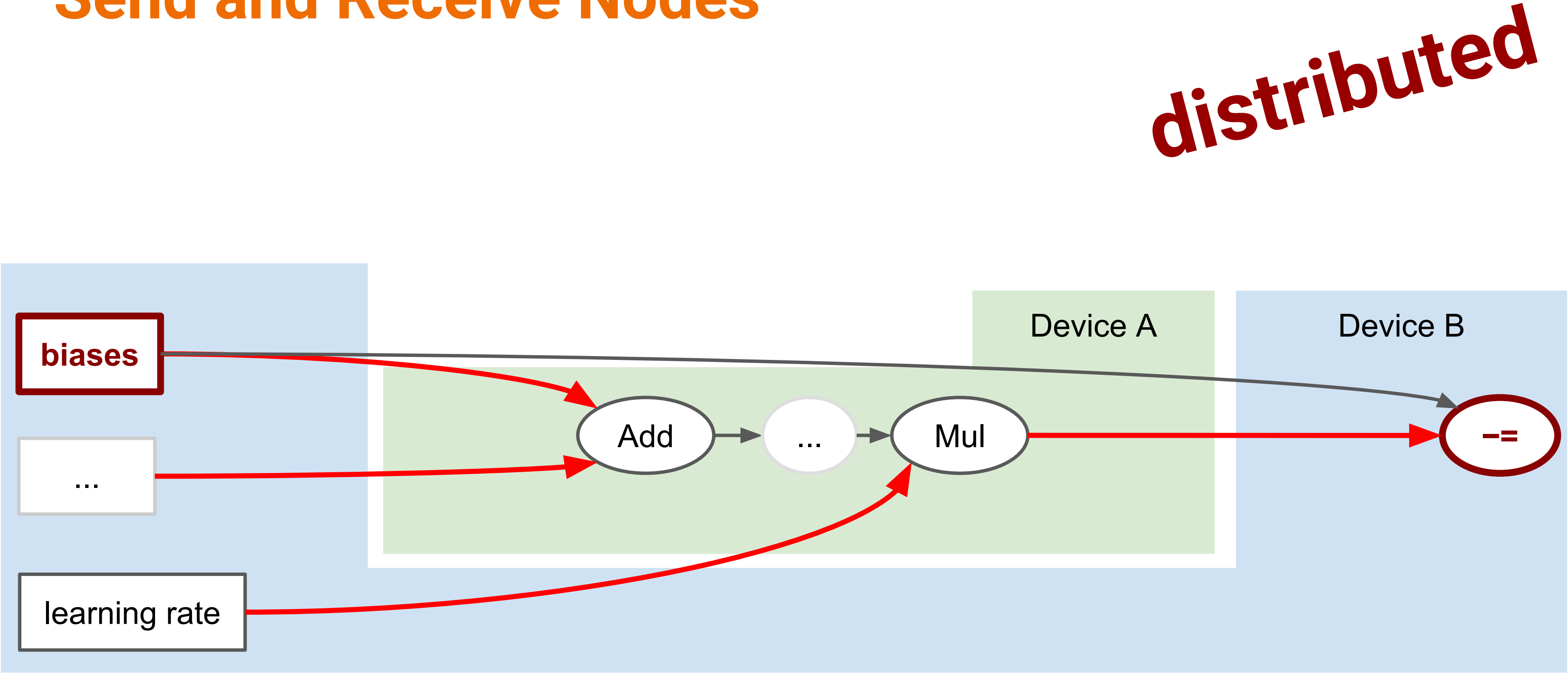


Any Computation is a TensorFlow Graph



Devices: Processes, Machines, CPUs, GPUs, TPUs, etc

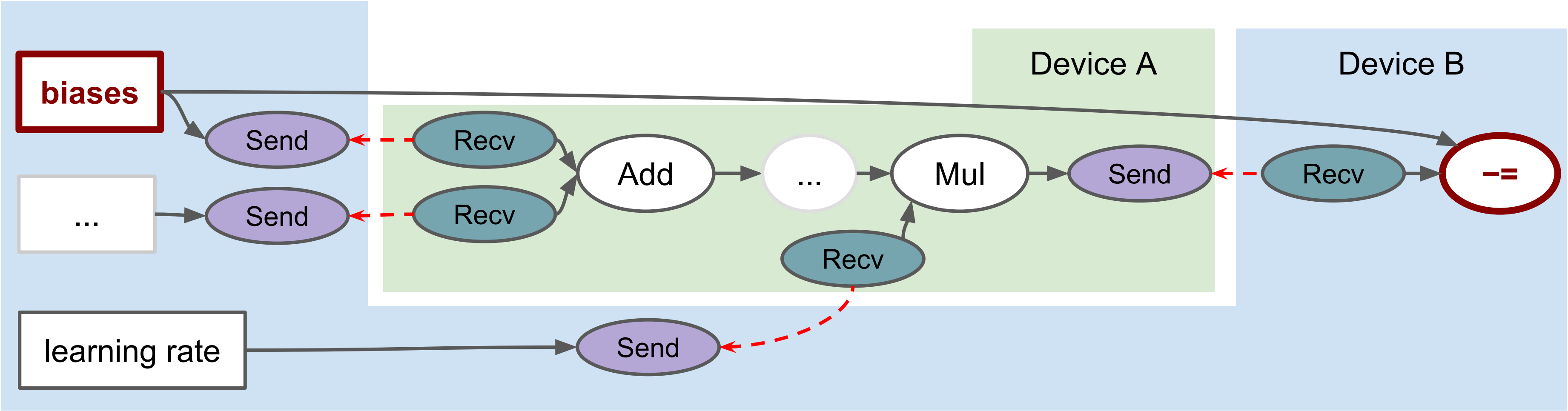
Send and Receive Nodes



Devices: Processes, Machines, CPUs, GPUs, TPUs, etc

Send and Receive Nodes

distributed



Devices: Processes, Machines, CPUs, GPUs, TPUs, etc

Linear Regression

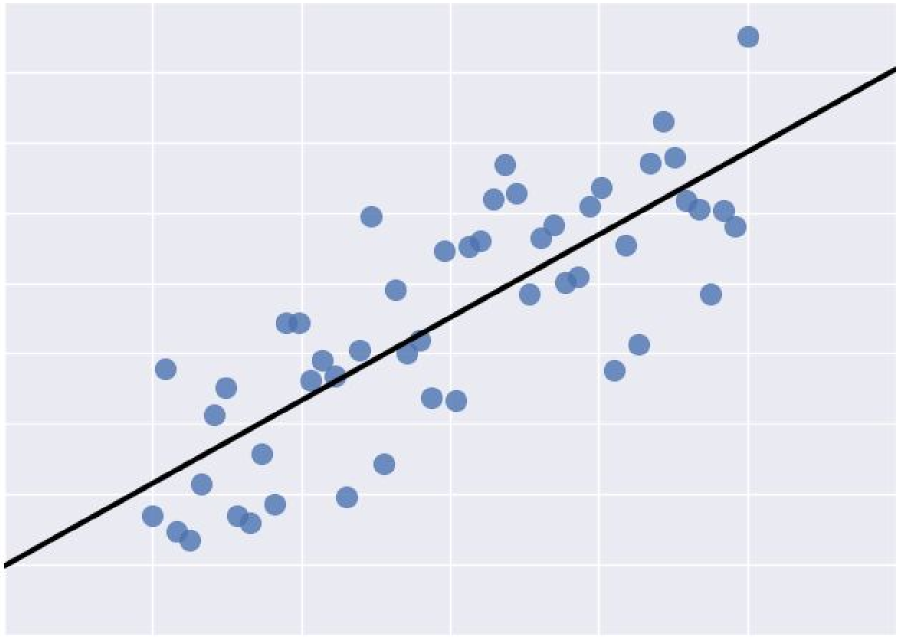
Linear Regression

result

input

$$y = Wx + b$$

parameters



What are we trying to do?

Mystery equation: `y = 0.1 * x + 0.3 + noise`

Model: `y = W * x + b`

Objective: Given enough (`x`, `y`) value samples, figure out the value of `W` and `b`.

y = Wx + b in TensorFlow

```
import tensorflow as tf
