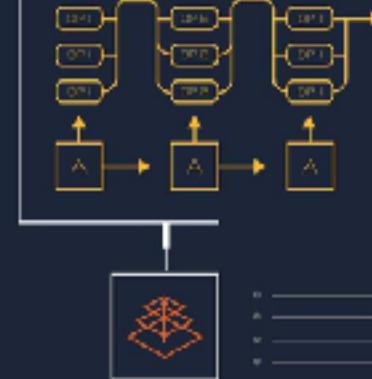
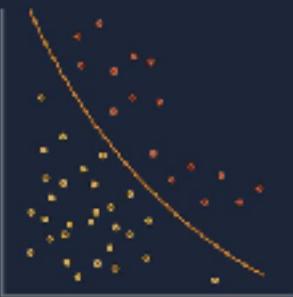




# TensorFlow



# TensorFlow

## Intro to TensorFlow



Sam Witteveen, GDE for Machine Learning  
[@sam\\_witteveen](https://twitter.com/sam_witteveen)



# About me

- Google Developer Expert for Machine Learning & Deep Learning
- Red Dragon AI The logo features a stylized red dragon head and body above the text "RED DRAGON AI".
- Use Deep Learning for Conversational Computing and Natural Language Understanding



# Today's Talk

- What is TensorFlow
- How its being used around the world
- How you can get started



# What is TensorFlow?



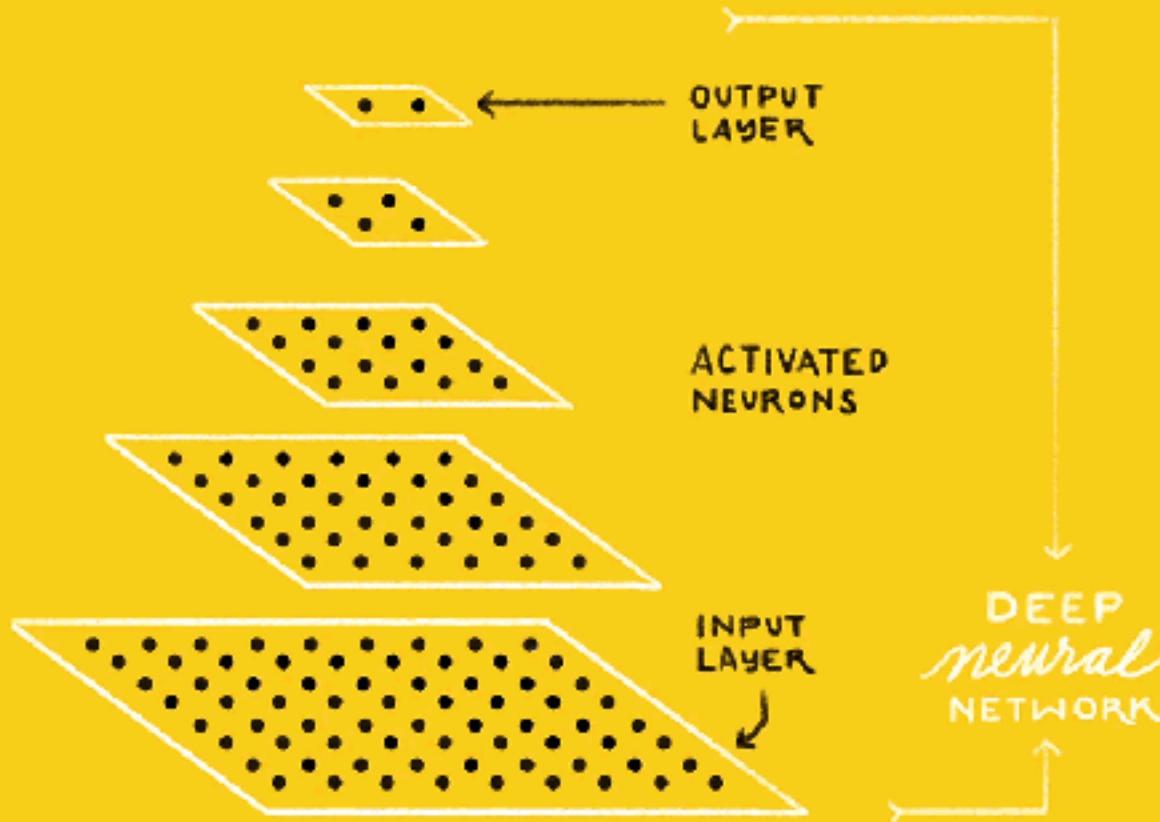
# TensorFlow: ML for Everyone

- An open-source machine learning platform for everyone
- **Fast, flexible, and production-ready**
- Scales from research to production

