Intro to TensorFlow

Sp18 ML@B Workshop Series

Hosted by Machine Learning at Berkeley

ML@B

Agenda

What is TensorFlow?

Why TensorFlow?

Computation graphs and sessions

Why graphs?

Constants, Variables, Placeholders

Basic Tensorflow Program Structure

Feed Forward Newtork on MNIST

Lab

Questions

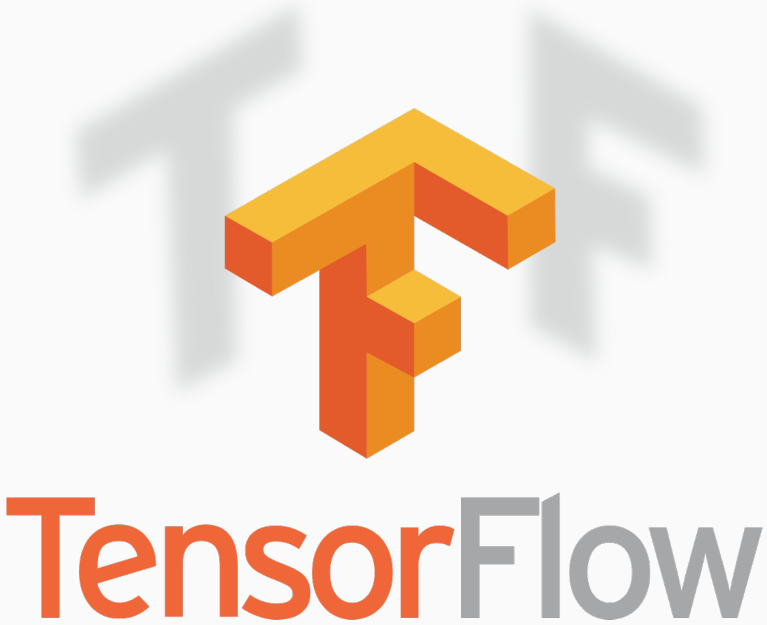
What is TensorFlow?

Tensor

A tensor is an n-D array of numbers and can have:

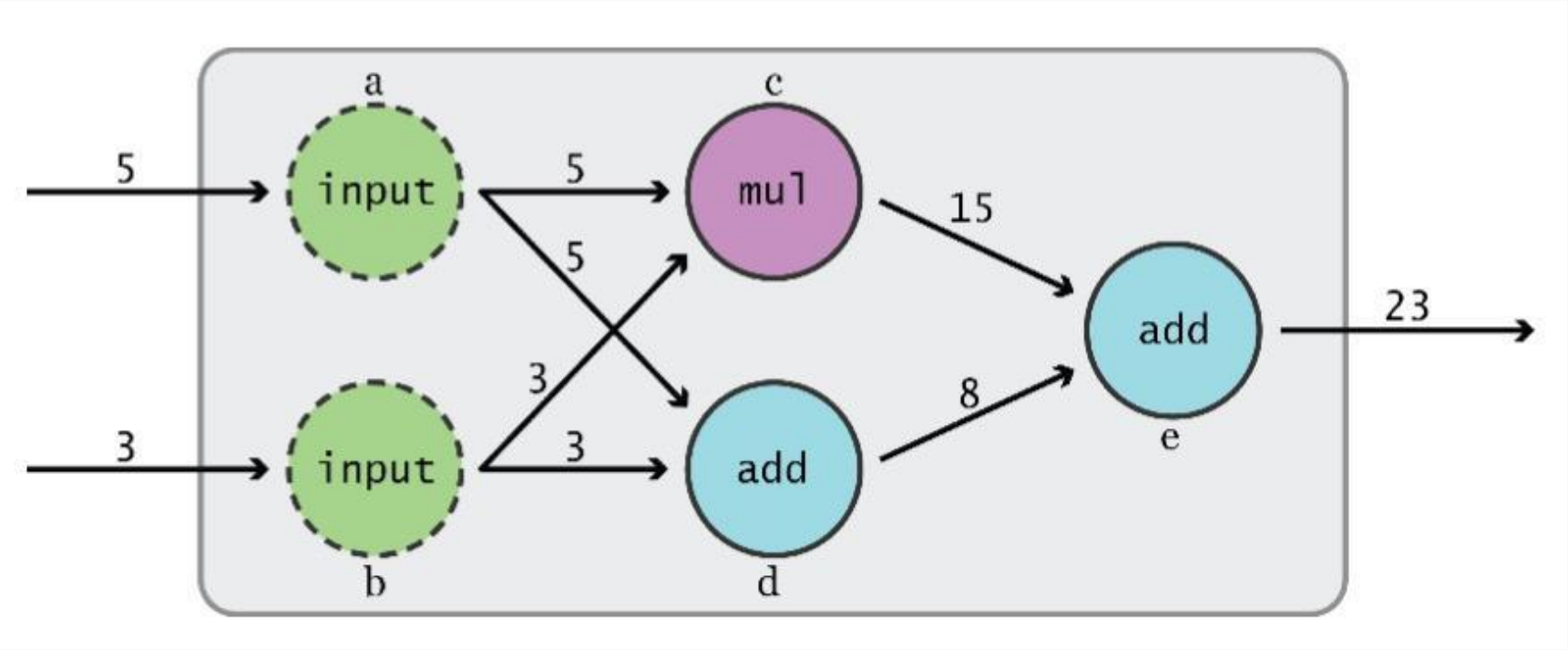
Dimensions	Example	Terminology
1	0 1 2	Vector
2	0 1 2 3 4 5 6 7 8	Matrix
3	0 1 2 3 4 5 6 7 8	3D Array (3rd order Tensor)
N	... ...	ND Array

