



Welcome to TensorFlow!

CS 20SI:
TensorFlow for Deep Learning Research
Lecture 1
1/13/2017

Agenda

Welcome

Overview of TensorFlow

Graphs and Sessions



Instructor

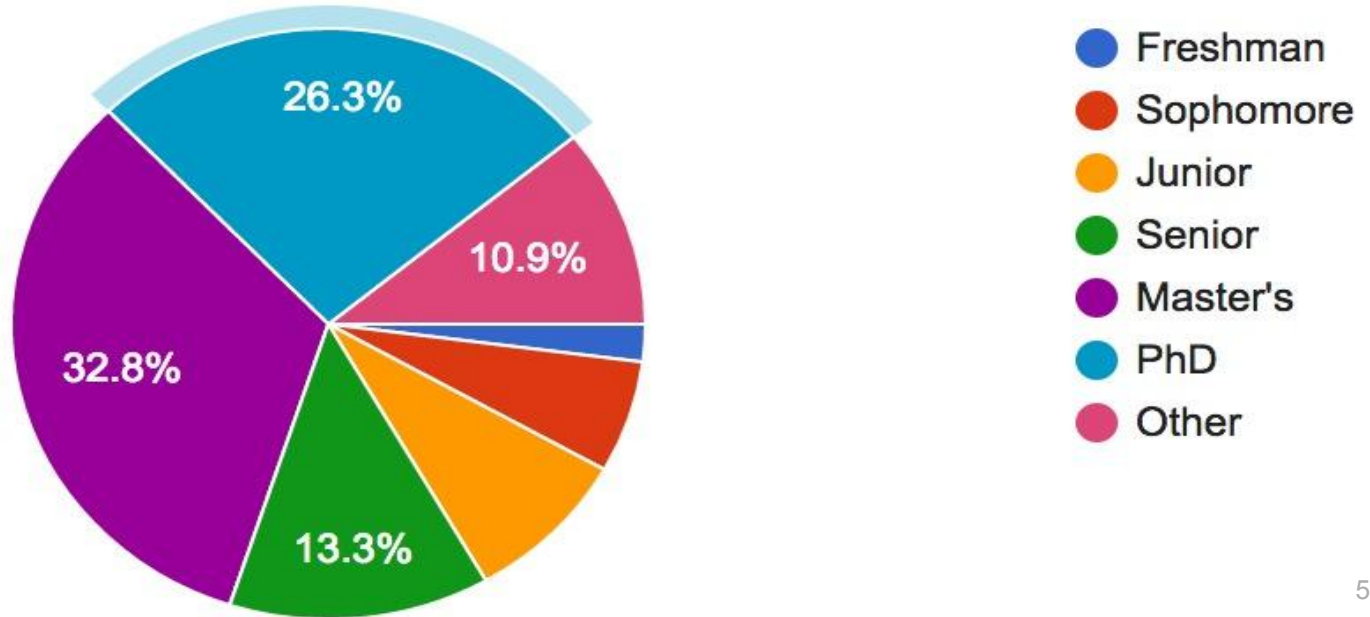


Chip Huyen

huyenn@stanford.edu

You

Year (293 responses)



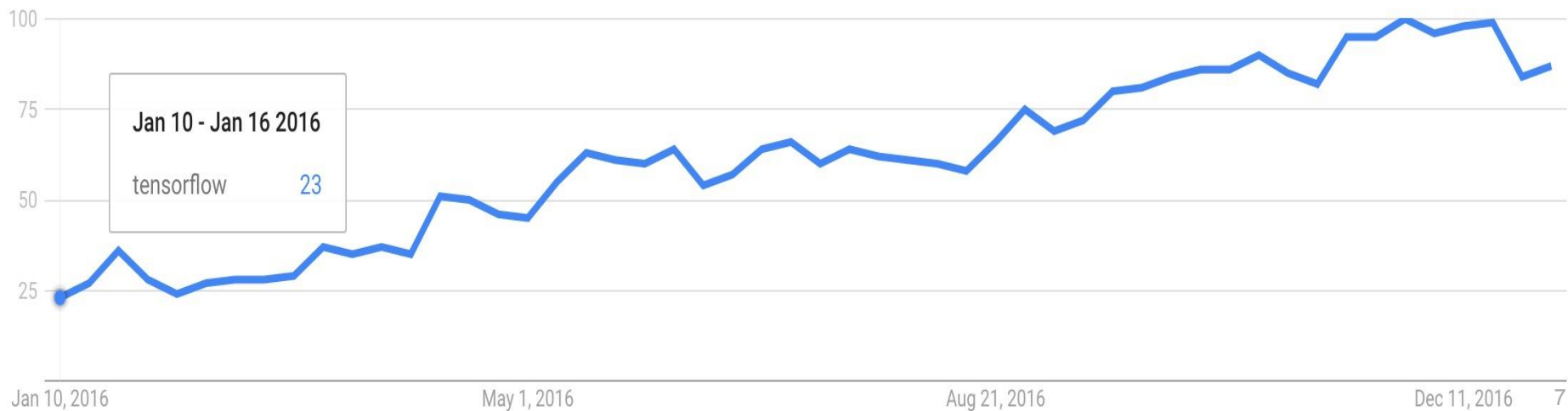
What's TensorFlow™?

- Open source software library for numerical computation using data flow graphs
- Originally developed by Google Brain Team to conduct machine learning and deep neural networks research
- General enough to be applicable in a wide variety of other domains as well

TensorFlow provides an extensive suite of functions and classes that allow users to build various models from scratch.

