



# Welcome to TensorFlow!

CS 20: TensorFlow for Deep Learning Research

Lecture 1

1/12/2018



# Agenda

Welcome

Overview of TensorFlow

Graphs and Sessions

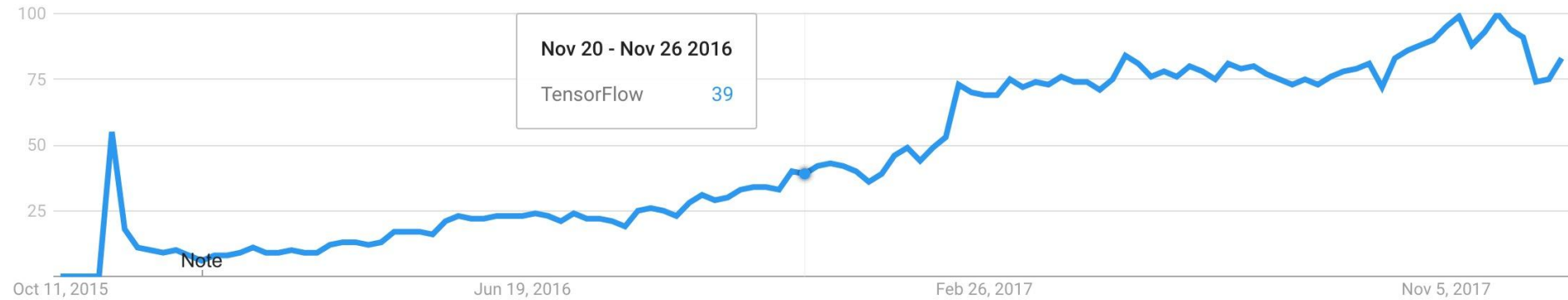


# What's TensorFlow™?

“Open source software library for  
numerical computation using data flow graphs”

# Launched Nov 2015

Interest over time ?



# Why TensorFlow?

- Many machine learning libraries



**Denny Britz** @dennybritz · 25 Dec 2017



I'm going through my newsletters to write up a year-end summary of developments and achievements in AI.

Fun fact: Almost every week, a company released a new generic or task-specific Deep Learning “framework” 😄

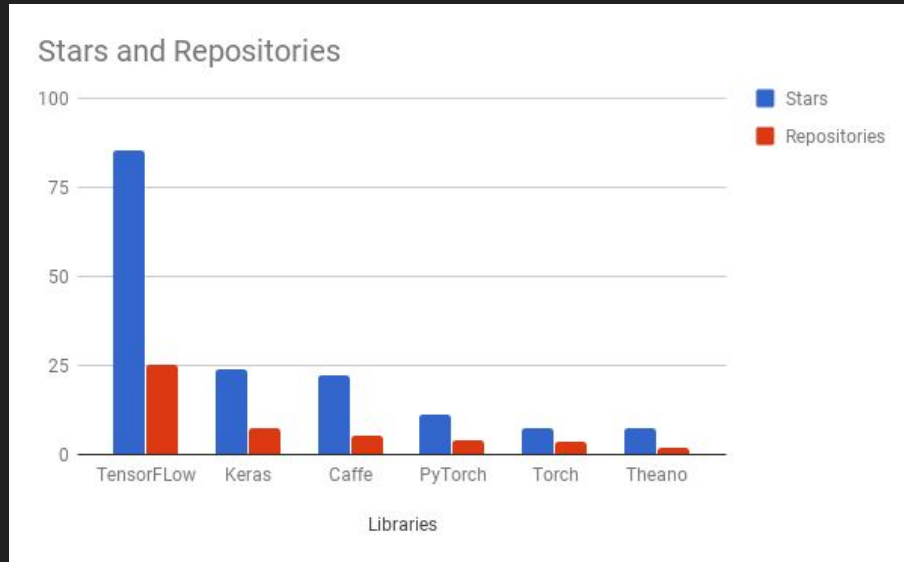
# Why TensorFlow?

- Flexibility + Scalability

Originally developed by Google as a single infrastructure for machine learning in both production and research