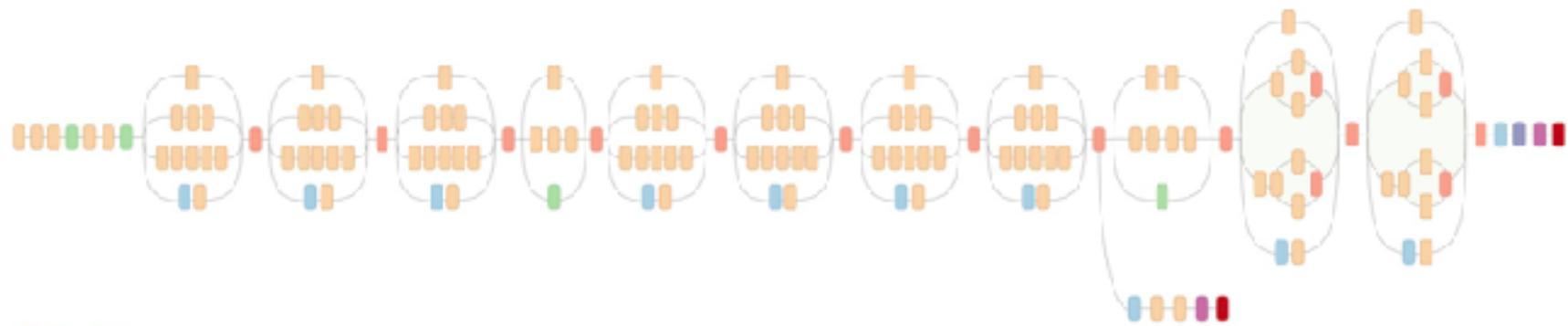


# CAT DOG

IS THIS A  
**CAT or DOG?**



## Inception V3 - Deep Learning Convolutional Neural Network



- Convolution
- AvgPool
- MaxPool
- Concat
- Dropout
- Fully connected
- Softmax

## Dataflow based Computational graph

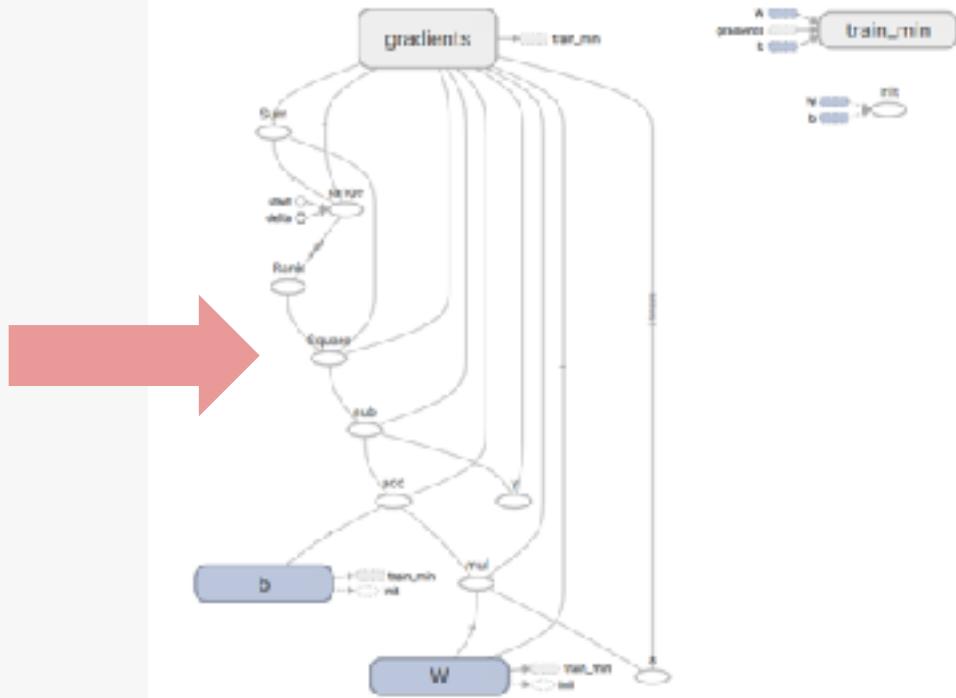
### Python Program

```
import numpy as np
import tensorflow as tf

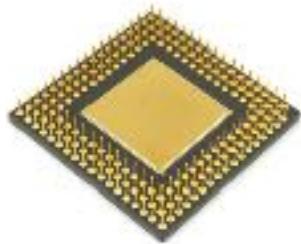
# Model parameters:
m = tf.Variable([.3], tf.float32)
b = tf.Variable([- .3], tf.float32)
# Model input and output
x = tf.placeholder(tf.float32)
y_instar_model = m * x + b
y = tf.placeholder(tf.float32)
# loss
loss = tf.reduce_sum(tf.square(y_instar_model - y)) # sum of the squares
# optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
# training data
x_train = [1, 2, 3, 4]
y_train = [0, -1, -2, -3]
# training loop
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
    sess.run(train, {x: x_train, y: y_train})

# evaluate training accuracy
curr_m, curr_b, curr_loss = sess.run([m, b, loss], {x: x_train, y: y_train})
print("W: %s b: %s loss: %s" % (curr_m, curr_b, curr_loss))
```

### TensorFlow Graph



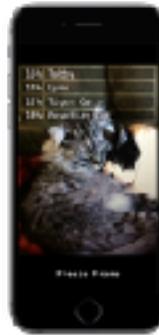
# TensorFlow Supports Many Platforms...