Tensor + Flow = TensorFlow



- Open source software library for numerical computation using data flow graphs
- Separates definition of computations from their execution

Data Flow Graphs



Phase 1 : assemble a graph

Phase 2 : use a session to execute operations in the graph

Why TensorFlow?

Why TensorFlow?

ML@B

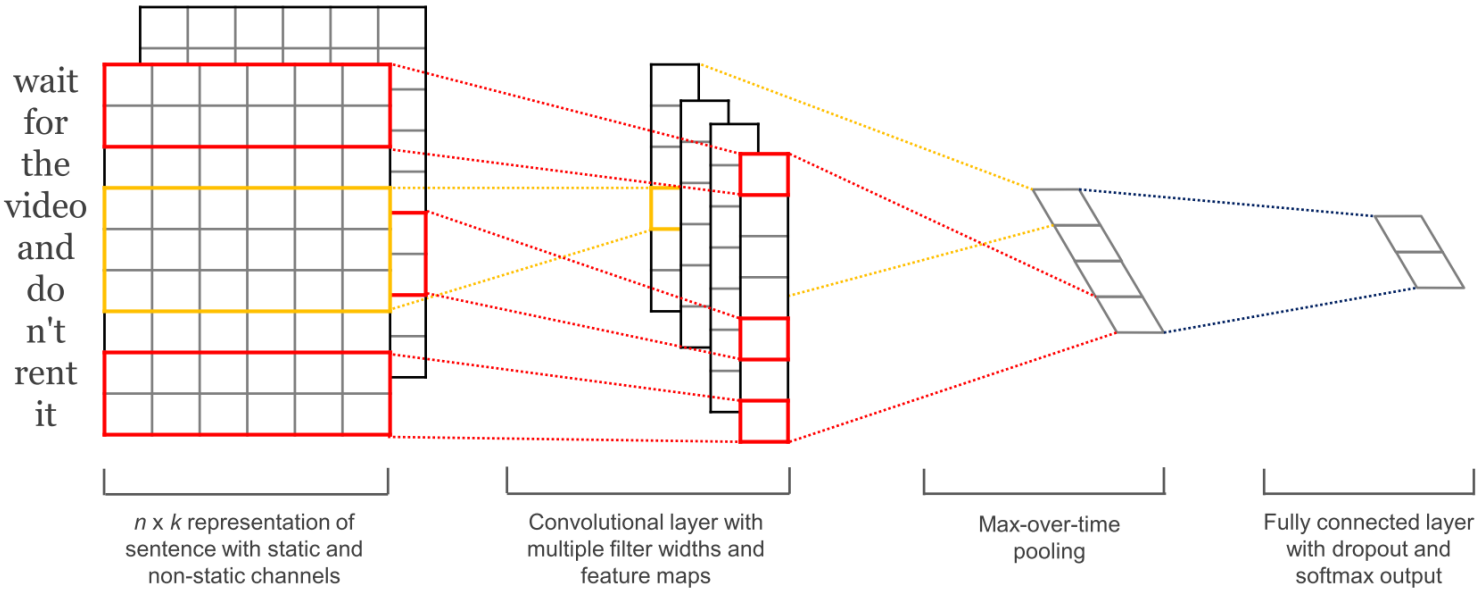
- Python API
- Portable

  deploy computation to CPUs or GPUs in a desktop, server or mobile devices with a single API
- Flexible

  compatible with Raspberry Pi, Android, Windows, iOS, Linux to server farms
- Visualization *TensorBoard*
- Checkpoints *for managing experiments*
- Auto-differentiation
- Large community $> 10,000$ *commits and* > 3000 *TF-related repos in a year*

