Getting Started with Tensorflow

Computing Lab II, Spring 2021

Deep Learning Packages

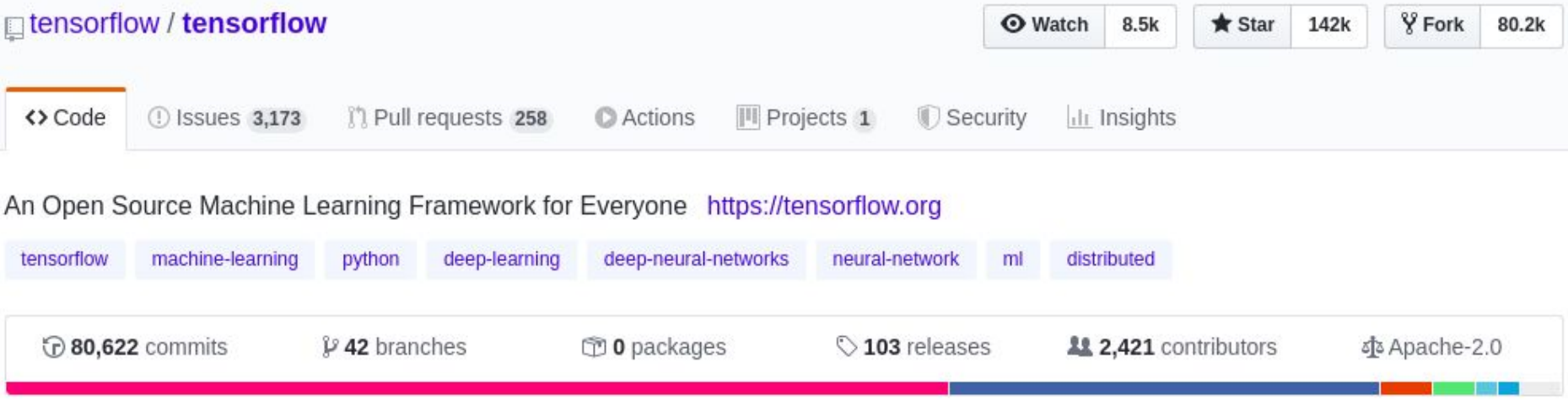
Caffe

and others ...

Deep Learning Frameworks

- Scale machine learning code
- **Compute gradients!**
- Standardize machine learning applications for sharing
- Plethora of deep learning packages available with different advantages, levels of abstraction, programming languages, etc.
- Provide an interface with GPU for parallel processing

What is Tensorflow



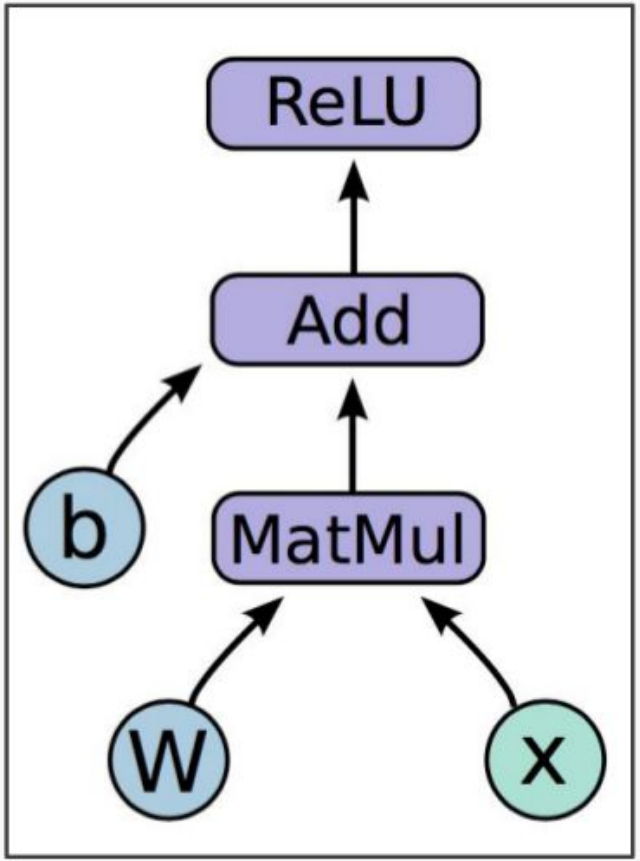
Programming Model

- **Idea:** Express numeric computation as a graph
- Graph nodes are operations that can have any number of inputs and exactly one output
- Graph edges are tensors that flow between nodes