CPU



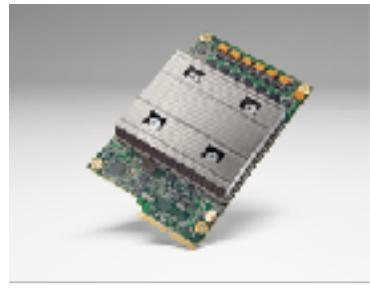
GPU



iOS



Android



TPU



Cloud TPU



Raspberry  
Pi



TensorFlow support many languages



C++

Java



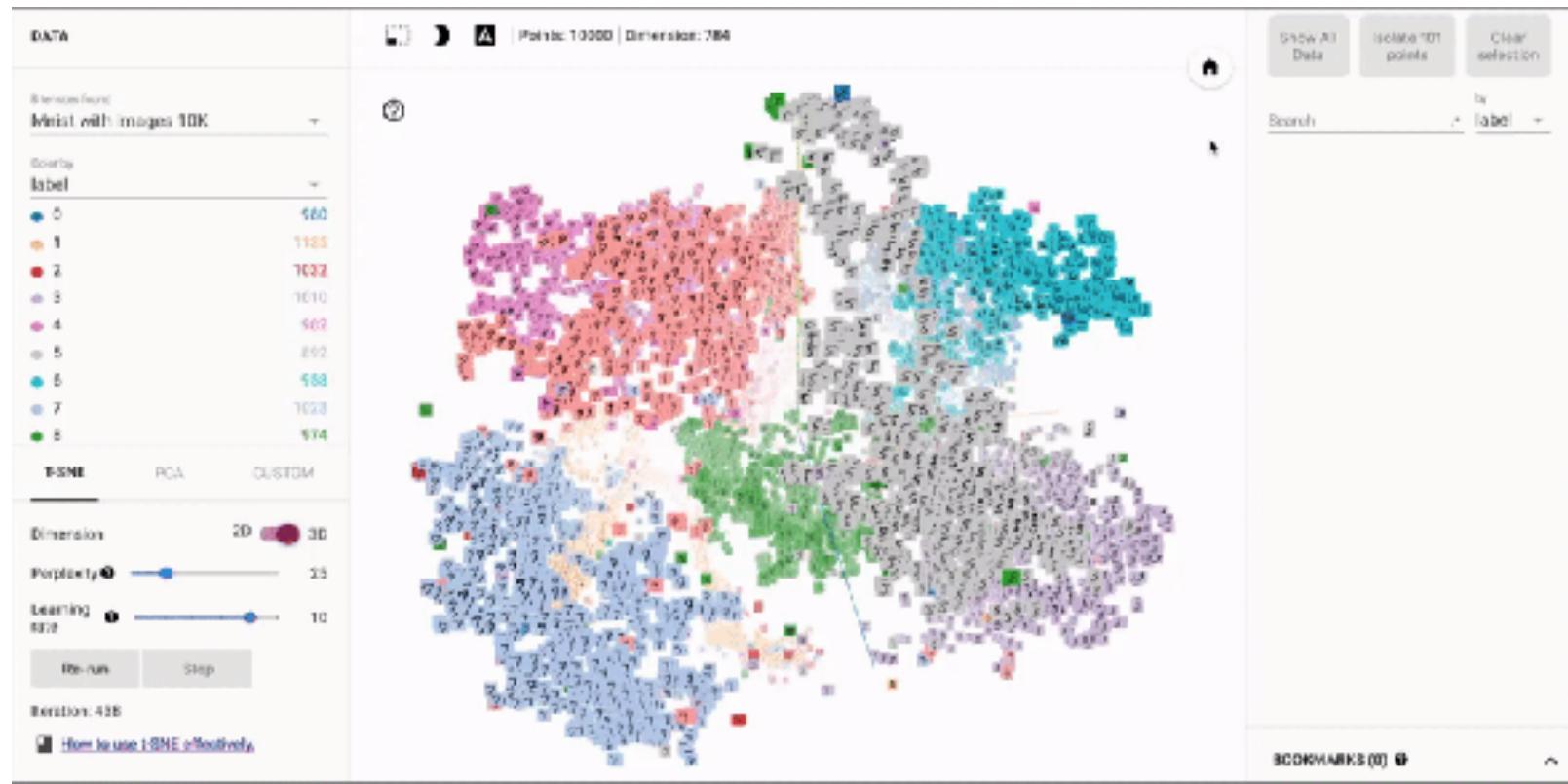
C#

julia

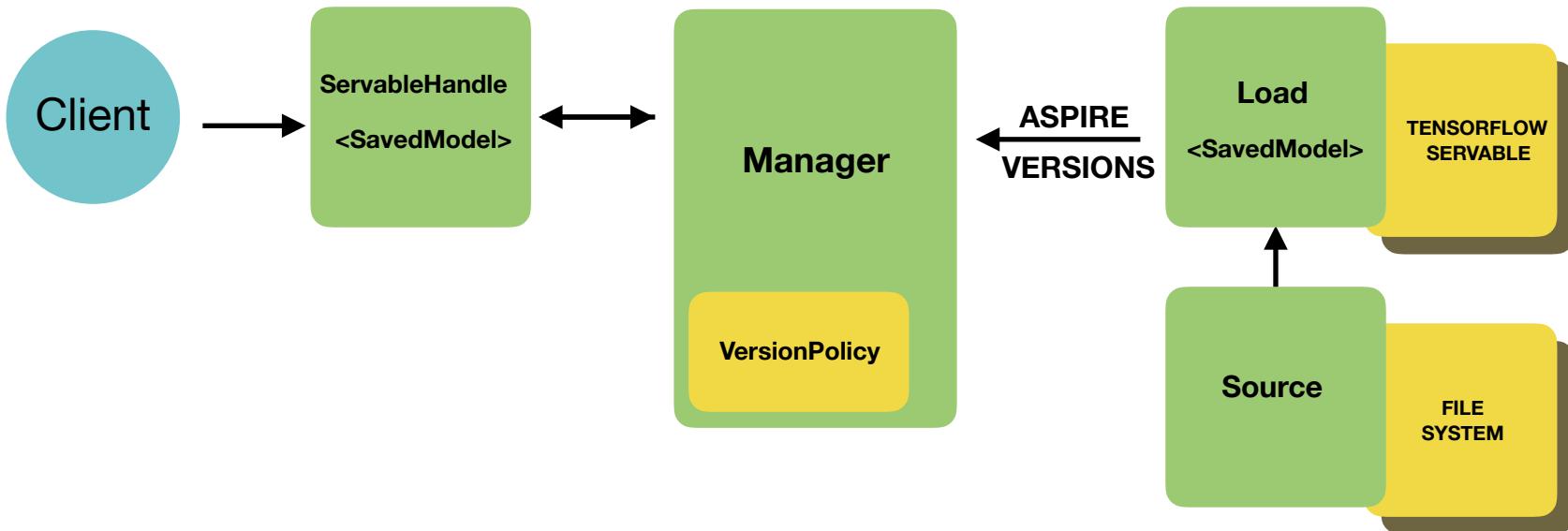
Haskell



# TensorFlow provides great tools like TensorBoard



# TensorFlow Serving



## Legend:

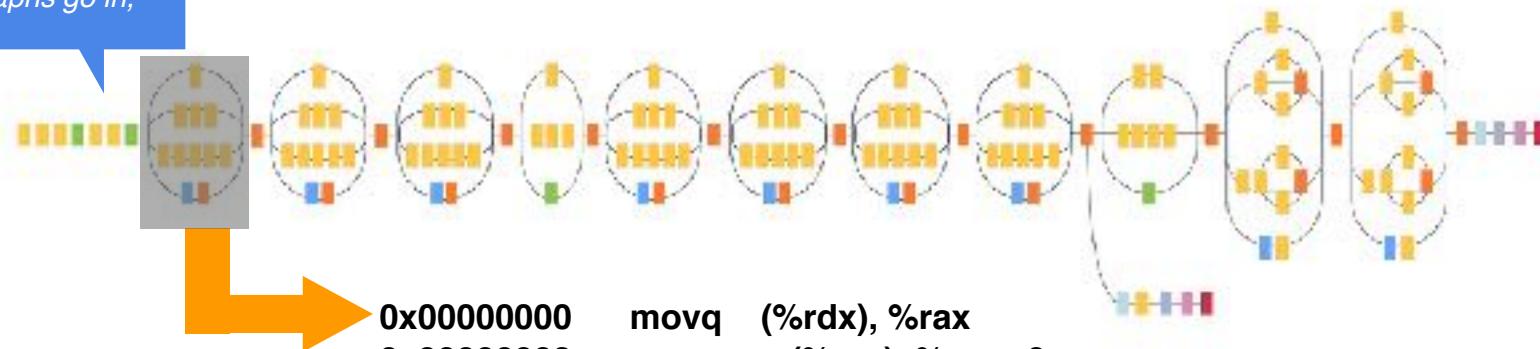
**Blue:** Caller-specific code

**Green:** TensorFlow Serving core classes / APIs

**Yellow:** Pluggable implementations

# XLA: An Experimental TensorFlow Compiler

TF graphs go in,



Optimized & specialized  
assembly comes out.

The screenshot shows the TensorFlow website homepage. At the top, there's a navigation bar with links for "Get Started", "Videos", "API & Docs", "Mobile", "Resources", and "Versions". Below the navigation is a search bar with a magnifying glass icon and the word "Search". A date "2016-06-20" is displayed in the top right corner. The main header features the text "TensorFlow" with a small "v" icon, followed by "An open-source software library for Machine Intelligence". A large orange button labeled "GET STARTED" is visible. The background of the page has a yellow-to-orange gradient with a faint, abstract geometric pattern. On the left side, there's a section titled "About TensorFlow" with a detailed description of what TensorFlow is and its capabilities. To the right of the description is the TensorFlow logo, which consists of a stylized orange and grey "T" shape above the word "TensorFlow" in a bold, sans-serif font. Below the logo, there are three cards with links: "Celebrating TensorFlow's First Year", "A Neural Network for Machine Translation at Production Scale", and "Improving Inception and Image Classification in TensorFlow".