```

y = Wx + b in TensorFlow

```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name=”x”)
```

y = Wx + b in TensorFlow

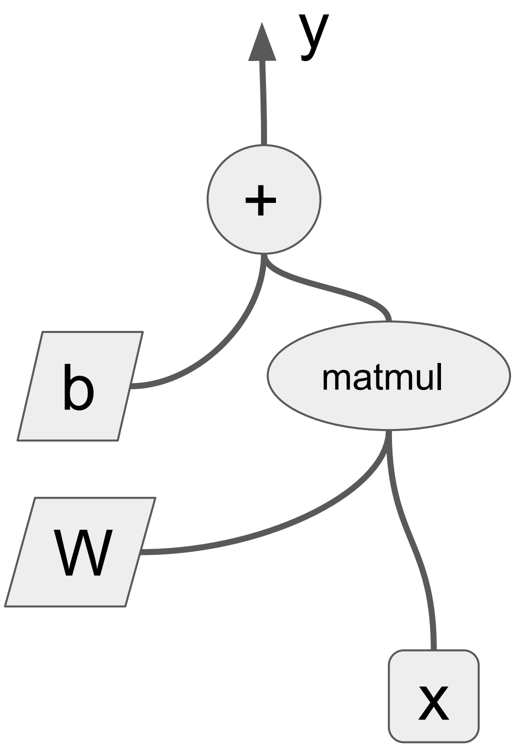
```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name=”x”)

W = tf.get_variable(shape=[], name=”W”)
```

y = Wx + b in TensorFlow

```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name=”x”)

W = tf.get_variable(shape=[], name=”W”)

b = tf.get_variable(shape=[], name=”b”)
```

y = Wx + b in TensorFlow

```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                   dtype=tf.float32, name="x")

W = tf.get_variable(shape=[], name="W")

b = tf.get_variable(shape=[], name="b")

y = W * x + b
```