Neural Style Transfer

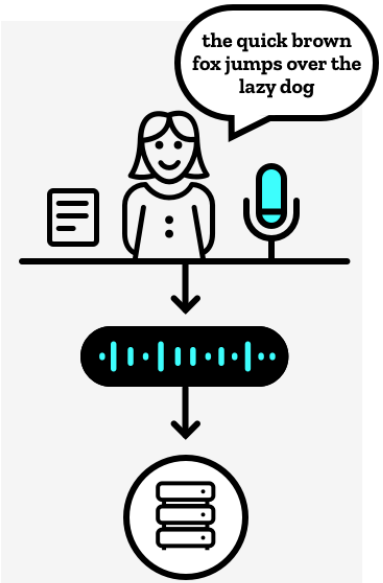
Sentence Classification



ML@B

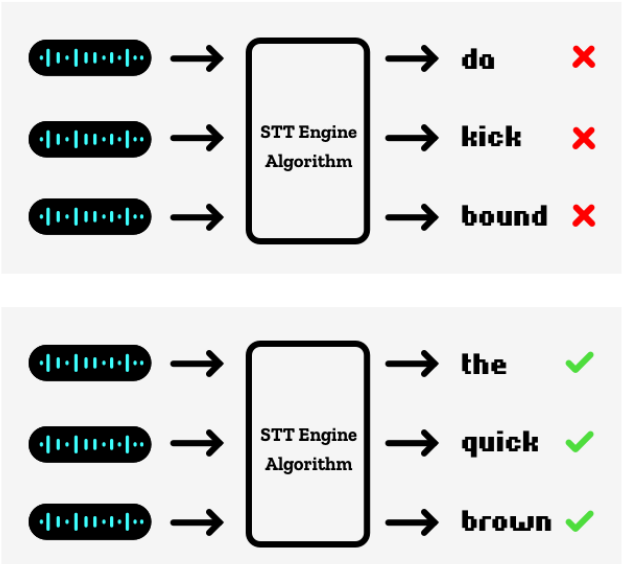
How a Speech Application Learns

Step 1. Record voices



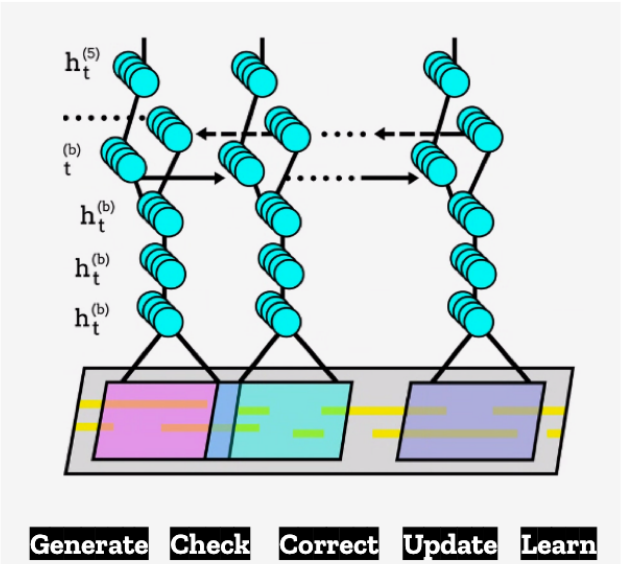
Common Voice Project

Step 2. Input voice data



Open Source STT Engine

Step 3. Train the speech algorithm



Deep Learning Architecture