An open-source software library for Machine Intelligence

GET STARTED

## About TensorFlow

TensorFlow™ is an open-source software library for numerical computation using data flow graphs, based on the graph computation framework developed while the authors were developing the multi-dimensional distributed tensor D-DNN. TensorFlow represents the multi-dimensional distributed data structure called Tensor and communication between them. The flexible architecture allows you to deploy your code to one or more CPU or GPU in a mobile device, in a desktop computer with a single API. TensorFlow was originally developed in research and industrial domains at the Google Brain Team before Google Machine Intelligence research organization for the purpose of conducting machine learning and deep neural network research, but the system is general enough to be applicable in a wide variety of other domains as well.

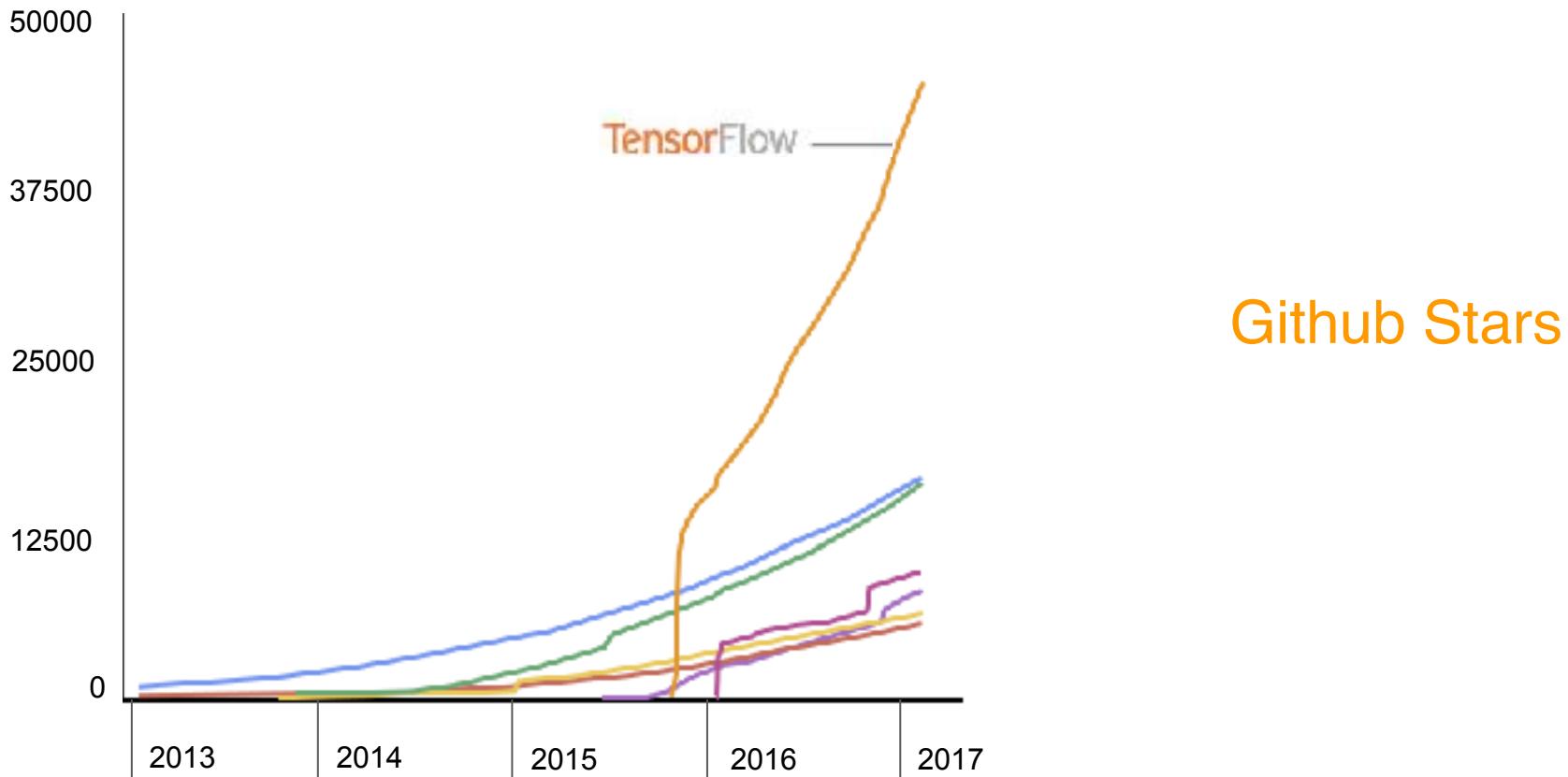
TensorFlow

Celebrating TensorFlow's First Year

A Neural Network for Machine Translation at Production Scale

Improving Inception and Image Classification in TensorFlow

#1  
repository  
in “machine learning”  
category on GitHub



Github Stars

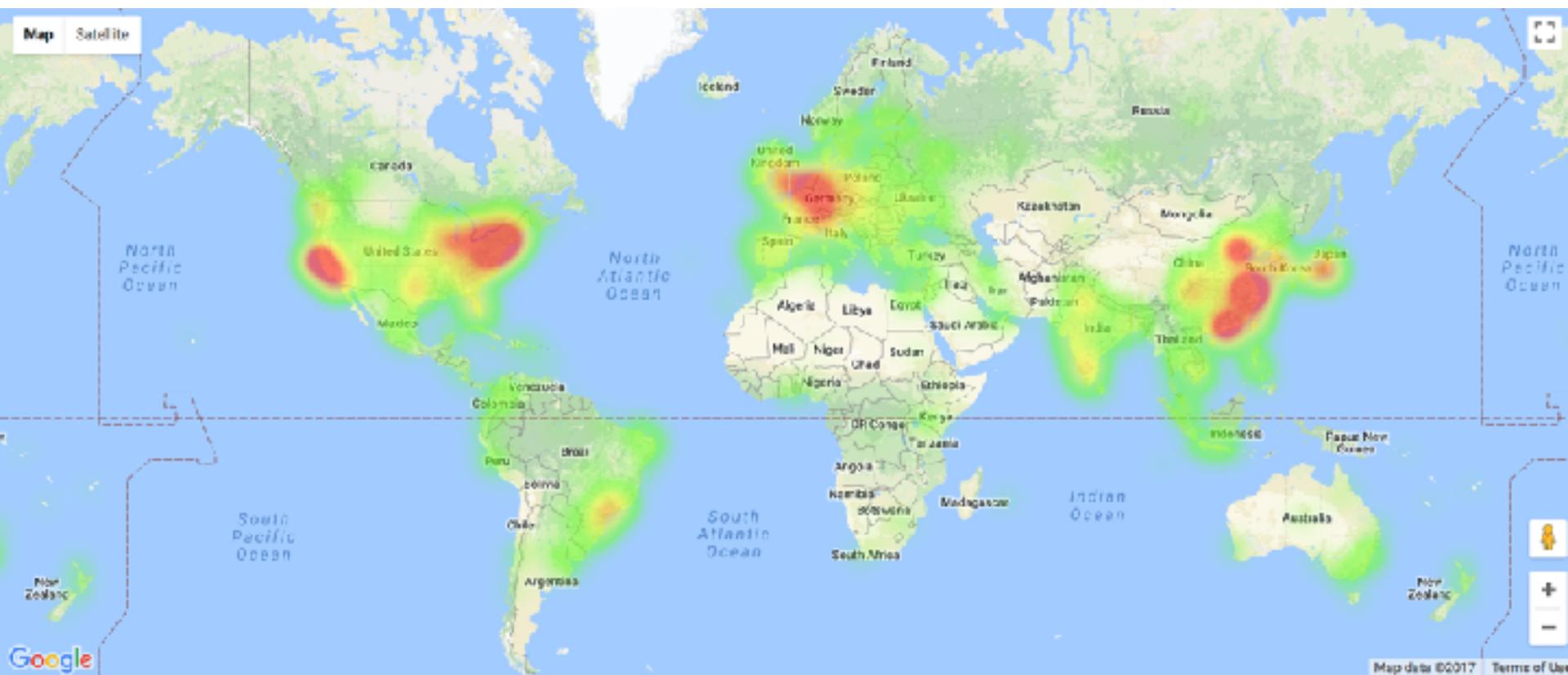