Variables Must be Initialized

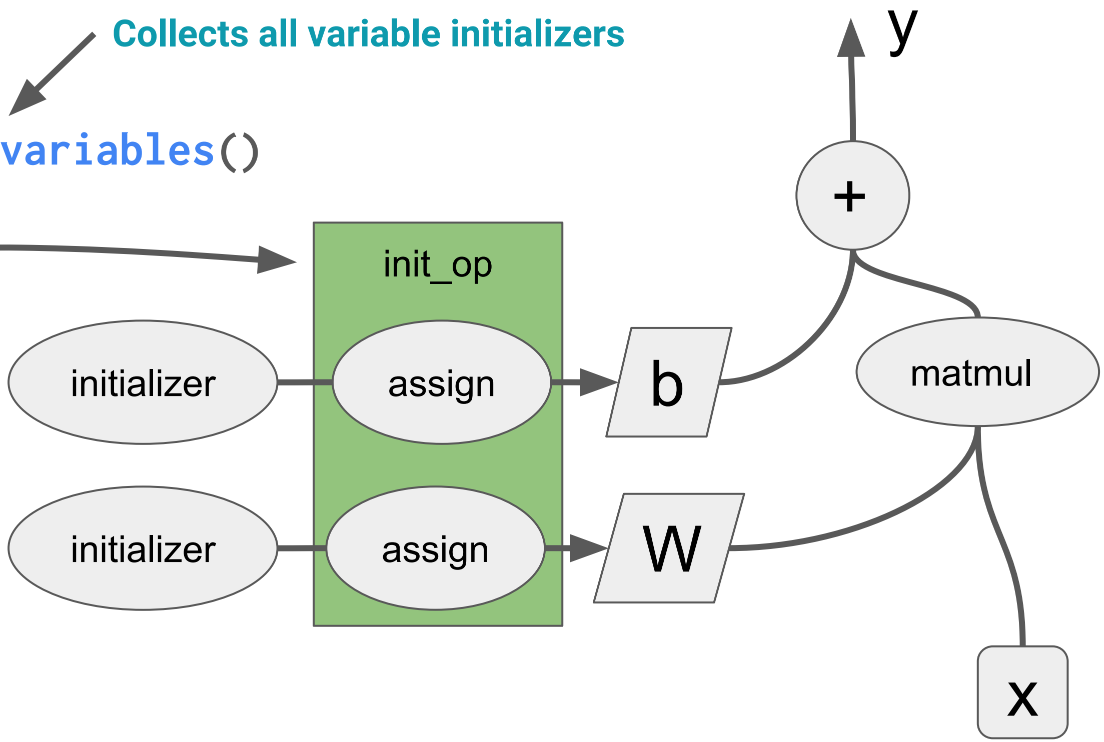
Collects all variable initializers

```
init_op = tf.initialize_all_variables()
```

Makes an execution environment

```
sess = tf.Session()

sess.run(init_op)
```

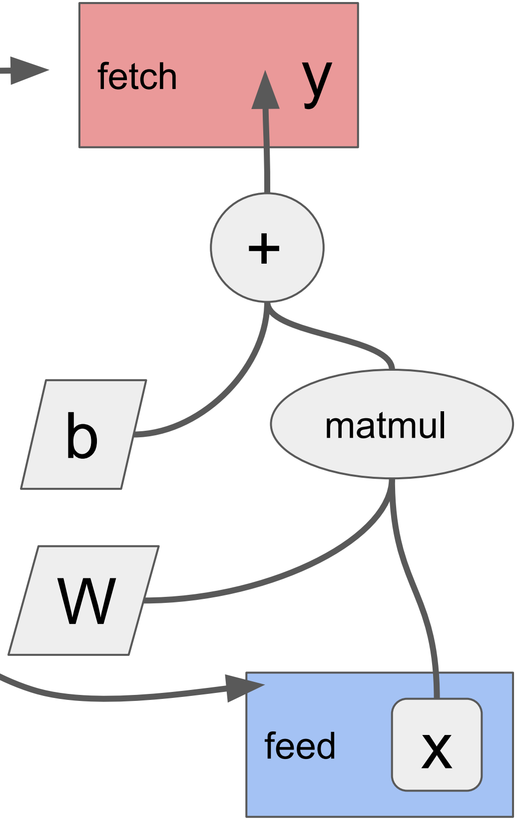
Actually initialize the variables



Running the Computation

```
x_in = 3

sess.run(y, feed_dict={x: x_in})
```

- Only what's used to compute a fetch will be evaluated
- All Tensors can be fed, but all placeholders must be fed



Putting it all together

```
import tensorflow as tf
x = tf.placeholder(shape=[None],
                   dtype=tf.float32,
                   name='x')
W = tf.get_variable(shape=[], name='W')
b = tf.get_variable(shape=[], name='b')
y = W * x + b
```

Build the graph

```
with tf.Session() as sess:
```

Prepare execution environment

```
  sess.run(tf.initialize_all_variables())
```

Initialize variables

```
  print(sess.run(y, feed_dict={x: x_in}))
```

Run the computation (usually often)