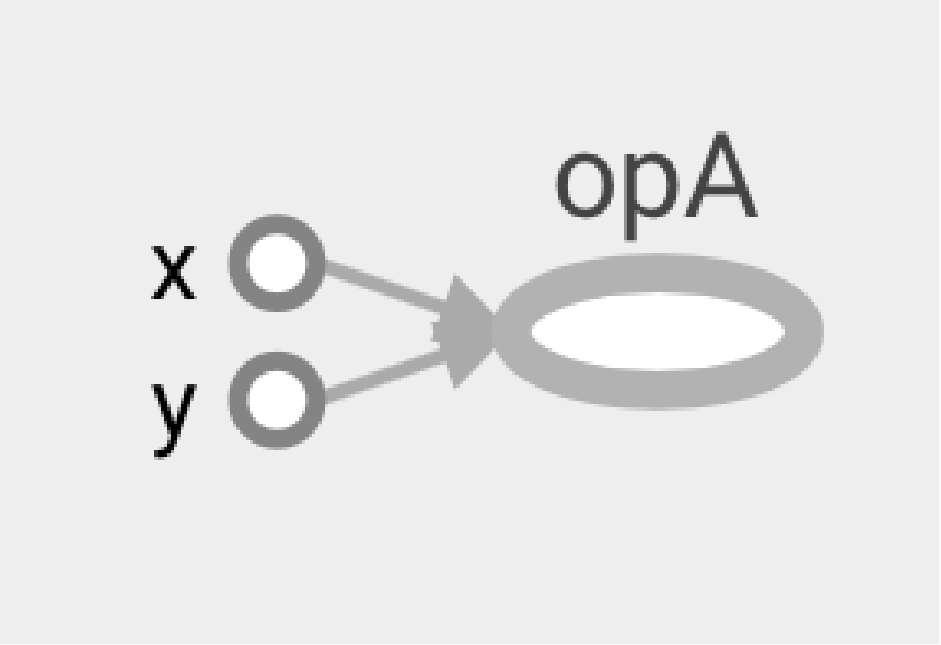
Computation graphs and sessions

Data Flow Graphs

```
import tensorflow as tf
a = tf.add(1, 2)
```

Nodes: operators, variables and constants

Edges: tensors

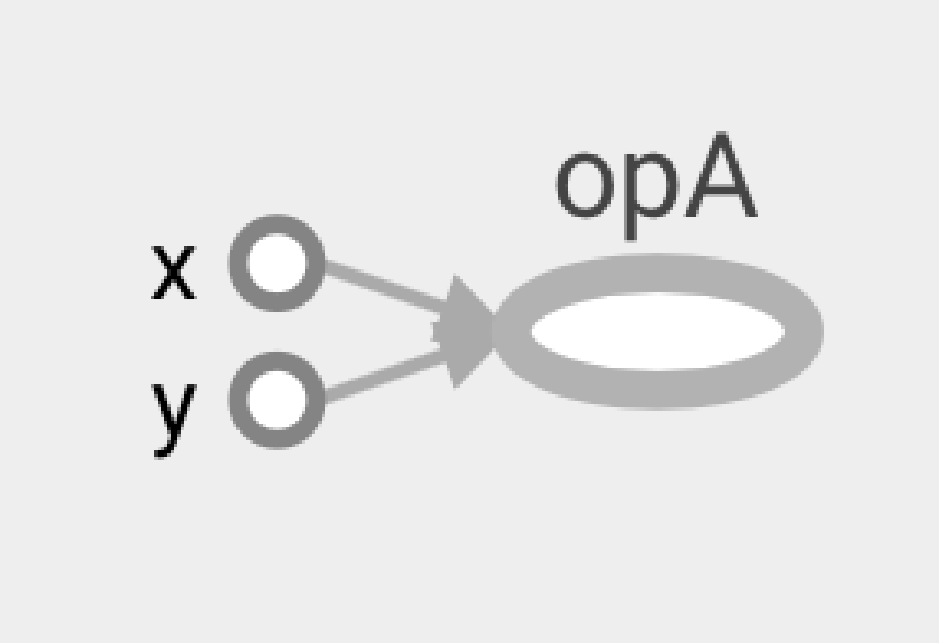


Data Flow Graphs

```
import tensorflow as tf
a = tf.add(1, 2)
print(a)
```

```
>>Tensor("Add:0",
shape=(), dtype=int32)
```