# Why TensorFlow?

- Flexibility + Scalability
- Popularity





# Companies using TensorFlow

Google

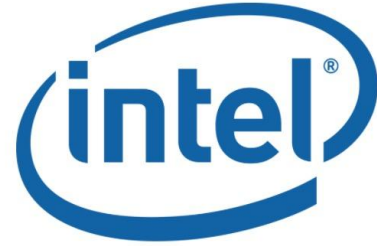


OpenAI

AIRBUS

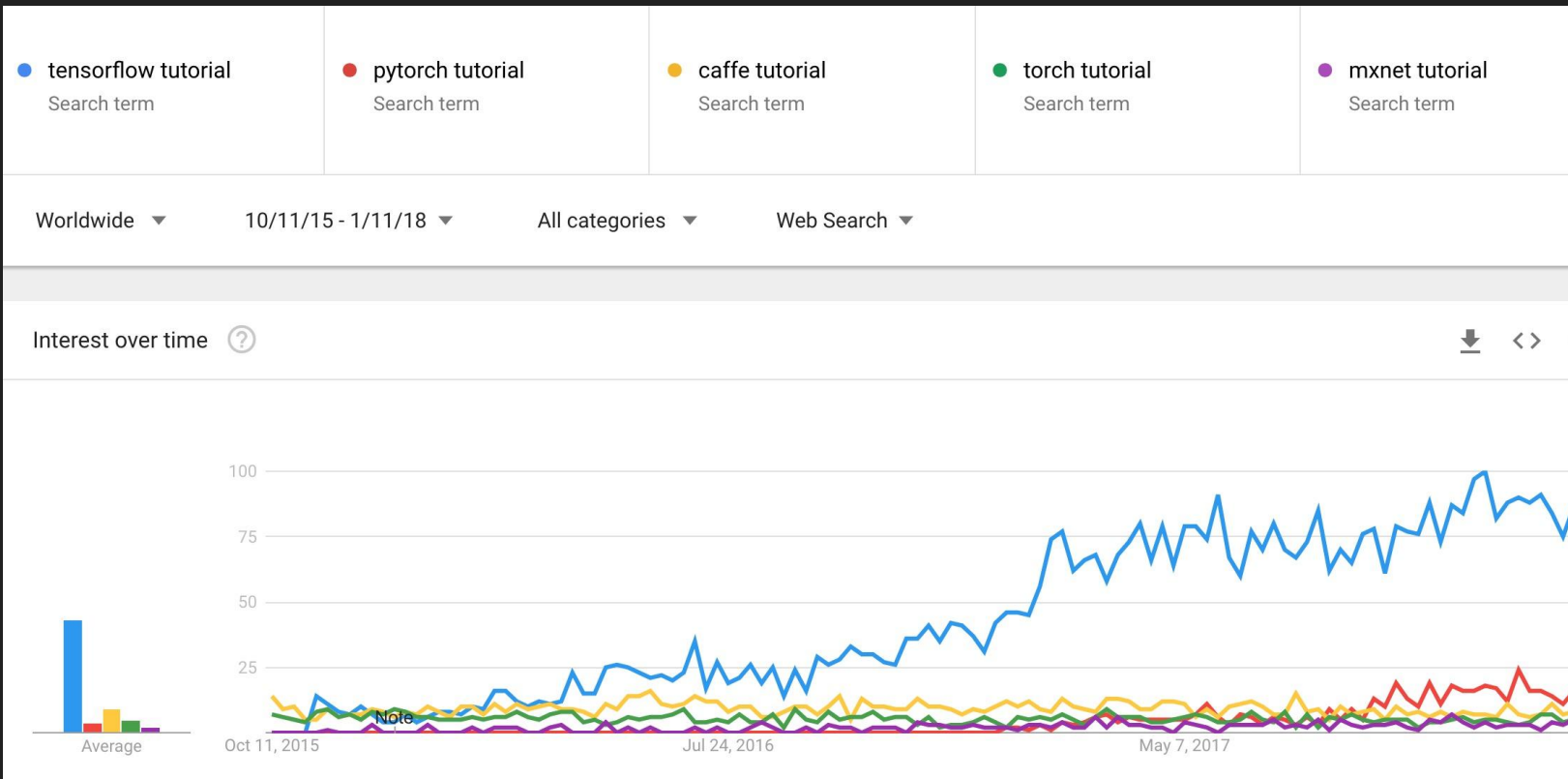


Google DeepMind



NVIDIA®

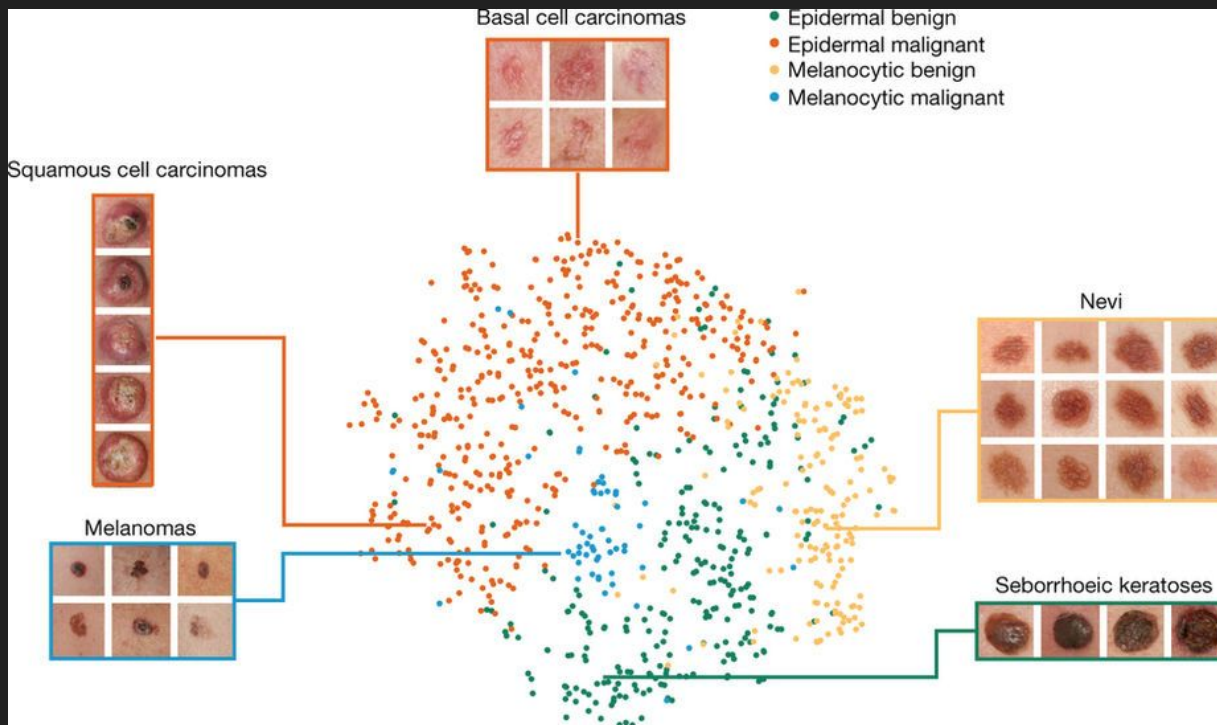
# Demand for tutorials on TensorFlow





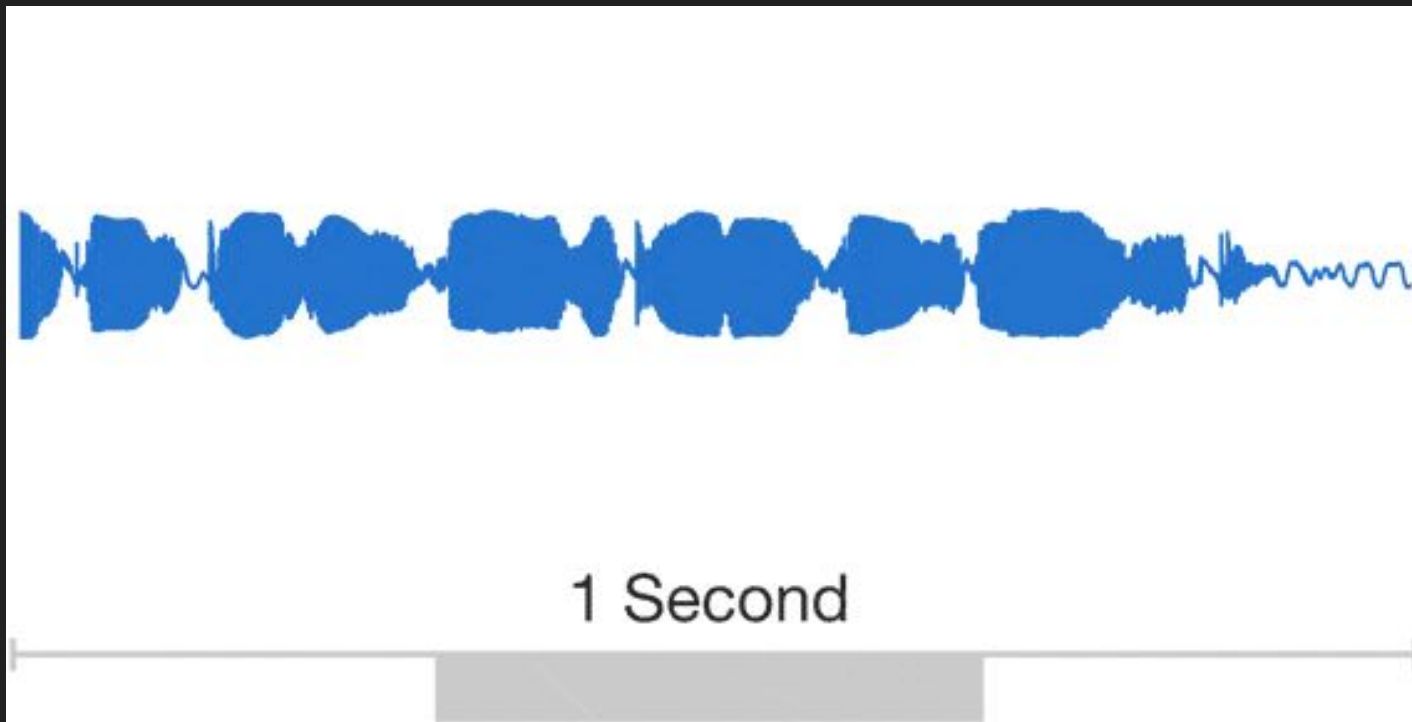
# Some cool projects using TensorFlow

# Classify skin cancer



# WaveNet: Text to Speech

It takes several hours to synthesize 1 second!



# Drawing



# Neural Style Translation



Image Style Transfer Using Convolutional Neural Networks (Gatys et al., 2016)  
Tensorflow adaptation by Cameroon Smith (cysmith@github)



I hope that this class will give you  
the tool to build cool projects like  
those!



# Goals

- Understand TF's computation graph approach
- Explore TF's built-in functions and classes
- Learn how to build and structure models best suited for a deep learning project



CS20

# Staff



Chip Huyen  
huyenn@stanford.edu



Michael Straka  
mstraka2@stanford.edu



Pedro Garzon  
pgarzon@stanford.edu

# Logistics

- Piazza: [piazza.com/stanford/winter2018/cs20](https://piazza.com/stanford/winter2018/cs20)
- Staff email: [cs20-win1718-staff@lists.stanford.edu](mailto:cs20-win1718-staff@lists.stanford.edu)
- Students mailing list: [cs20-win1718-students](#)
- Guests mailing list: [cs20-win1718-guests](#)

# Grading

- Assignments (3)
- Participation
- Check in

# Resources

- [The official documentations](#)
- [TensorFlow's official sample models](#)
- StackOverflow should be your first port of call in case of bug
- Books
  - Aurélien Géron's Hands-On Machine Learning with Scikit-Learn and TensorFlow (O'Reilly, March 2017)
  - François Chollet's Deep Learning with Python (Manning Publications, November 2017)
  - Nishant Shukla's Machine Learning with TensorFlow (Manning Publications, January 2018)
  - Lieder et al.'s Learning TensorFlow A Guide to Building Deep Learning Systems (O'Reilly, August 2017)

# Permission Number

[Link](#)



Many of you are ahead of me in academia so I probably need more of your help than you do mine