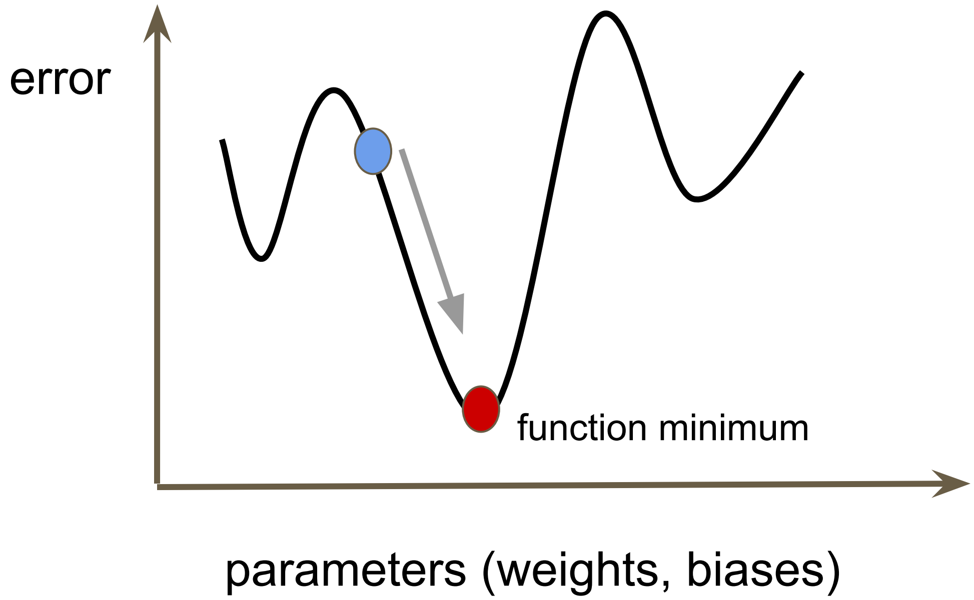
Define a Loss

Given x, y compute a loss, for instance:

$$L = (y - y_{label})^2$$

```
# create an operation that calculates loss.
loss = tf.reduce_mean(tf.square(y - y_data))
```

Minimize loss: optimizers




```
tf.train.AdadeltaOptimizer

tf.train.AdagradOptimizer

tf.train.AdagradDAOptimizer

tf.train.AdamOptimizer

…
```

Train

Feed (x, y_{label}) pairs and adjust W and b to decrease the loss.

$$W \leftarrow W - \eta \,(dL/dW)$$

$$b \leftarrow b - \eta \,(dL/db)$$

TensorFlow computes gradients automatically

```
# Create an optimizer

optimizer = tf.train.GradientDescentOptimizer(0.5)

# Create an operation that minimizes loss.

train = optimizer.minimize(loss)
```

Learning rate

Putting it all together

```
loss = tf.reduce_mean(tf.square(y - y_label))
optimizer = tf.train.GradientDescentOptimizer(0.5)
train = optimizer.minimize(loss)
with tf.Session() as sess:
  sess.run(tf.initialize_all_variables())
  for i in range(1000):
    sess.run(train, feed_dict={x: x_in[i],
                               y_label: y_in[i]})
```