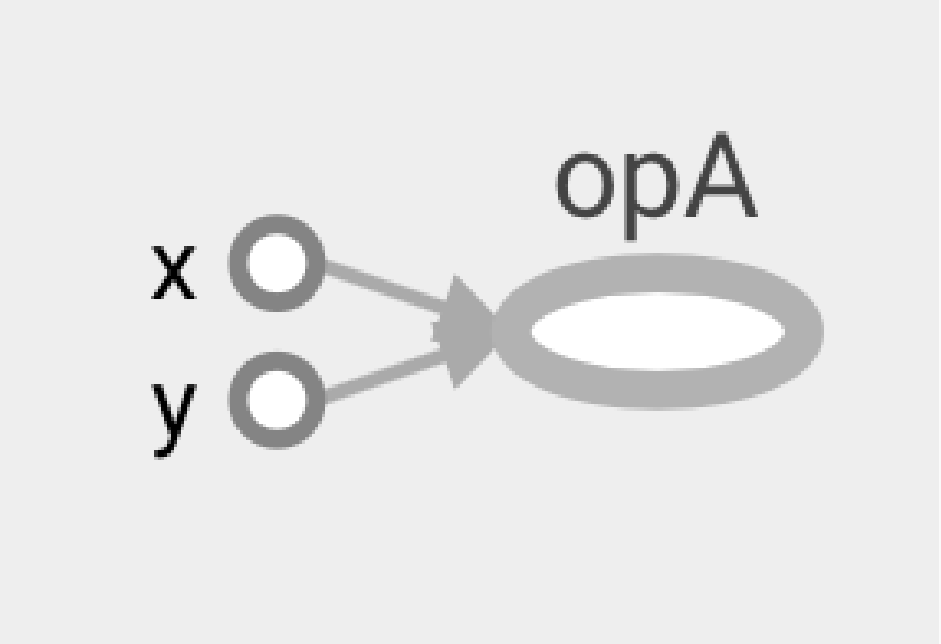
(Not 3)



How to get the value of a?

- Create a **session**
- Assign it to variable sess so we can call it later
- Evaluate the graph in the session to get the value of a

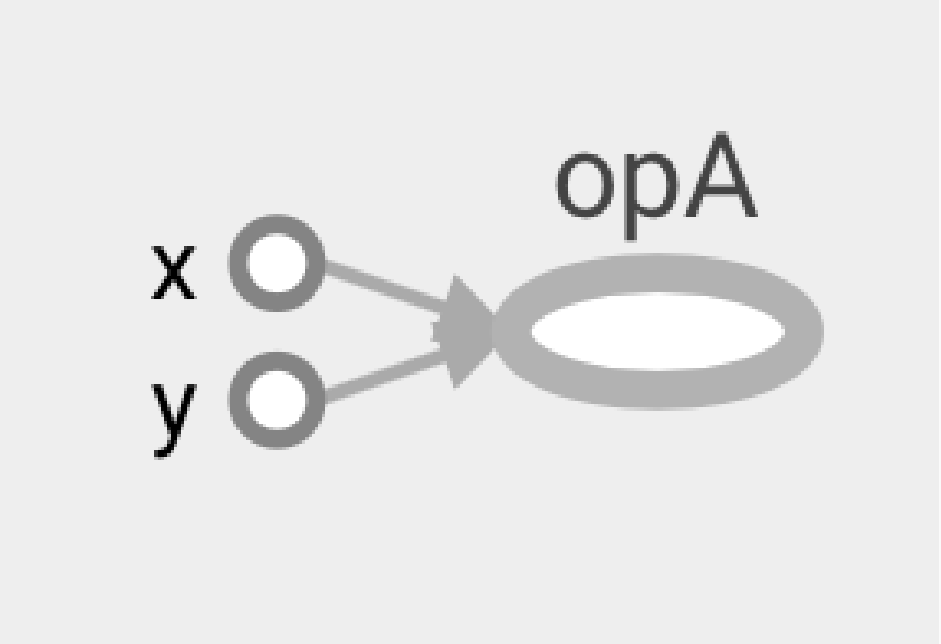
```
import tensorflow as tf
a = tf.add(1, 2)
sess = tf.Session()
print(sess.run(a))
sess.close()

>> 3
```



How to get the value of a?

- Create a **session**
- Assign it to variable sess so we can call it later
- Evaluate the graph in the session to get the value of a