Tensors are n-dimensional arrays

Programming Model

$$h = ReLU(Wx + b)$$



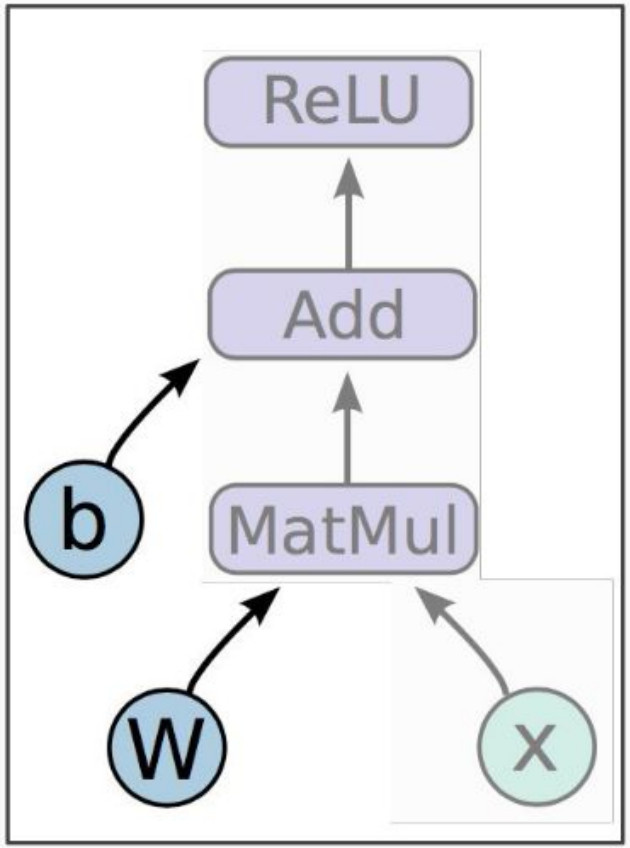
Programming Model

$$h = ReLU(Wx + b)$$

Variables are stateful nodes which output their current value.

State is retained across multiple executions of a graph

(mostly parameters)

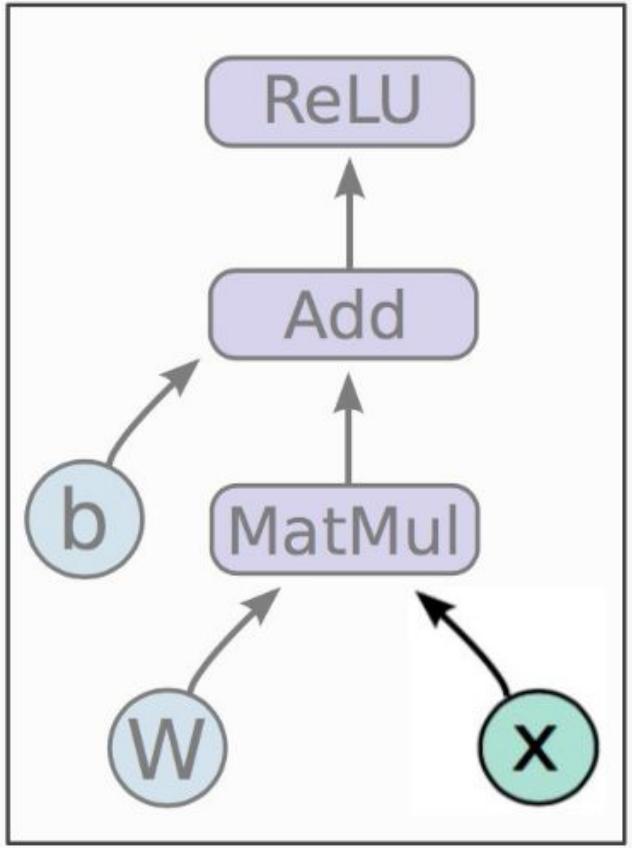


Programming Model

$$h = ReLU(Wx + b)$$

Placeholders are nodes whose value is fed in at execution time

(inputs, labels, ...)