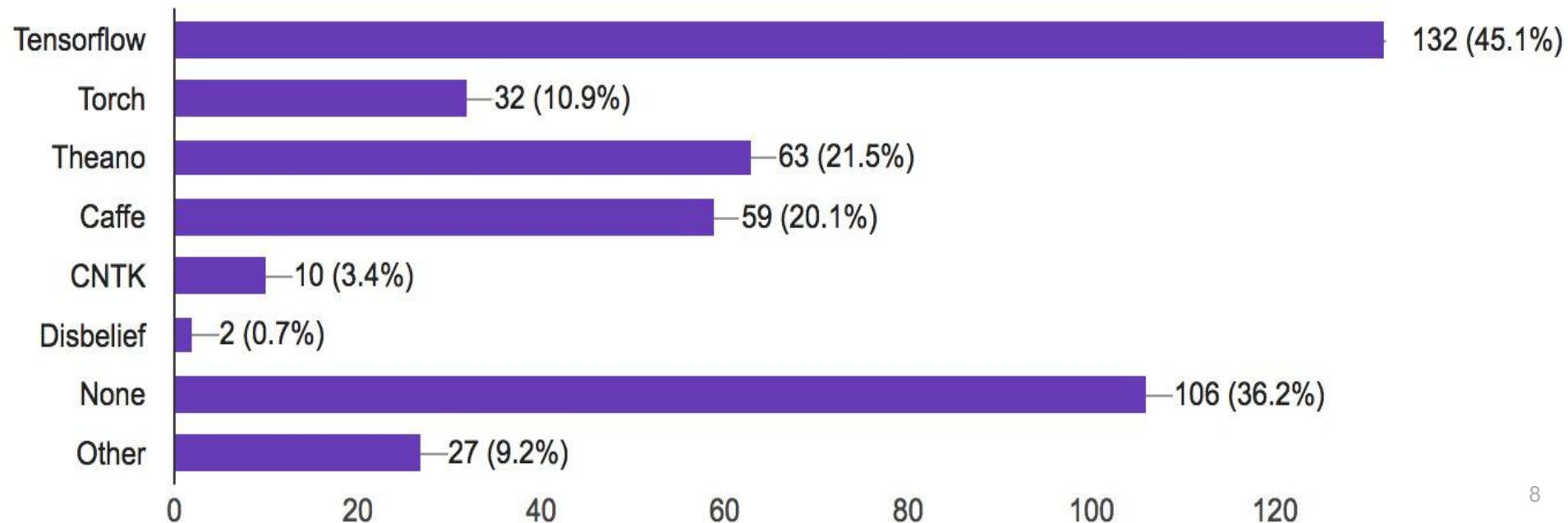
Launched Nov 2015

Interest over time ?

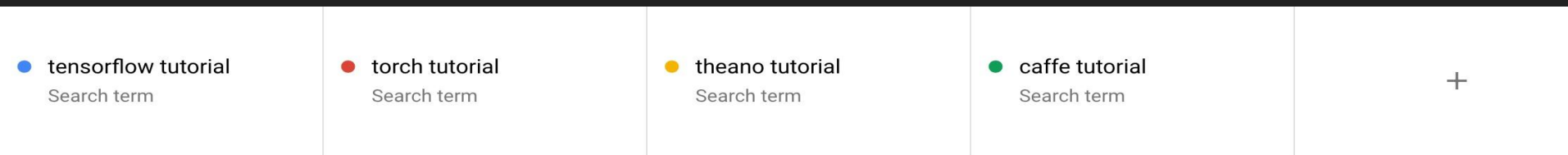


TF is not the only deep learning library

From students signed up for this class

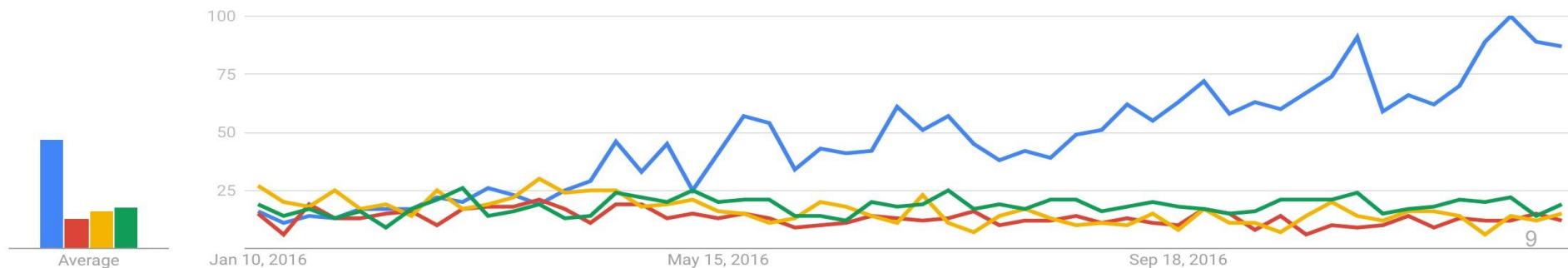


Why TensorFlow?



Worldwide ▼ Past 12 months ▼ All categories ▼ Web Search ▼

Interest over time ?



Why TensorFlow?