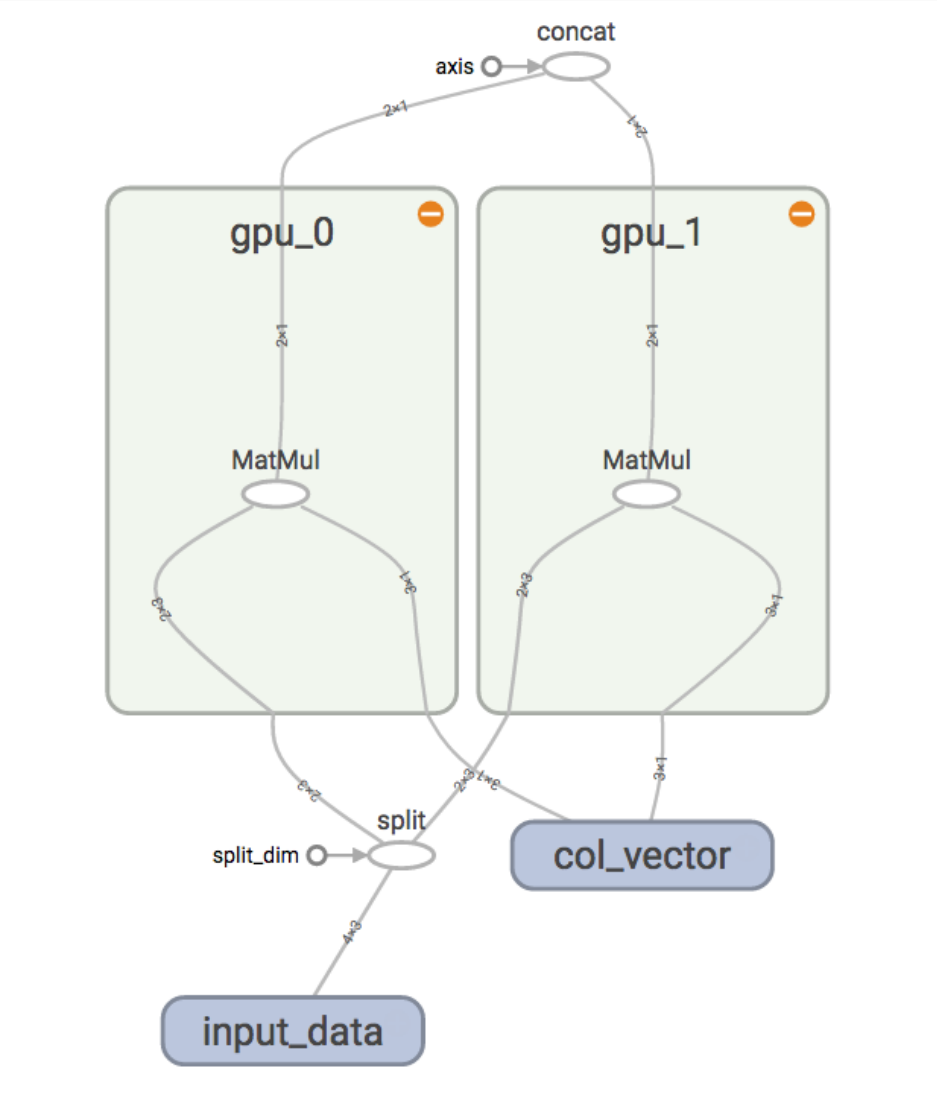
```
import tensorflow as tf
a = tf.add(1, 2)
~~sess = tf.Session()~~
with tf.Session() as sess:
    print(sess.run(a))
~~sess.close()~~
```



Why graphs?

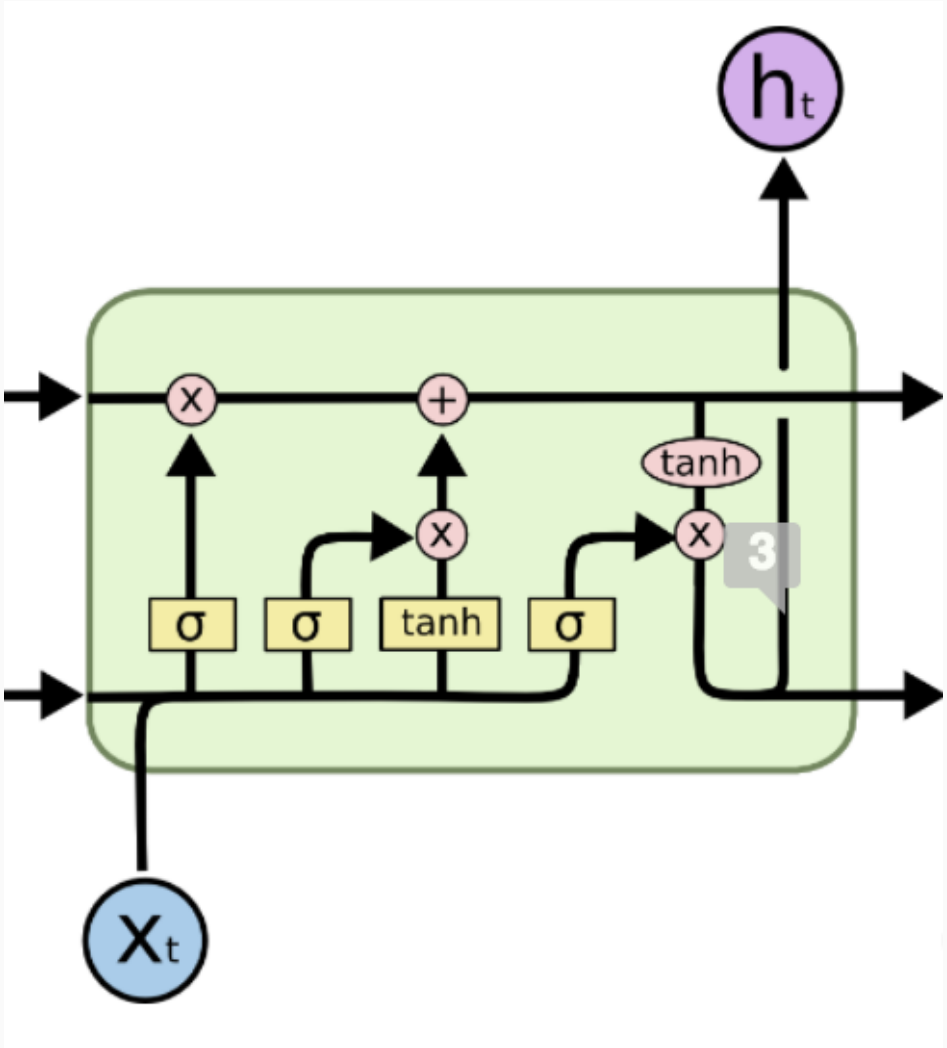
Why graphs?