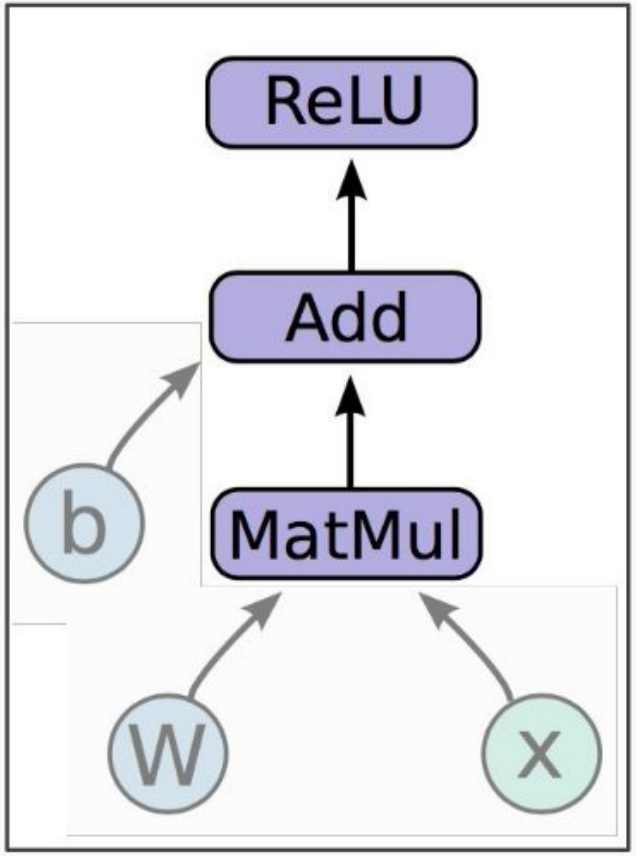


Programming Model

$$h = ReLU(Wx + b)$$

Mathematical operations:

MatMul: Multiply two matrix values.
Add: Add elementwise (with broadcasting).
ReLU: Activate with elementwise rectified linear function.



In code,

1. Create weights, including Initialization

   W ~ Uniform(-1, 1); b = 0

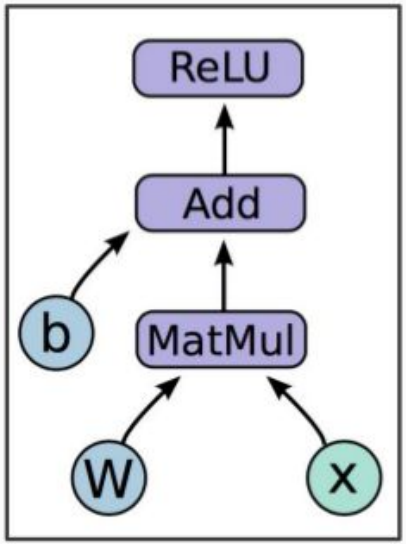
2. Create input placeholder x

   m * 784 input matrix

3. Build flow graph

```
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100), -1, 1))

x = tf.placeholder(tf.float32, (100, 784))

h = tf.nn.relu(tf.matmul(x, W) + b)
```

$$h = ReLU(Wx + b)$$



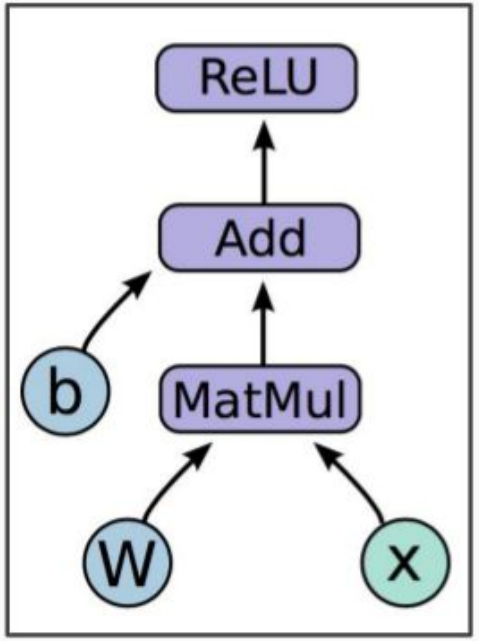
But where is the graph?