- Python API
- Portability: deploy computation to one or more CPUs or GPUs in a desktop, server, or mobile device with a single API
- Flexibility: from Raspberry Pi, Android, Windows, iOS, Linux to server farms
- Visualization (TensorBoard is da bomb)
- Checkpoints (for managing experiments)
- Auto-differentiation *autodiff* (no more taking derivatives by hand. Yay)
- Large community (> 10,000 commits and > 3000 TF-related repos in 1 year)
- Awesome projects already using TensorFlow

Companies using Tensorflow

- Google
- OpenAI
- DeepMind
- Snapchat
- Uber
- Airbus
- eBay
- Dropbox
- A bunch of startups



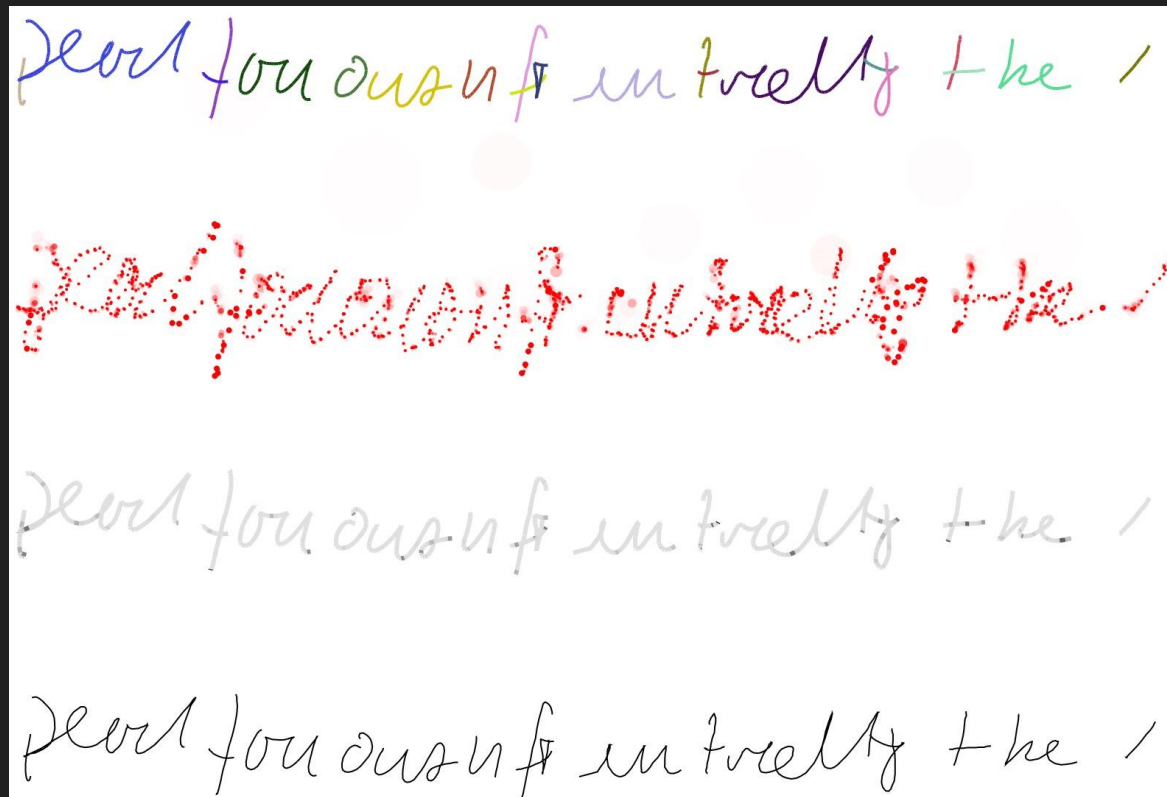
Some cool projects using TensorFlow

Neural Style Transfer



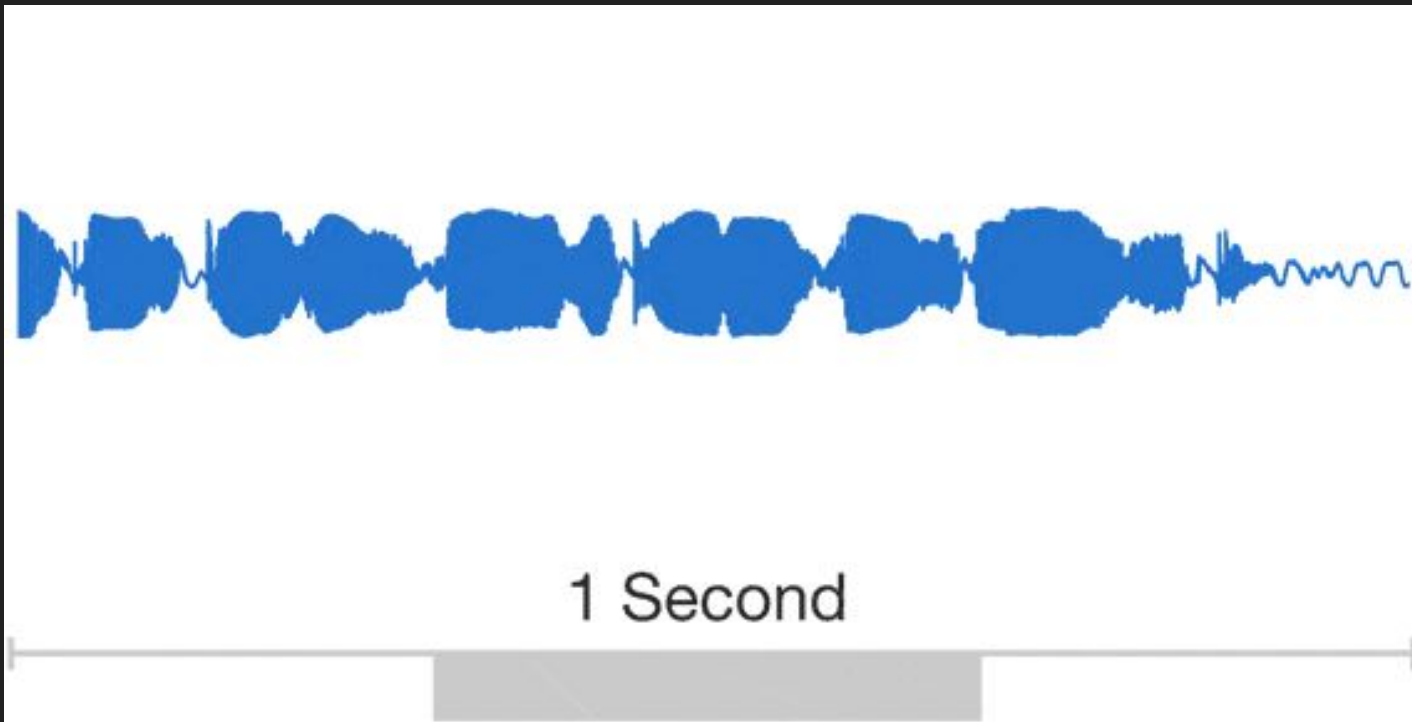
"Image Style Transfer Using Convolutional Neural Networks" by Leon A. Gatys et al. (2016)