# Getting Started

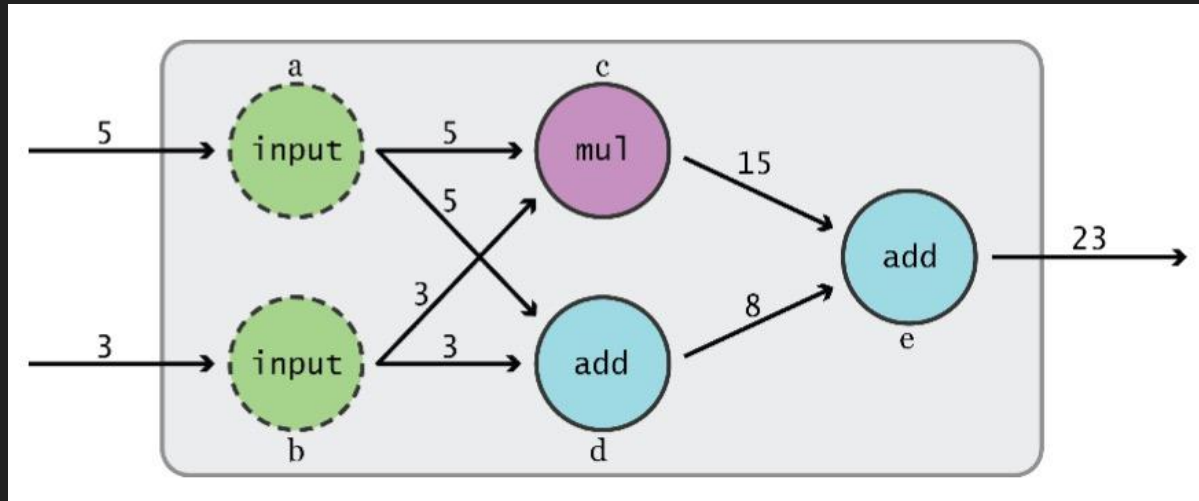
```
import tensorflow as tf
```



# Graphs and Sessions

# Data Flow Graphs

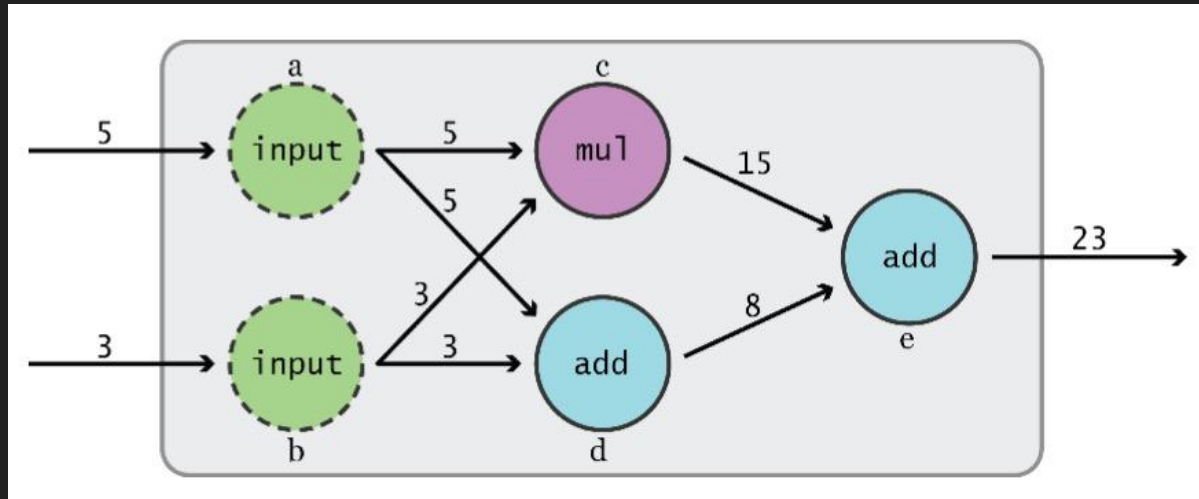
TensorFlow separates definition of computations from their execution



# Data Flow Graphs

Phase 1: assemble a graph

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

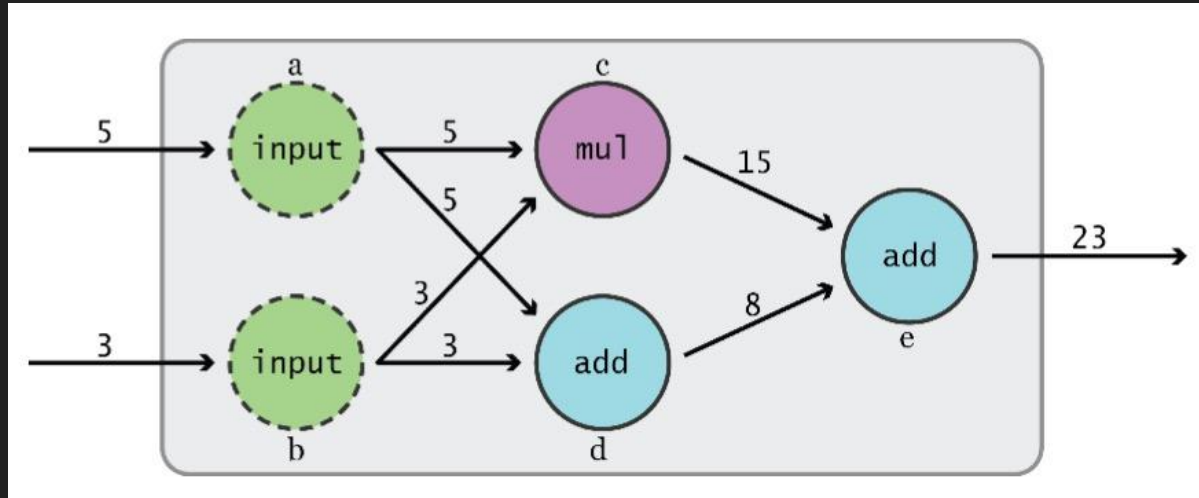


# Data Flow Graphs

Phase 1: assemble a graph

This might change in the future with eager mode!!

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



# What's a tensor?

# What's a tensor?

## An n-dimensional array

0-d tensor: scalar (number)

1-d tensor: vector

2-d tensor: matrix

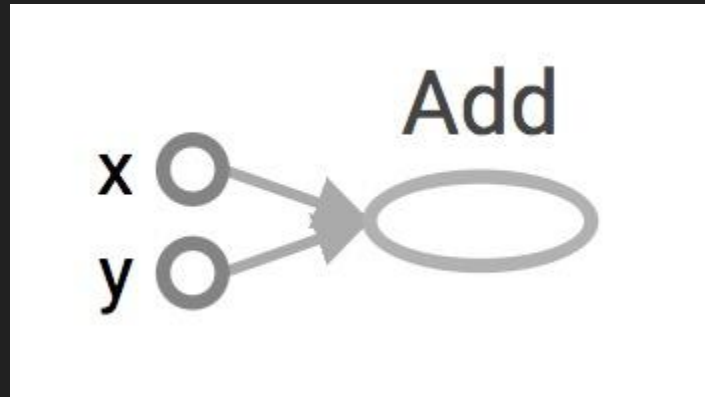
and so on



# Data Flow Graphs

```
import tensorflow as tf  
a = tf.add(3, 5)
```

Visualized by TensorBoard



# Data Flow Graphs

```
import tensorflow as tf  
a = tf.add(3, 5)
```

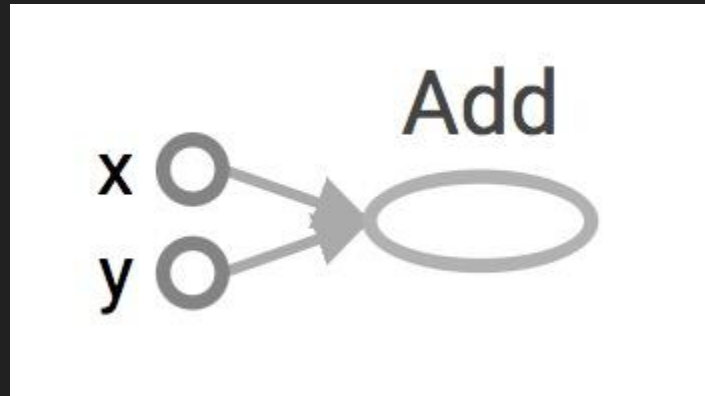
Why x, y?