Break graphs into subgraphs and run them across multiple CPUs, GPUs or devices in parallel



Why graphs?

- Distributed computation
- Save computation
  only run subgraphs that evaluate to the target value
- Divide and conquer
  break computation into small, differential pieces to facilitate auto-differentiation
- Most common framework to visualize ML models



Constants, Variables, Placeholders

Constants

```
tf.constant(value, dtype=None, shape=None,
name='Const', verify_shape=False)

import tensorflow as tf
a = tf.constant([1,2], name="a")
b = tf.constant([[3, 4], [5, 6]], name="b")
x = tf.add(a, b, name="add") y = tf.multiply(a, b,
name="mul")
with tf.Session() as sess:
    x, y = sess.run([x, y])
    print(x,y)

>> [4 6][6 8]
```

What's wrong with constants?

Constants are stored in the graph definition

```
import tensorflow as tf
a = tf.constant([1,2], name="a")
with tf.Session() as sess:
    print(sess.graph.as_graph_def())

# will see a stored in the graph's definition
```

What's wrong with constants?

ML@B

- When constants are big, loading graphs become expensive
- Only use **constants** for primitive types.
- Use **variables** for data that requires more memory

Basic Tensorflow Program Structure

Step 1: Read in Data

ML@B

- Can load dataset from CSV
- Tensorflow has nice interface for MNIST that provides many convenient methods

Step 2: Define Placeholders

ML@B

- Placeholders act as the "input" markers for our TF graph
- Need to define placeholders for features X and corresponding labels Y
- First dimension of X and Y can be set to **None** - signals to graph constructor that we want to accept variable batch sizes to our model
- Placeholders not just limited to data - sometimes we can use them for other values like dropout probability

Step 3: Define Variables for Each Layer

ML@B

- Variables hold the parameters of the model - trainable values that are updated during optimization
- Weights - typically initialized from a normal random distribution to break symmetry in backprop step
- Biases - typically initialized as slightly positive

Step 4: Construct Each Layer

We limit our focus to fully-connected layers for now

We need to construct the following function for each layer:

$$\sigma(X\dot{W} + b)$$

- σ is activation function for the layer (eg: relu, sigmoid)
- X is the input for the previous layer (or the input data for the first layer). The size will be [batch size, num input features]
- W are the weights and will be of size [num input features, num output features]
- b is the bias and will be of size [1, num output features]

Step 5: Loss Function

Define objective function that takes in the Y placeholder and compares to the prediction $\hat{Y}$

- Mean Squared Error $L = \|_{\|} 2Y - \hat{Y}^2$
- Cross Entropy $L = -\Sigma_i (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)$
- Some popular loss functions are already provided

Step 6: Define Optimizer

ML@B

- Takes in learning rate as parameter
- Set the optimizer to minimize the loss function
- If loss is failing to converge with each epoch of training or appearing as nan, try decreasing the learning rate

Step 7: Create a Session

ML@B

- Can use InteractiveSession() for iPython notebooks and Session() for Python scripts
- Make sure to initialize all global variables

Step 8: Train the Model

ML@B

- For each epoch, go through all batches of training data and run the optimizer
- Aggregate loss over all batches in each epoch - print this value after every few epochs
- Can use Tensorboard to check if model is converging