Generative Handwriting



WaveNet: Text to Speech

It takes several hours to synthesize 1 second!





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
- Learn how to build and structure models best suited for a deep learning project.

Introduction

Logistics

- Piazza
- Emails:
([cs20si-win1617-staff/cs20si-win1617-students/cs20si-win1617-guests](#))
- Assignments (3)
- Participation is a big chunk of grades
- A lot of you are ahead of me in your academic career so I probably need more of your help than you do mine . Feedback is greatly appreciated!

Books

- TensorFlow for Machine Intelligence (TFFMI)
- Hands-On Machine Learning with Scikit-Learn and TensorFlow. Chapter 9: Up and running with TensorFlow
- Fundamentals of Deep Learning. Chapter 3: Implementing Neural Networks in TensorFlow (FODL)

TensorFlow is being constantly updated so books might become outdated fast

Check [tensorflow.org](https://www.tensorflow.org) directly



Getting Started

```
import tensorflow as tf
```

Simplified TensorFlow?

1. TF Learn (`tf.contrib.learn`): simplified interface that helps users transition from the the world of one-liner such as scikit-learn
2. TF Slim (`tf.contrib.slim`): lightweight library for defining, training and evaluating complex models in TensorFlow.
3. High level API: Keras, TFLearn, Pretty Tensor

But we don't need baby TensorFlow ...

Off-the-shelf models are not the main purpose of TensorFlow.

TensorFlow provides an extensive suite of functions and classes that allow users to define models from scratch.

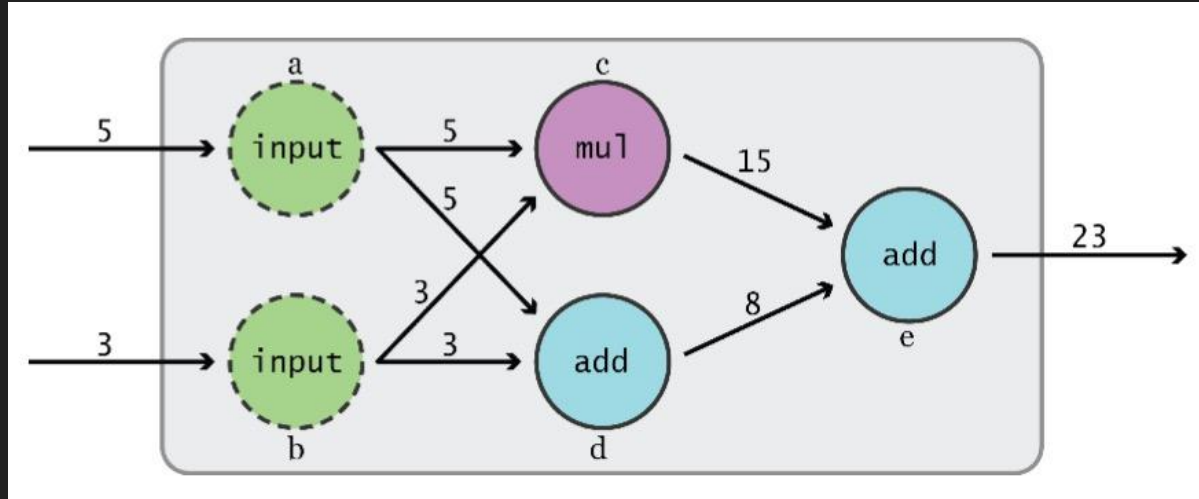
And this is what we are going to learn.



Graphs and Sessions

Data Flow Graphs

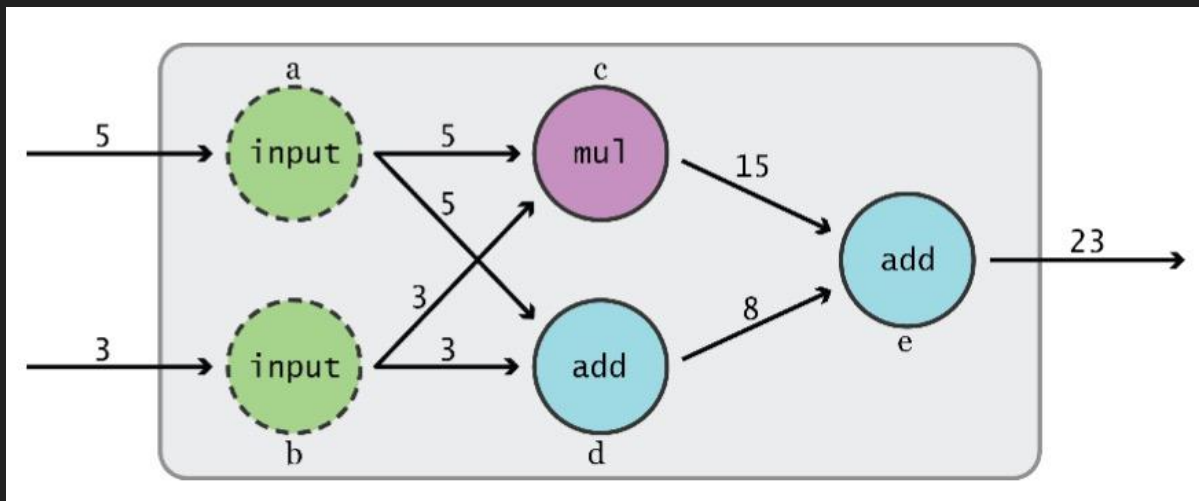
TensorFlow separates definition of computations from their execution