New nodes are automatically built into the underlying graph!

tf.get_default_graph().get_operations():

zeros/shape
zeros/Const
zeros
Variable
Variable/Assign
Variable/read
random_uniform/shape
random_uniform/min
random_uniform/max
random_uniform/RandomUniform
random_uniform/sub
random_uniform/mul
random_uniform
Variable_1
Variable_1/Assign
Variable_1/read
Placeholder
MatMul
add
Relu == h

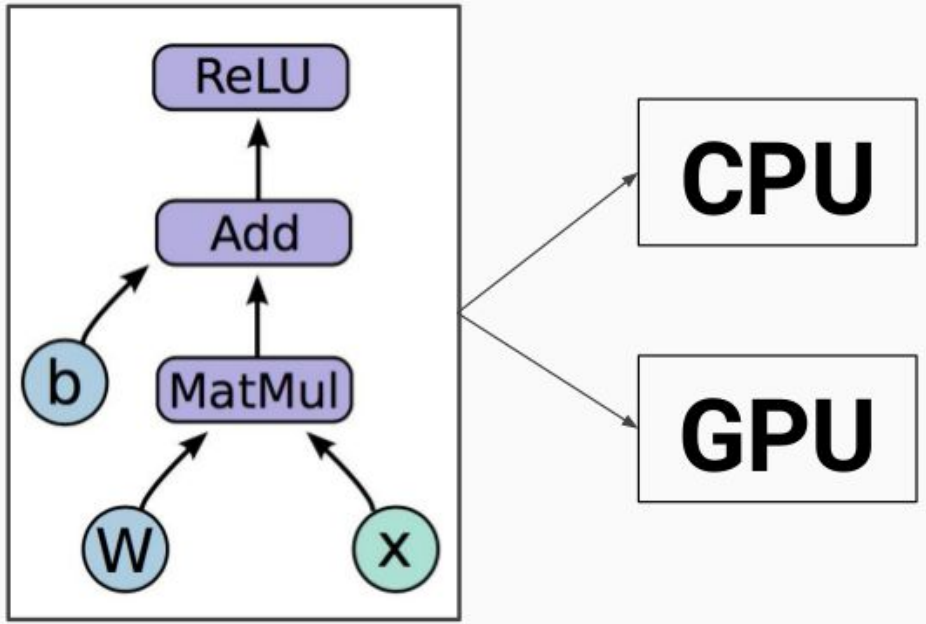
h refers to an op!



How do we run it?

So far we have defined a **graph**.

We can deploy this graph with a **session**: a binding to a particular execution context (e.g. CPU, GPU)



Getting output

`sess.run(fetches, feeds)`

Fetches: List of graph nodes.

Return the outputs of these nodes.

Feeds: Dictionary mapping from graph nodes to concrete values.

Specifies the value of each graph node given in the dictionary.

```
import numpy as np
import tensorflow as tf

b = tf.Variable(tf.zeros((100,)))
W = tf.Variable(tf.random_uniform((784, 100),
                                  -1, 1))

x = tf.placeholder(tf.float32, (100, 784))
h = tf.nn.relu(tf.matmul(x, W) + b)

sess = tf.Session()
sess.run(tf.initialize_all_variables())
sess.run(h, {x: np.random.random(100, 784)})
```

So what have we covered so far?

We first built a **graph** using **variables** and **placeholders**

We then deployed the graph onto a **session**, which is the **execution environment**

Next we will see how to **train** the model

How do we define the loss?

Use **placeholders** for labels

Build loss node using labels and **predictions**

```
prediction = tf.nn.softmax(...)  #Output of neural network
label = tf.placeholder(tf.float32, [100, 10])

cross_entropy = -tf.reduce_sum(label * tf.log(prediction), axis=1)
```

How do we compute Gradients?

```
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

`tf.train.GradientDescentOptimizer` is an **Optimizer** object

`tf.train.GradientDescentOptimizer(lr).minimize(cross_entropy)` adds optimization **operation** to computation graph

TensorFlow graph **nodes** have **attached gradient operations**

Gradient with respect to **parameters** computed with **backpropagation**

...automatically

Creating the train_step op