Step 9: Test Model

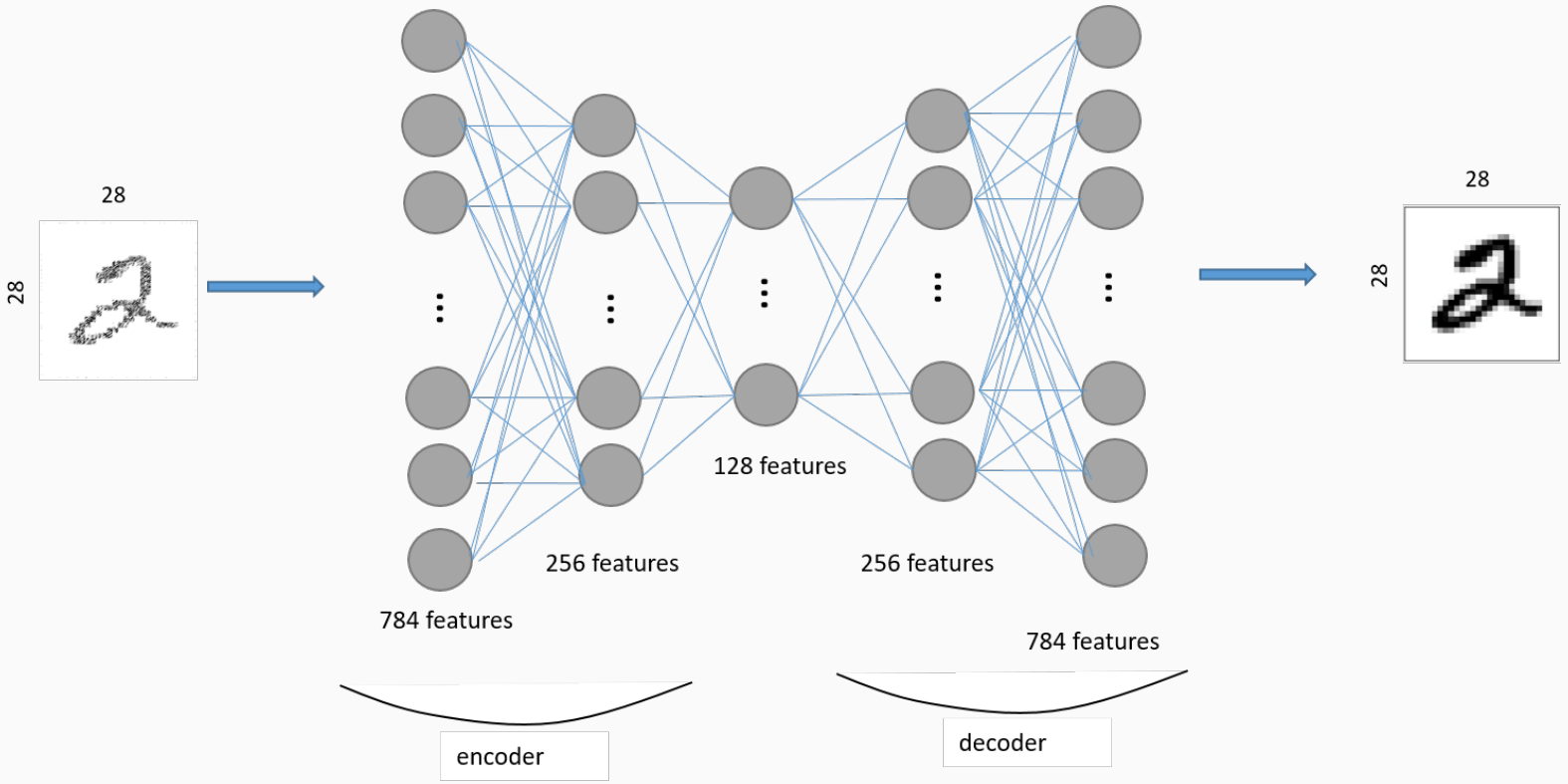
ML@B

- Depends on type of problem you are trying to solve
- Can include printing accuracy, plotting regression line, displaying generated data, etc.

Feed Forward Newtork on MNIST

Lab

Extra Exercise - Denoising Autoencoder



Questions

Additional Resources

ML@B

- https://github.com/mlberkeley/tensorflow-bootcamp
- https://github.com/mlberkeley/Bootcamp-Spring-2018

Feedback

ML@B

- https://tinyurl.com/tfworkshop131