Define a loss

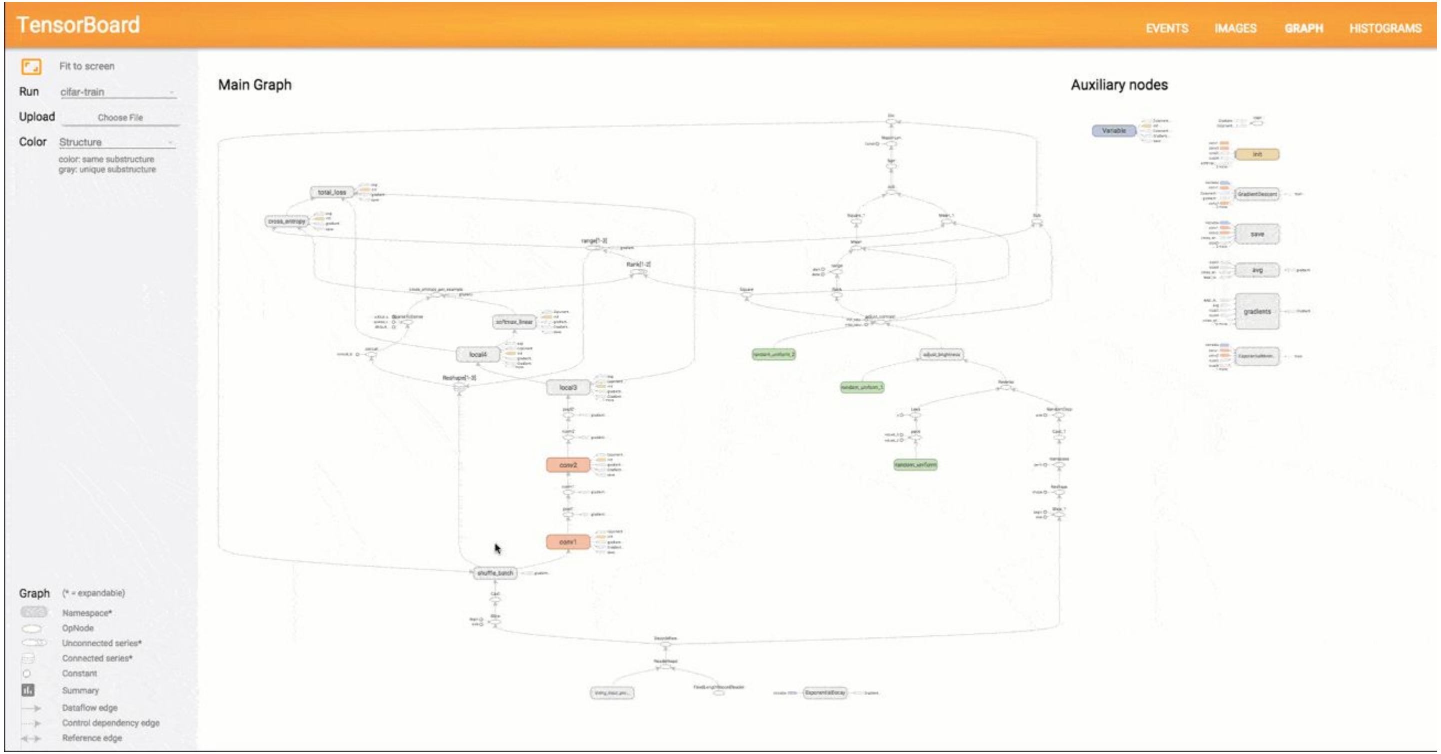
Create an optimizer

Op to minimize the loss

Initialize variables

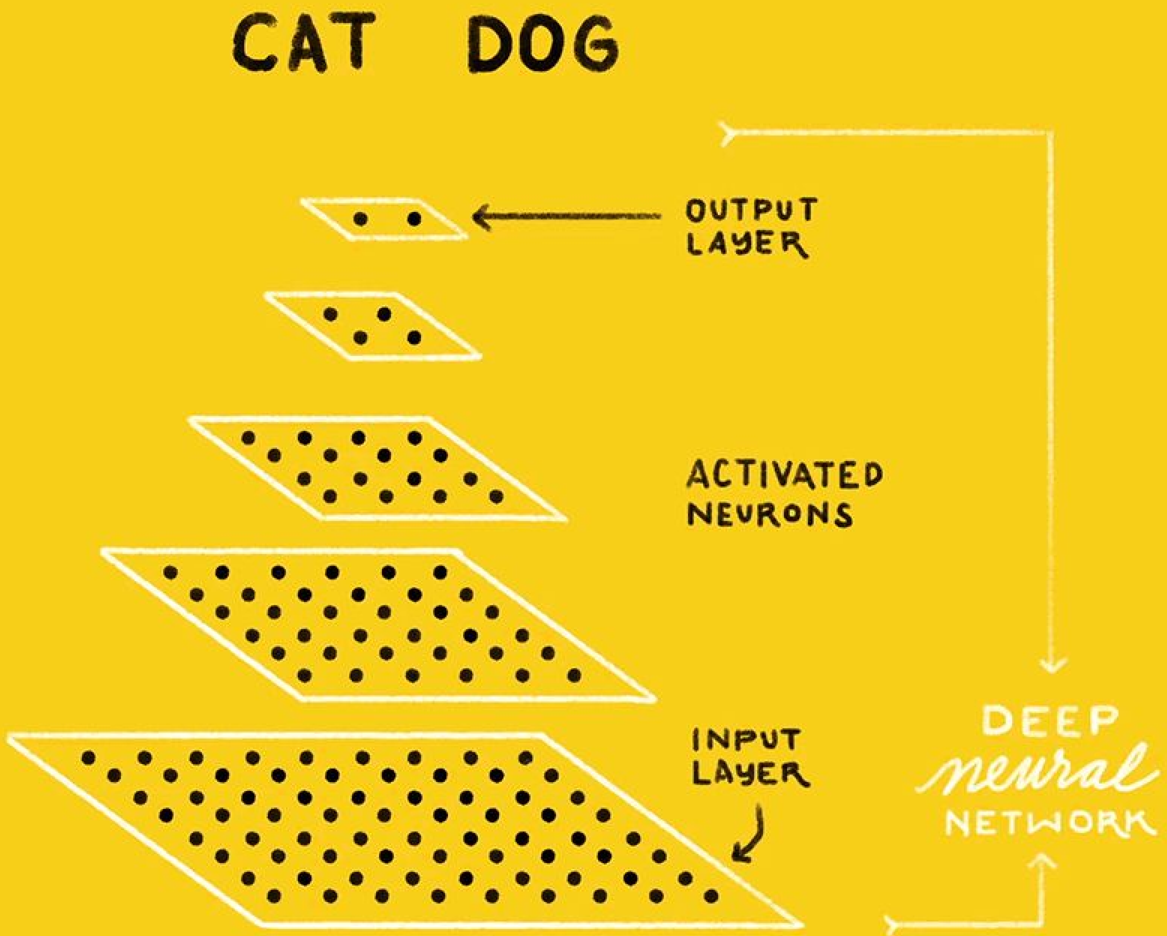
Iteratively run the training op

TensorBoard



Deep Neural Network

IS THIS A
CAT or DOG?



Remember linear regression?

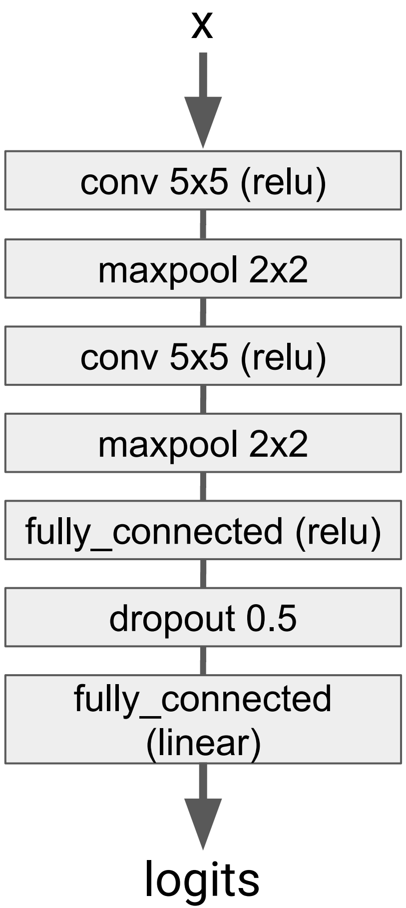
```
import tensorflow as tf
x = tf.placeholder(shape=[None],
                   dtype=tf.float32,
                   name='x')
W = tf.get_variable(shape=[], name='W')
b = tf.get_variable(shape=[], name='b')
y = W * x + b
```