Data Flow Graphs

Phase 1: assemble a graph

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



What's a tensor?

What's a tensor?

An n-dimensional matrix

0-d tensor: scalar (number)

1-d tensor: vector

2-d tensor: matrix

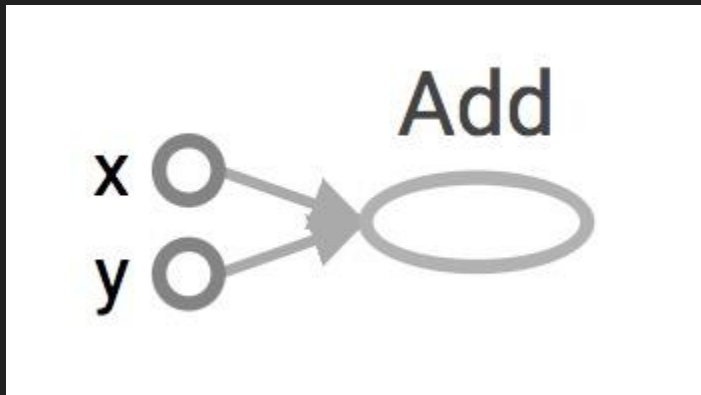
and so on

Data Flow Graphs

Visualized by TensorBoard

```
import tensorflow as tf
```

```
a = tf.add(2, 3)
```



Why x, y?

TF automatically names the nodes when you don't explicitly name them. More about this next lecture! For

now:

x = 3

y = 5

Data Flow Graphs

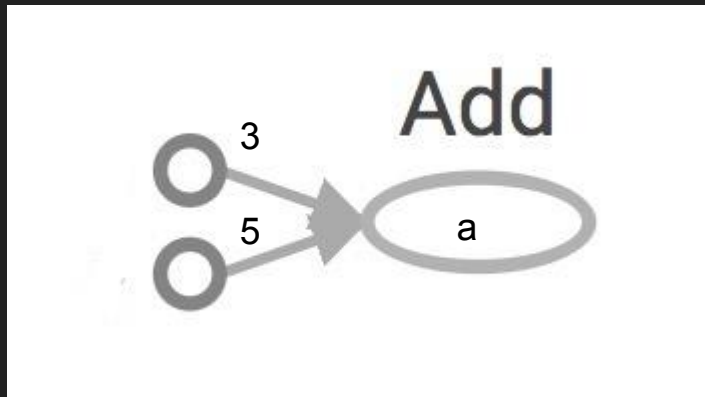
Interpreted?

```
import tensorflow as tf
```

```
a = tf.add(3, 5)
```

Nodes: operators, variables, and constants

Edges: tensors



Data Flow Graphs

Interpreted?

```
import tensorflow as tf
```

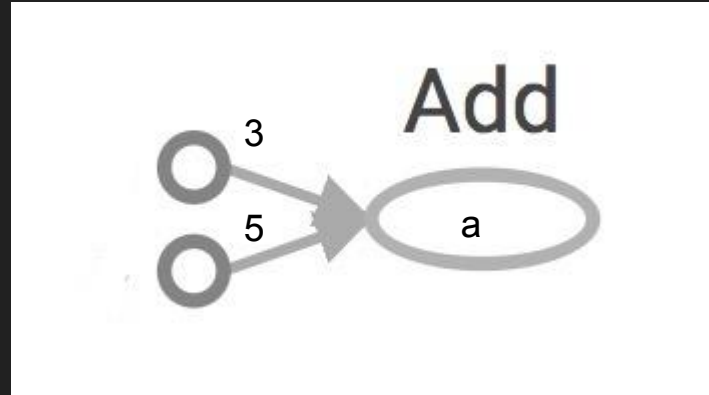
```
a = tf.add(3, 5)
```

Nodes: operators, variables, and constants

Edges: tensors

Tensors are data.

Data Flow -> Tensor Flow (I know, mind=blown)



Data Flow Graphs

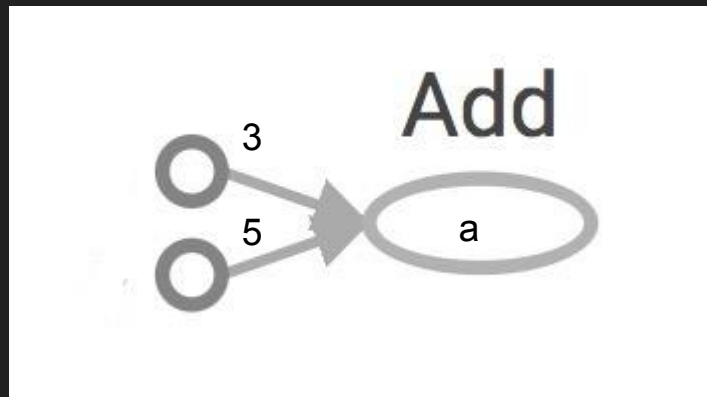
```
import tensorflow as tf
```

```
a = tf.add(3, 5)
```

```
print a
```