TF automatically names the nodes when you don't explicitly name them.

x = 3

y = 5

Visualized by TensorBoard

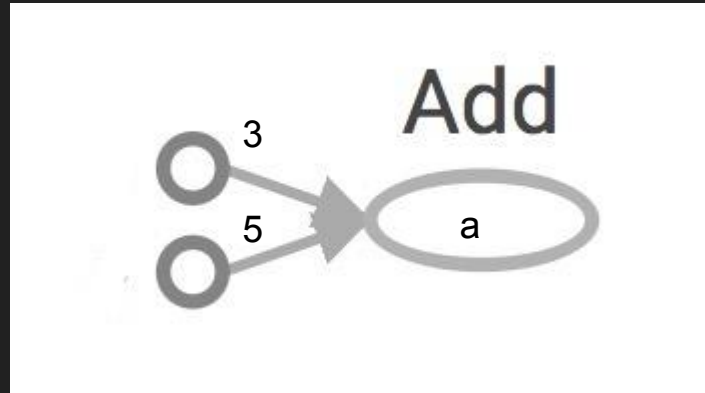


# Data Flow Graphs

```
import tensorflow as tf  
a = tf.add(3, 5)
```

Nodes: operators, variables, and constants  
Edges: tensors

Interpreted?



# Data Flow Graphs

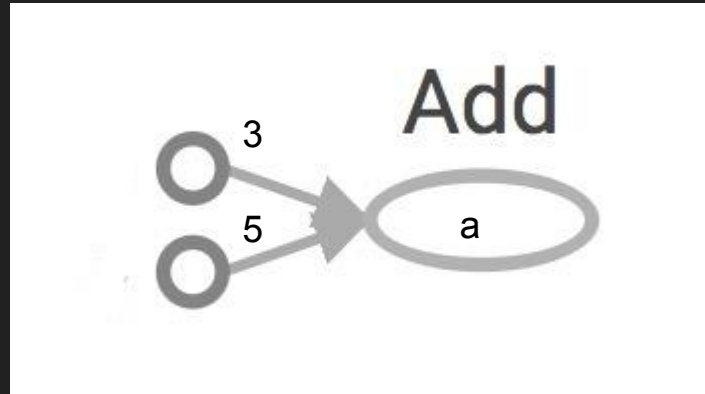
```
import tensorflow as tf  
a = tf.add(3, 5)
```

Nodes: operators, variables, and constants  
Edges: tensors

Tensors are data.

TensorFlow = tensor + flow = data + flow  
(I know, mind=blown)

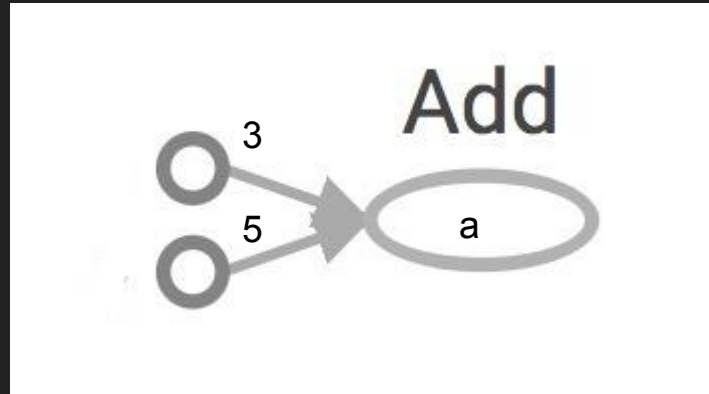
Interpreted?



# Data Flow Graphs

```
import tensorflow as tf  
a = tf.add(3, 5)  
print(a)
```

```
>> Tensor("Add:0", shape=(), dtype=int32)  
(Not 8)
```



# How to get the value of a?

Create a **session**, assign it to variable sess so we can call it later

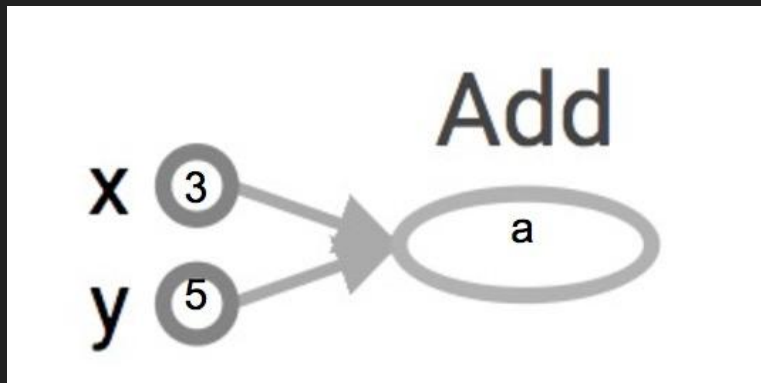
Within the session, evaluate the graph to fetch the value of a

# How to get the value of a?

Create a **session**, assign it to variable `sess` so we can call it later

Within the session, evaluate the graph to fetch the value of `a`

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print(sess.run(a))
sess.close()
```



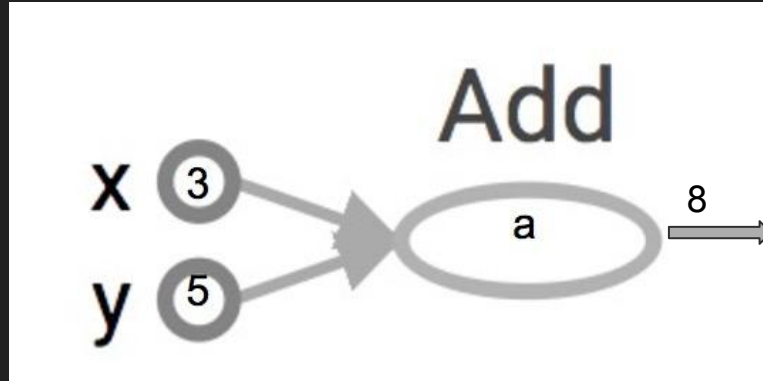
The session will look at the graph, trying to think: hmm, how can I get the value of `a`, then it computes all the nodes that leads to `a`.

# How to get the value of a?

Create a **session**, assign it to variable `sess` so we can call it later

Within the session, evaluate the graph to fetch the value of `a`

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
print(sess.run(a))      >> 8
sess.close()
```



The session will look at the graph, trying to think: hmm, how can I get the value of `a`, then it computes all the nodes that leads to `a`.