Build the graph

```
loss = tf.reduce_mean(tf.square(y - y_label))

optimizer = tf.train.GradientDescentOptimizer(0.5)

train = optimizer.minimize(loss)

...
```

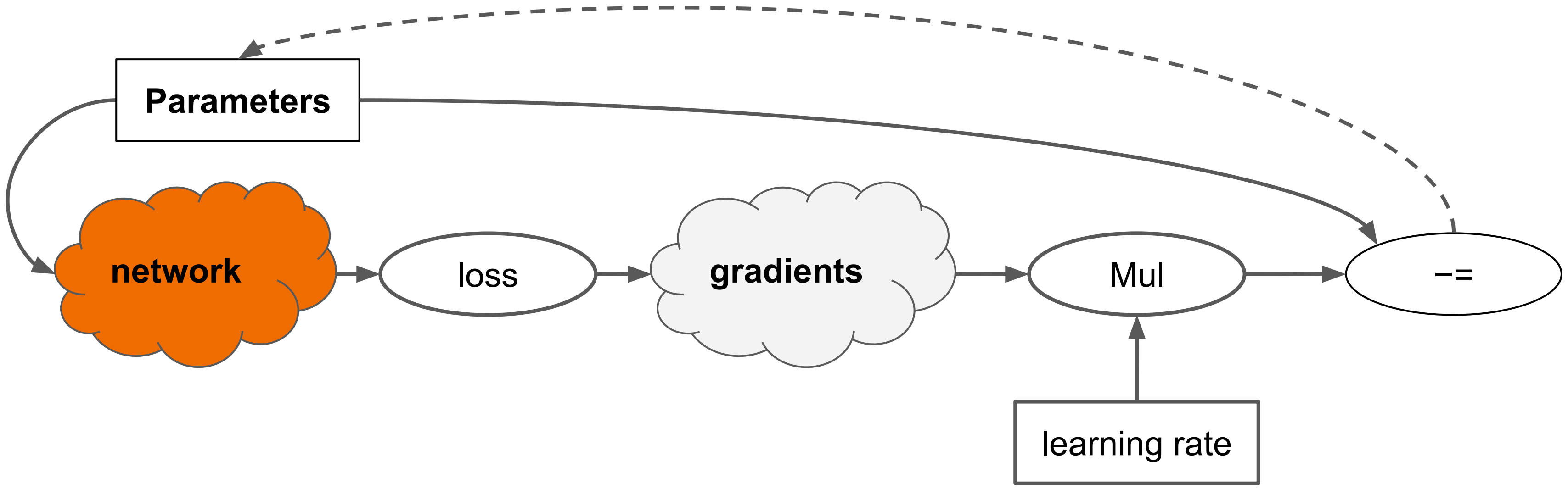
Convolutional DNN

```
x = tf.contrib.layers.conv2d(x, kernel_size=[5,5], ...)

x = tf.contrib.layers.max_pool2d(x, kernel_size=[2,2], ...)

x = tf.contrib.layers.conv2d(x, kernel_size=[5,5], ...)

x = tf.contrib.layers.max_pool2d(x, kernel_size=[2,2], ...)

x = tf.contrib.layers.fully_connected(x, activation_fn=tf.nn.relu)

x = tf.contrib.layers.dropout(x, 0.5)

logits = tf.config.layers.linear(x)
```