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

(Not 5)



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
```

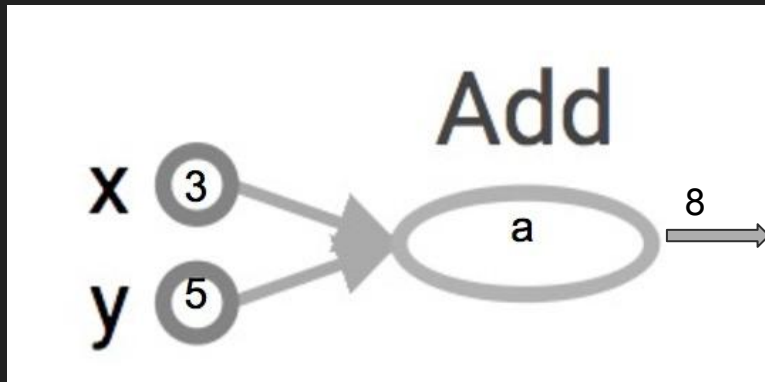
```
a = tf.add(3, 5)
```

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

```
print sess.run(a)
```

```
sess.close()
```

```
>> 8
```



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

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

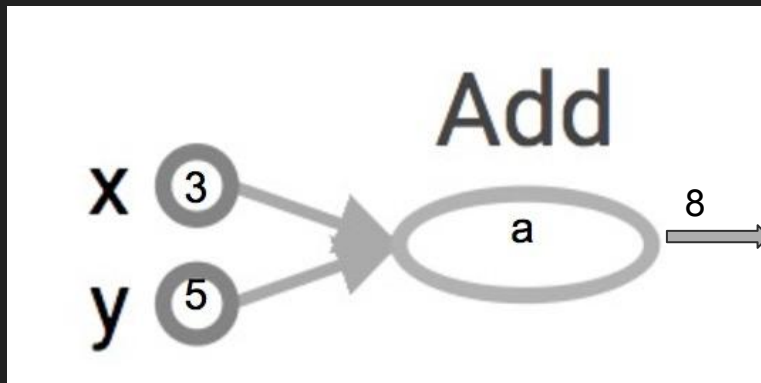
```
import tensorflow as tf

a = tf.add(3, 5)

# with clause takes care
# of sess.close()

with tf.Session() as sess:

    print sess.run(a)
```



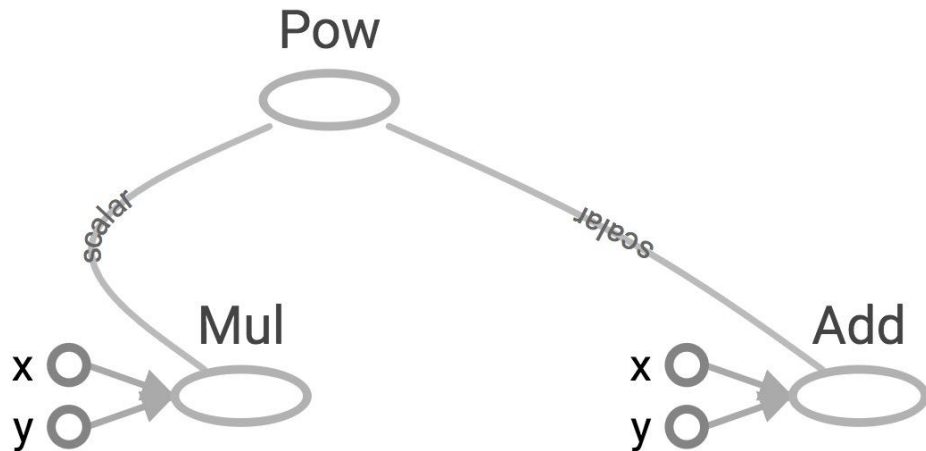
tf.Session()

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

More graphs

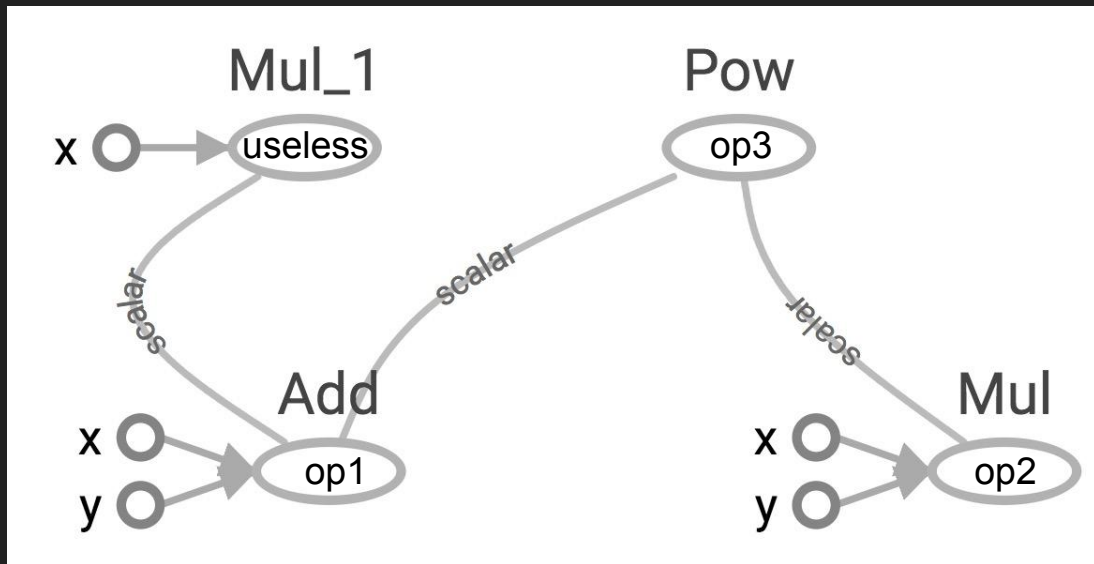
Visualized by TensorBoard

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



More (sub)graphs

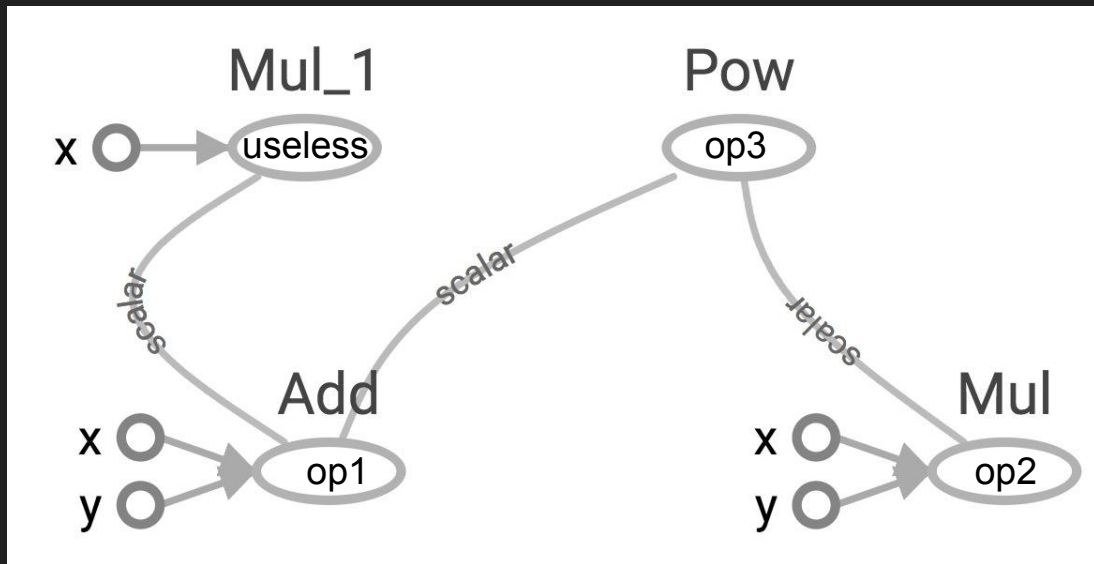
```
x = 2  
y = 3  
op1 = tf.add(x, y)  
op2 = tf.mul(x, y)  
useless = tf.mul(x, op1)  
op3 = tf.pow(op2, op1)  
  