```
prediction = tf.nn.softmax(...)
label = tf.placeholder(tf.float32, [None, 10])

cross_entropy = tf.reduce_mean(-tf.reduce_sum(label * tf.log(prediction),
reduction_indices=[1]))

train_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)
```

Other Optimizers

`class AdadeltaOptimizer` : Optimizer that implements the Adadelta algorithm.

`class AdagradDAOptimizer` : Adagrad Dual Averaging algorithm for sparse linear models.

`class AdagradOptimizer` : Optimizer that implements the Adagrad algorithm.

`class AdamOptimizer` : Optimizer that implements the Adam algorithm.

READ tensorflow.org

These are other options

Training the Model

`sess.run(train_step, feeds)`

1. Create Session
2. Build training schedule
3. Run train_step

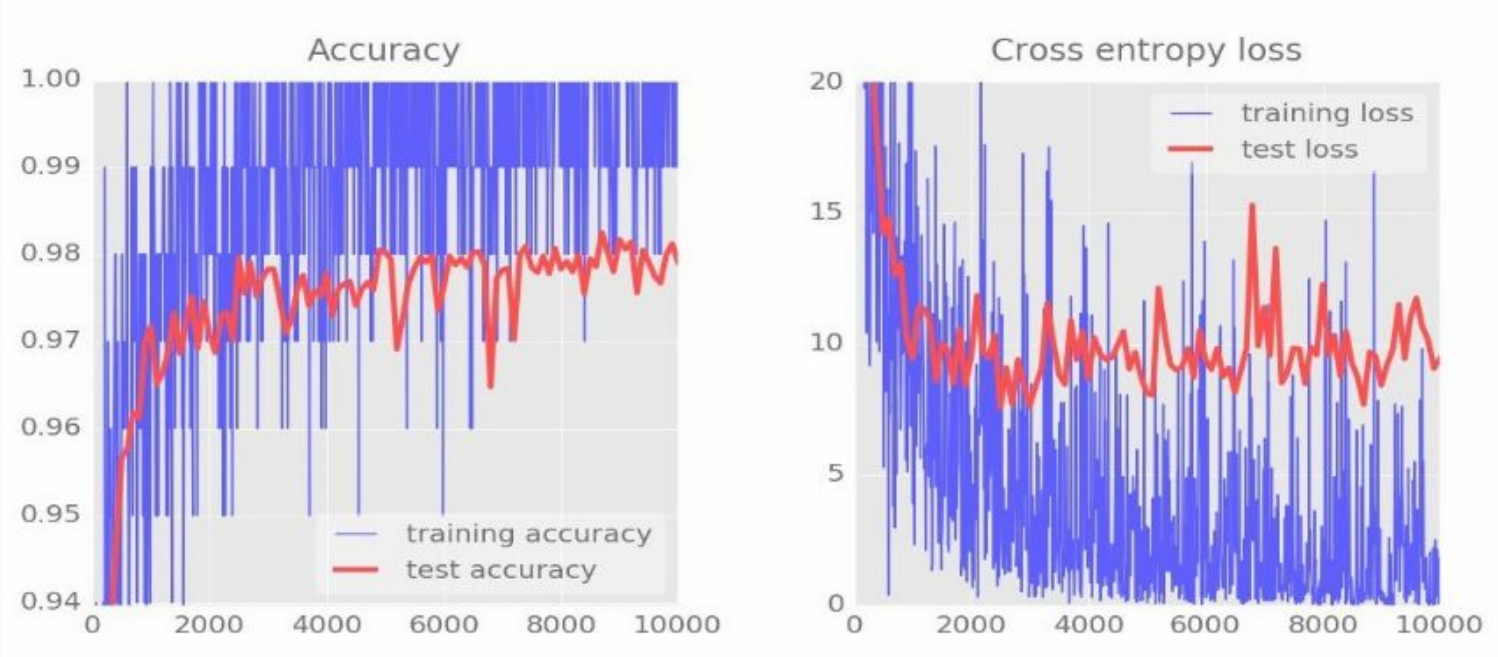
```
sess = tf.Session()
sess.run(tf.global_variables_initializer())

for i in range(1000):
    batch_x, batch_label = data.next_batch()
    sess.run(train_step, feed_dict={x: batch_x,
                                    label: batch_label}
```

Saving model weights

```
# Create a saver.
saver = tf.train.Saver(...variables...)
# Launch the graph and train, saving the model every 1,000 steps.
sess = tf.Session()
for step in xrange(1000000):
    sess.run(..training_op..)
    if step % 1000 == 0:
        # Append the step number to the checkpoint name:
        saver.save(sess, 'my-model', global_step=step)
```

READ tensorflow.org

Noisy Accuracy Curve



Thanks!

References:

1. http://cs231n.stanford.edu/
2. http://cs224n.stanford.edu/
3. Martin Gorner's Slides on Tensorflow
4. Keras Tutorial on Fashion-MNIST: https://www.tensorflow.org/tutorials/keras/classification