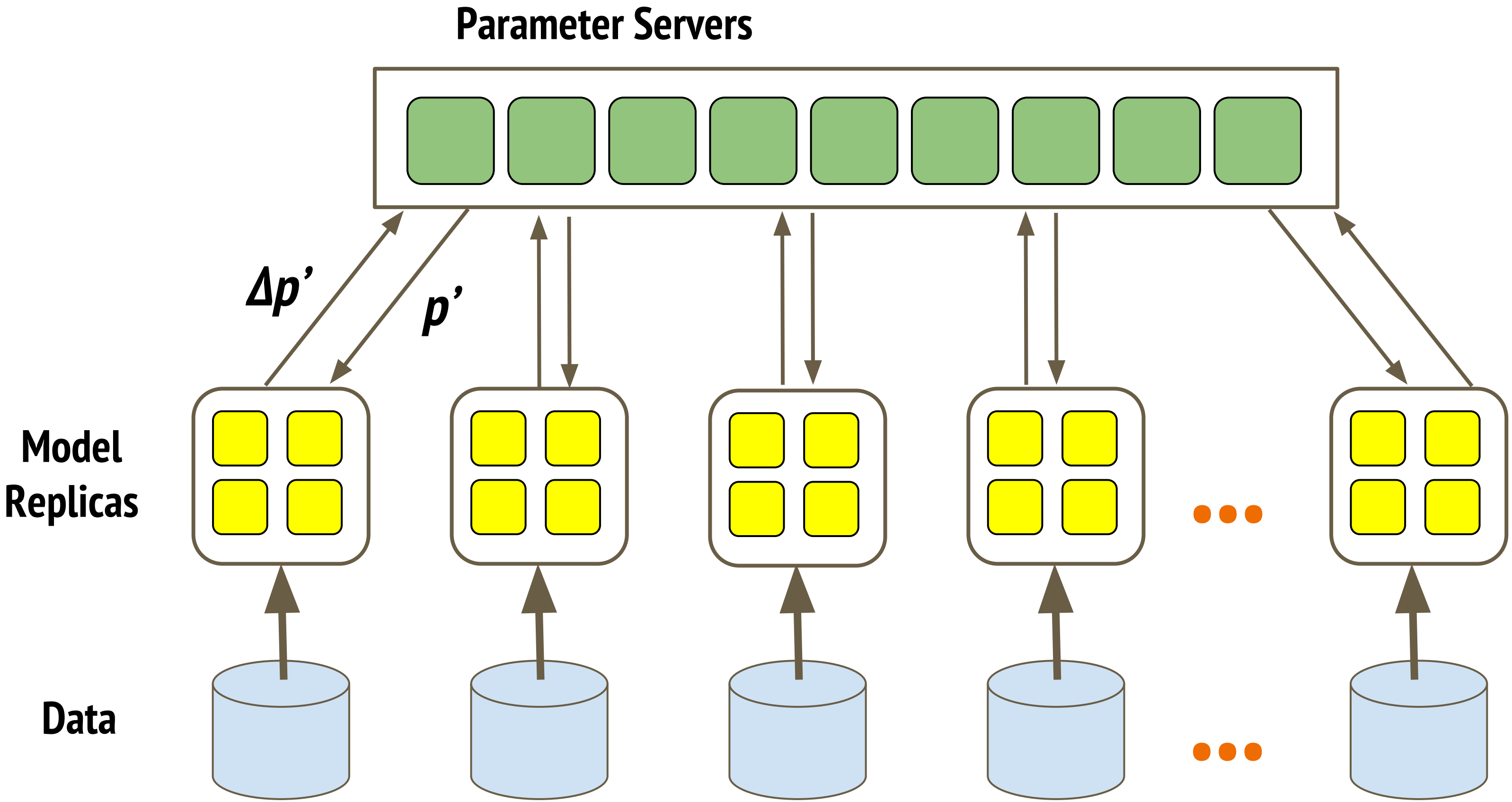
https://github.com/martinwicke/tensorflow-tutorial/blob/master/2_mnist.ipynb

Defining Complex Networks



Distributed TensorFlow

Data Parallelism



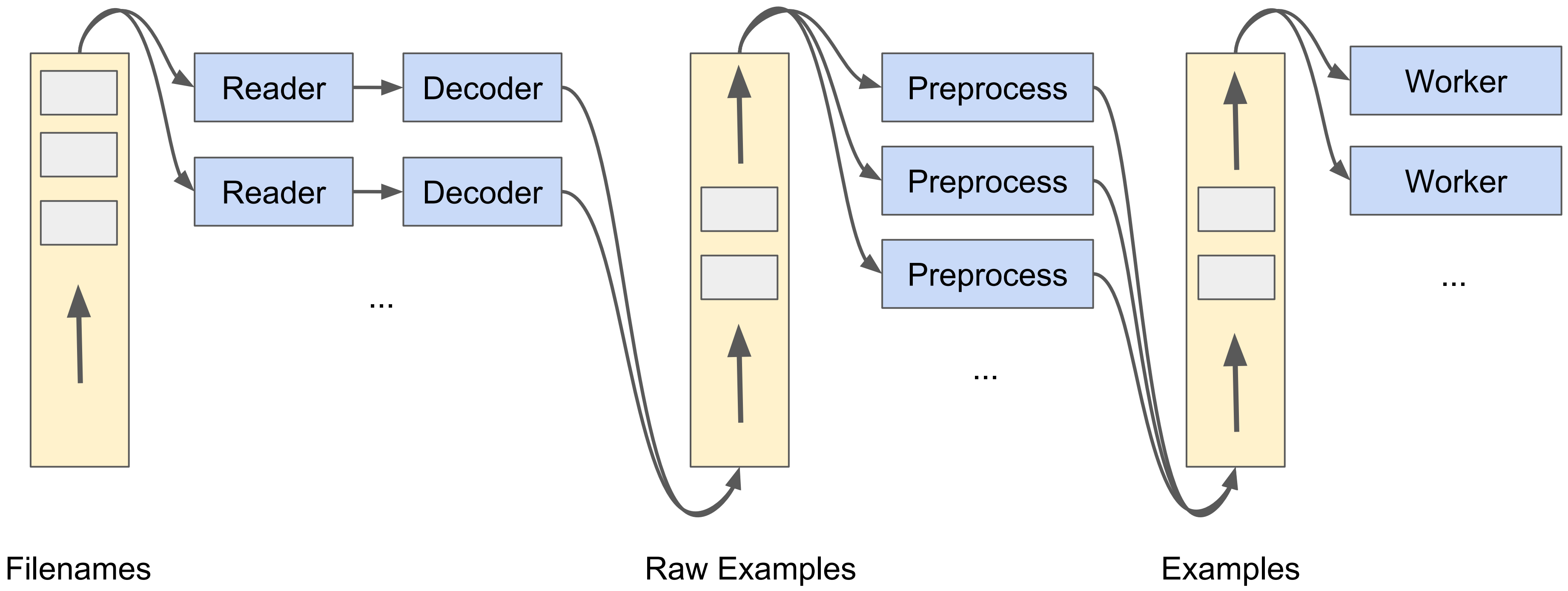
Describe a cluster: ClusterSpec

```
tf.train.ClusterSpec({
    "worker": [
        "worker0.example.com:2222",
        "worker1.example.com:2222",
        "worker2.example.com:2222"
    ],
    "ps": [
        "ps0.example.com:2222",
        "ps1.example.com:2222"
    ]})
```