with tf.Session() as sess:  
    op3 = sess.run(op3)
```



Because we only want the value of `z` and `z` doesn't depend on `useless`, session won't compute values of `useless`
→ save computation

More (sub)graphs

```
x = 2  
y = 3  
  
op1 = tf.add(x, y)  
op2 = tf.mul(x, y)  
useless = tf.mul(x, op1)  
op3 = tf.pow(op2, op1)  
  
with tf.Session() as sess:  
    op3, not_useless = sess.run([op3, useless])
```

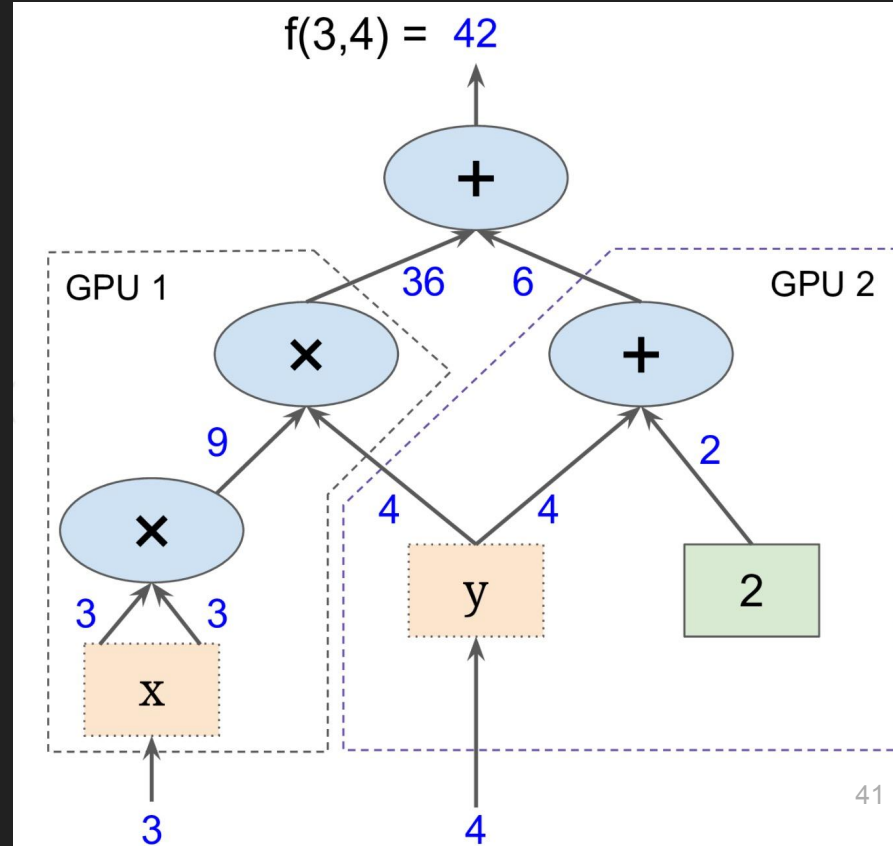


```
tf.Session.run(fetches, feed_dict=None,  
options=None, run_metadata=None)
```

pass all variables whose values you want to a list in fetches

More (sub)graphs

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



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.matmul(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)
```

Don't worry about this yet.
More on this in week 8!