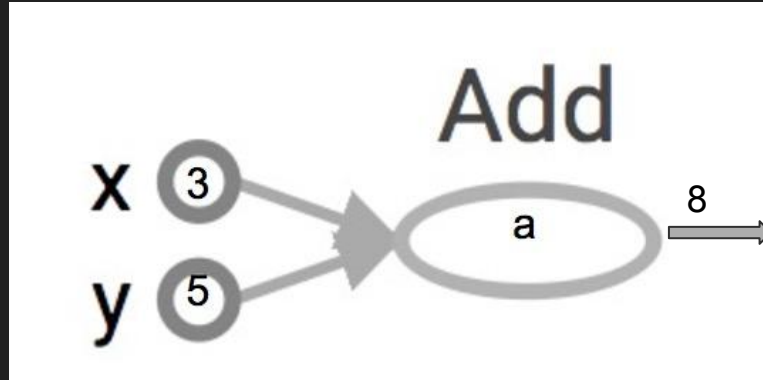


# How to get the value of a?

Create a **session**, assign it to variable `sess` so we can call it later

Within the session, evaluate the graph to fetch the value of `a`

```
import tensorflow as tf
a = tf.add(3, 5)
sess = tf.Session()
with tf.Session() as sess:
    print(sess.run(a))
sess.close()
```



# `tf.Session()`

A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

# tf.Session()

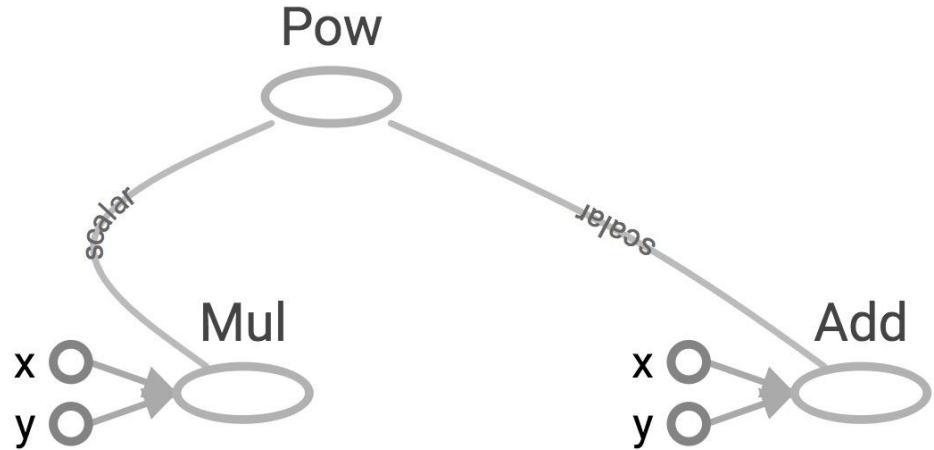
A Session object encapsulates the environment in which Operation objects are executed, and Tensor objects are evaluated.

Session will also allocate memory to store the current values of variables.

# More graph

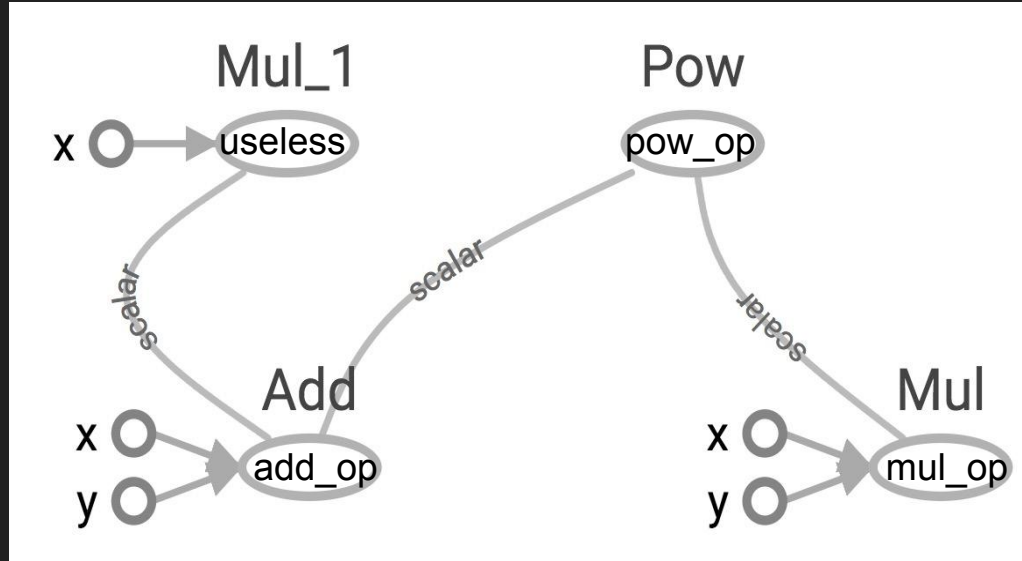
Visualized by TensorBoard

```
x = 2
y = 3
op1 = tf.add(x, y)
op2 = tf.multiply(x, y)
op3 = tf.pow(op2, op1)
with tf.Session() as sess:
    op3 = sess.run(op3)
```