Share the graph across devices

```
with tf.device("/job:ps/task:0"):
  weights_1 = tf.Variable(...)
  biases_1 = tf.Variable(...)

with tf.device("/job:ps/task:1"):
  weights_2 = tf.Variable(...)
  biases_2 = tf.Variable(...)

with tf.device("/job:worker/task:7"):
  input, labels = ...
  layer_1 = tf.nn.relu(tf.matmul(input, weights_1) + biases_1)
  logits = tf.nn.relu(tf.matmul(layer_1, weights_2) + biases_2)
  train_op = ...

with tf.Session("grpc://worker7.example.com:2222") as sess:
  for _ in range(10000):
    sess.run(train_op)
```

Input Pipelines with Queues



Tutorials & Courses

Tutorials on tensorflow.org**:**

Image recognition: https://www.tensorflow.org/tutorials/image_recognition

Word embeddings: https://www.tensorflow.org/versions/word2vec

Language Modeling: https://www.tensorflow.org/tutorials/recurrent

Translation: https://www.tensorflow.org/versions/seq2seq

Deep Dream:
https://tensorflow.org/code/tensorflow/examples/tutorials/deepdream/deepdream.ipynb

Thank you and have fun!

Martin Wicke
@martin_wicke

Rajat Monga
@rajatmonga

Extras

Inception

An Alaskan Malamute (left) and a Siberian Husky (right). Images from Wikipedia.

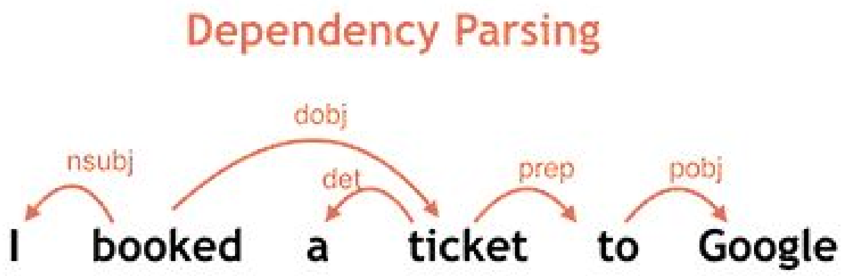
https://research.googleblog.com/2016/08/improving-inception-and-image.html

Show and Tell


A person on a beach flying a kite.

https://research.googleblog.com/2016/09/show-and-tell-image-captioning-open.html

Parsey McParseface



https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html

Text Summarization

Original text

- *Alice and Bob took the train to visit the zoo. They saw a* ***baby giraffe, a lion, and a flock of colorful tropical birds****.*

Abstractive summary

- *Alice and Bob visited the zoo and saw* ***animals and birds****.*

https://research.googleblog.com/2016/08/text-summarization-with-tensorflow.html https://www.tensorflow.org/mobile.html

TensorFlow™

GET STARTED TUTORIALS HOW TO MOBILE API RESOURCES ABOUT

Mobile TensorFlow

TensorFlow was designed with mobile and embedded platforms in mind. We have sample code and build support you can try now for these platforms:

Android

iOS

Raspberry Pi

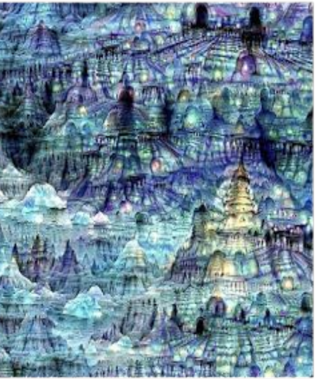
Many applications can benefit from on-device processing. Google Translate's instant visual translation is a great example. By running its processing locally, users get an incredibly responsive and interactive experience.

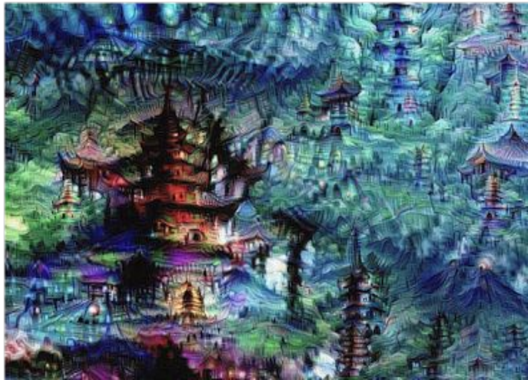
Mobile TensorFlow makes sense when there is a poor or missing network connection, or where sending continuous data to a server would be too expensive. We are working to help developers make lean mobile apps using TensorFlow, both by continuing to reduce the code footprint, and supporting quantization and lower precision arithmetic that reduce model size.

Claude Monet - Bouquet of Sunflowers
Images from the Metropolitan Museum of Art (with permission)

Image by @random_forests





Architecture