**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

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

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

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

```
    a = 3
```

```
    b = 5
```

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

Same as previous

```
sess = tf.Session(graph=g) # session is run on the graph g
```

```
# run session
```

```
sess.close()
```

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)
```

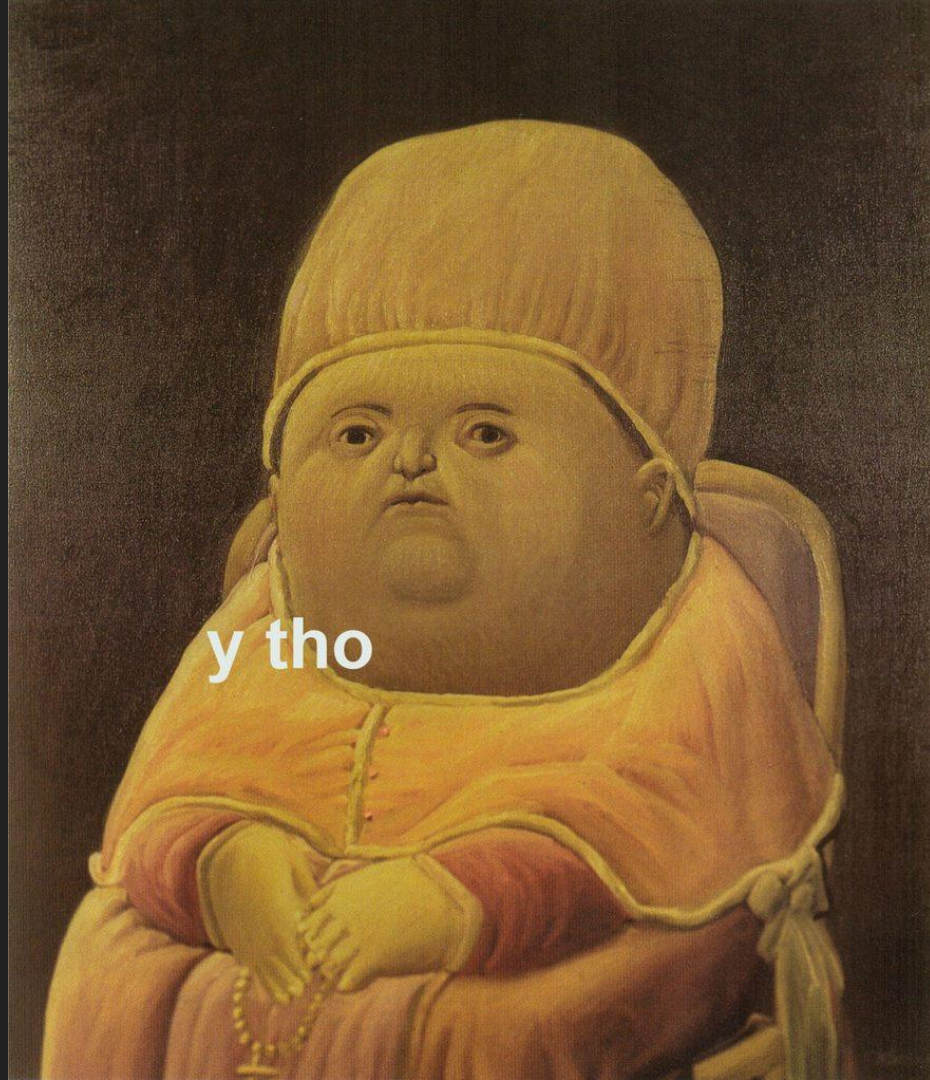
```
# add ops to the user created graph
```

```
with g2.as_default():
```

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

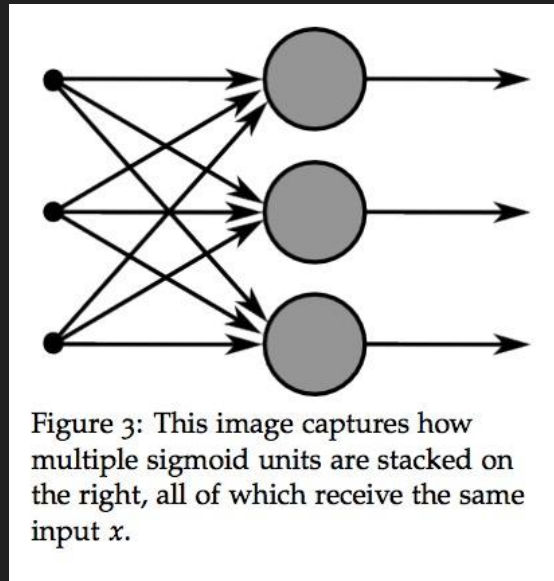
Better

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)
2. Break computation into small, differential pieces to facilitates auto-differentiation
3. Facilitate distributed computation, spread the work across multiple CPUs, GPUs, or devices
4. Many common machine learning models are commonly taught and visualized as directed graphs already



A neural net graph by Richard Socher (CS224D)

Next class

Basic operations

Constants and variables

Feeding inputs

Fun with TensorBoard

Feedback: huyenn@stanford.edu

Thanks!