# Subgraphs

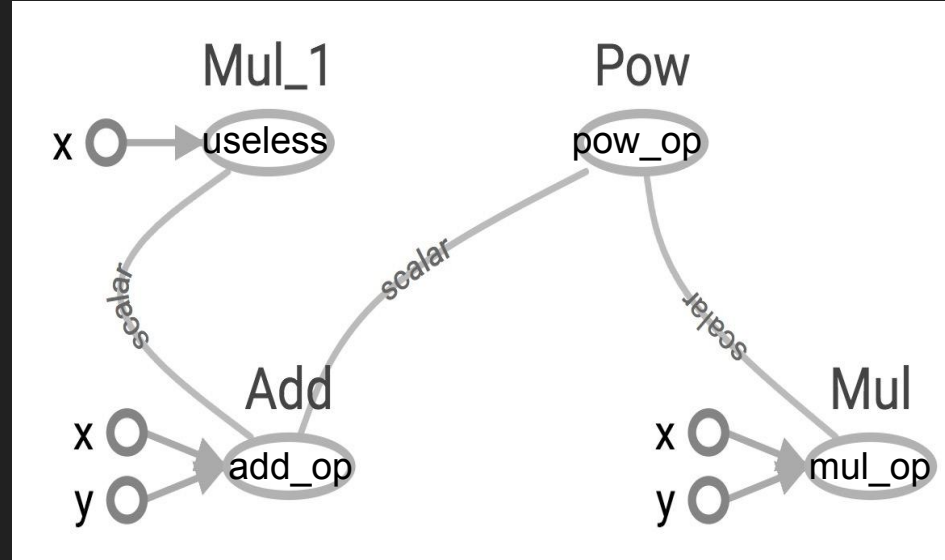
```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
    z = sess.run(pow_op)
```



Because we only want the value of pow\_op and pow\_op doesn't depend on useless, session won't compute value of useless  
→ save computation

# Subgraphs

```
x = 2
y = 3
add_op = tf.add(x, y)
mul_op = tf.multiply(x, y)
useless = tf.multiply(x, add_op)
pow_op = tf.pow(add_op, mul_op)
with tf.Session() as sess:
    z, not_useless = sess.run([pow_op, useless])
```



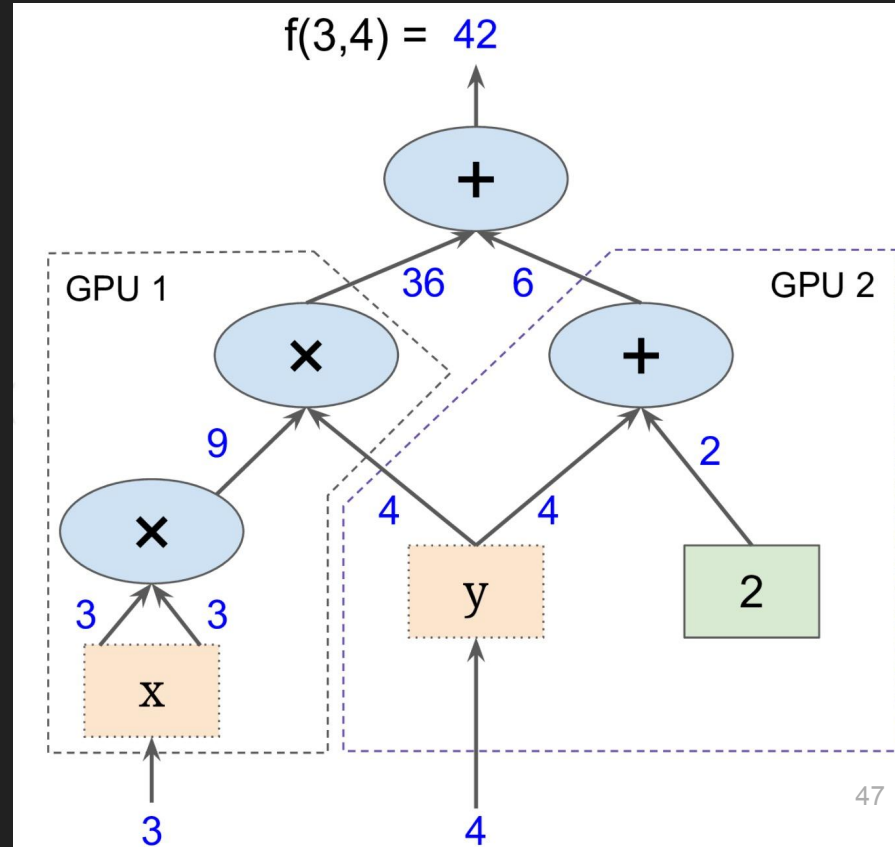
```
tf.Session.run(fetches,
                feed_dict=None,
                options=None,
                run_metadata=None)
```

fetches is a list of tensors whose values you want

# Subgraphs

Possible to break graphs into several chunks and run them parallelly across multiple CPUs, GPUs, TPUs, or other devices

Example: AlexNet



# Distributed Computation

To put part of a graph on a specific CPU or GPU:

```
# Creates a graph.
with tf.device('/gpu:2'):
    a = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='a')
    b = tf.constant([1.0, 2.0, 3.0, 4.0, 5.0, 6.0], name='b')
    c = tf.multiply(a, b)

# Creates a session with log_device_placement set to True.
sess = tf.Session(config=tf.ConfigProto(log_device_placement=True))

# Runs the op.
print(sess.run(c))
```



**What if I want to build more  
than one graph?**

**You can  
but you don't need more than one graph  
The session runs the default graph**

**But what if I really want to?**

**URGH, NO**

## BUG ALERT!

- Multiple graphs require multiple sessions, each will try to use all available resources by default
- Can't pass data between them without passing them through python/numpy, which doesn't work in distributed
- It's better to have disconnected subgraphs within one graph

**I insist ...**

# tf.Graph()

create a graph:

```
g = tf.Graph()
```

# tf.Graph()

to add operators to a graph, set it as default:

```
g = tf.Graph()
with g.as_default():
    x = tf.add(3, 5)
sess = tf.Session(graph=g)
with tf.Session() as sess:
    sess.run(x)
```



# tf.Graph()

To handle the default graph:

```
g = tf.get_default_graph()
```

# tf.Graph()

Do not mix default graph and user created graphs

```
g = tf.Graph()
```

```
# add ops to the default graph
```

```
a = tf.constant(3)
```

```
# add ops to the user created graph
```

```
with g.as_default():
```

```
    b = tf.constant(5)
```

Prone to errors

# tf.Graph()

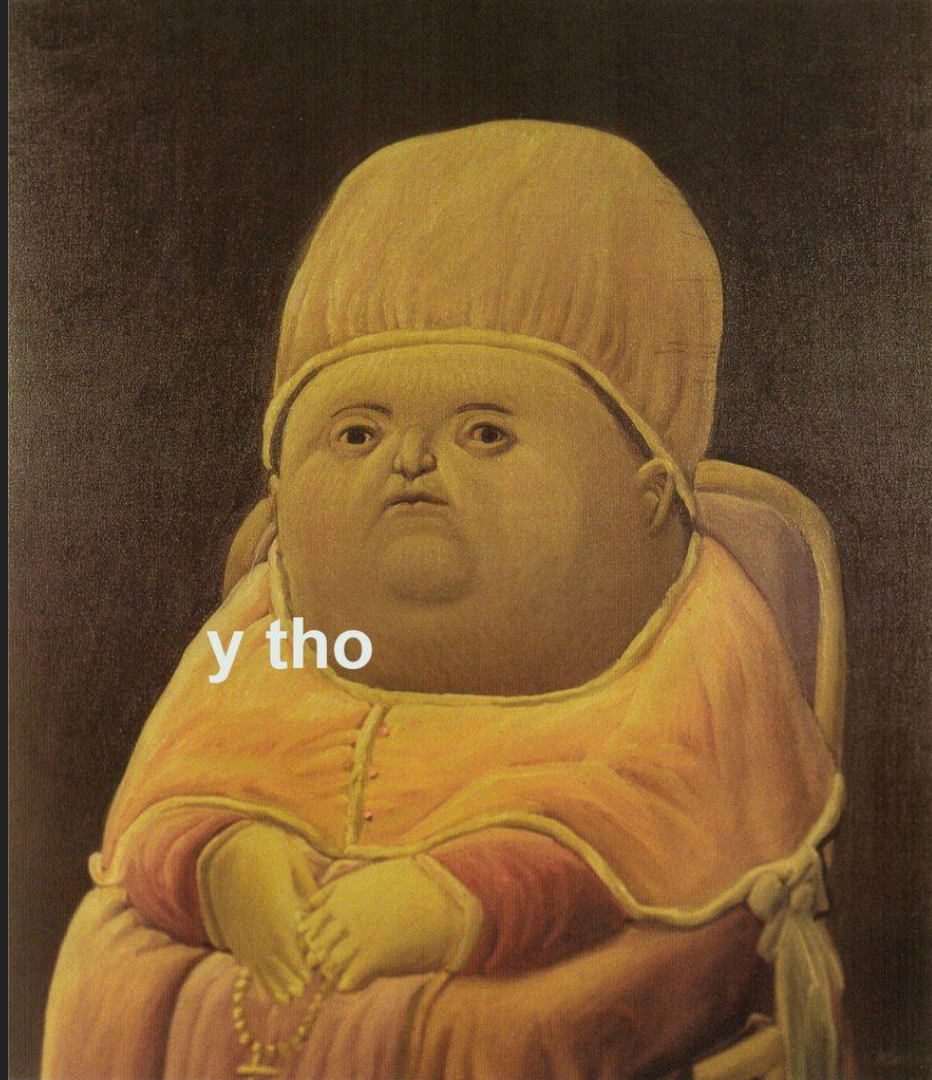
Do not mix default graph and user created graphs

```
g1 = tf.get_default_graph()
g2 = tf.Graph()

# add ops to the default graph
with g1.as_default():
    a = tf.Constant(3)

# add ops to the user created graph
with g2.as_default():
    b = tf.Constant(5)
```

Better  
But still not good enough because no more than  
one graph!



# Why graphs

1. Save computation. Only run subgraphs that lead to the values you want to fetch.

# Why graphs

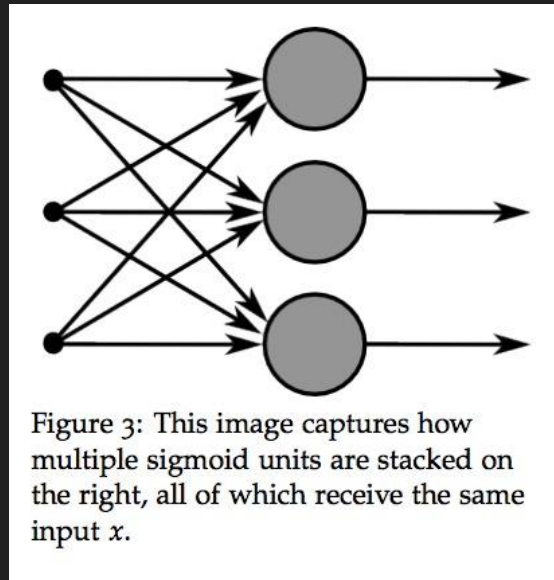
1. Save computation. Only run subgraphs that lead to the values you want to fetch.
2. Break computation into small, differential pieces to facilitate auto-differentiation

# Why graphs

1. Save computation. Only run subgraphs that lead to the values you want to fetch.
2. Break computation into small, differential pieces to facilitate auto-differentiation
3. Facilitate distributed computation, spread the work across multiple CPUs, GPUs, TPUs, or other devices

# Why graphs

1. Save computation. Only run subgraphs that lead to the values you want to fetch.
2. Break computation into small, differential pieces to facilitate auto-differentiation
3. Facilitate distributed computation, spread the work across multiple CPUs, GPUs, TPUs, or other devices
4. Many common machine learning models are taught and visualized as directed graphs



A neural net graph from Stanford's CS224N course



# Next class

Basic operations

Constants and variables

Data pipeline

Fun with TensorBoard

Feedback: [huyenn@stanford.edu](mailto:huyenn@stanford.edu)

Thanks!