TensorFlow

# TensorFlow: A Vibrant Open-Source Community

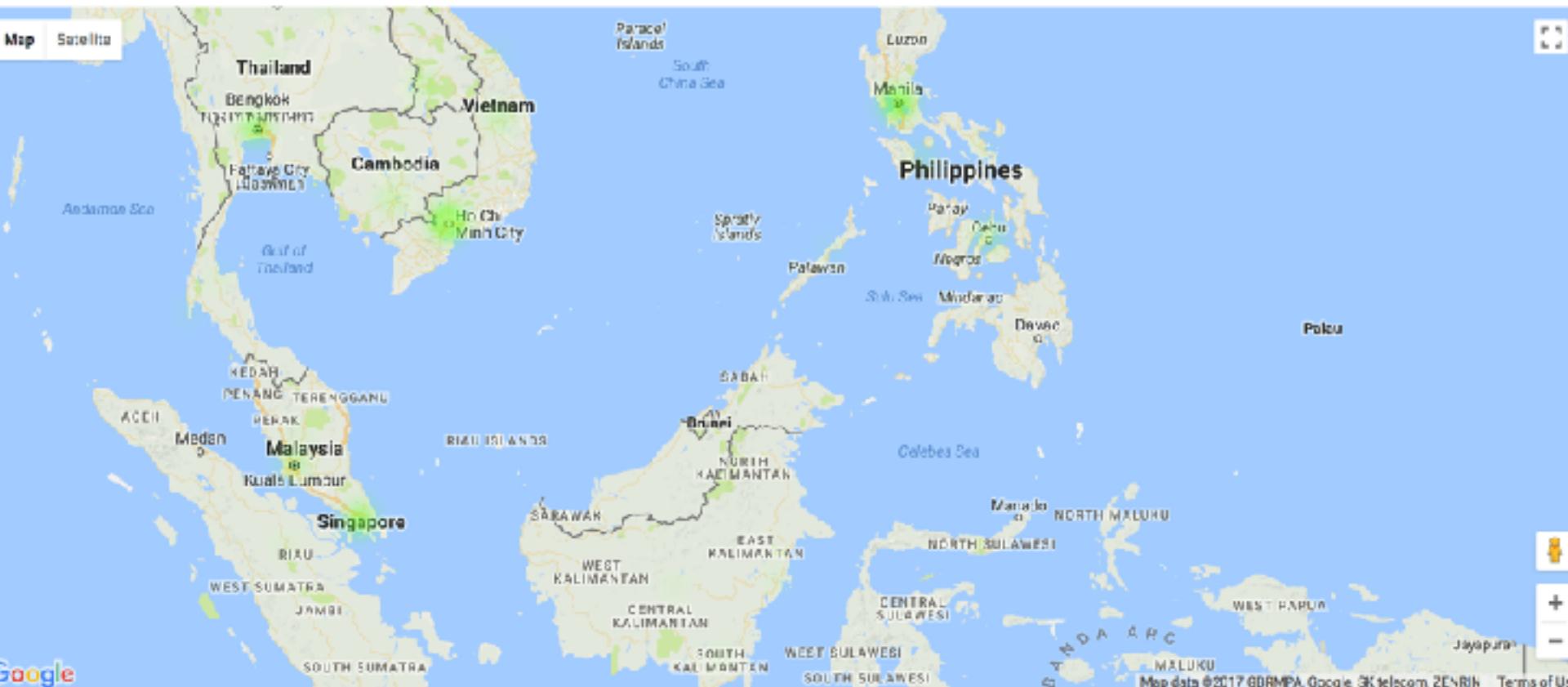
- **Rapid development, many outside contributors**
  - 500+ non-Google contributors
  - 20,000+ commits
  - Many community created tutorials, models, translations, and projects
    - ~5,500 GitHub repositories with ‘TensorFlow’ in the title
- **Direct engagement between community and TensorFlow team**
  - 5000+ Stack Overflow questions answered
  - 5000+ GitHub issues filed and answered; 160+ new issues / week
- **Use in ML classes is growing: Toronto, Berkeley, Stanford, ...**



# Use of TensorFlow Worldwide



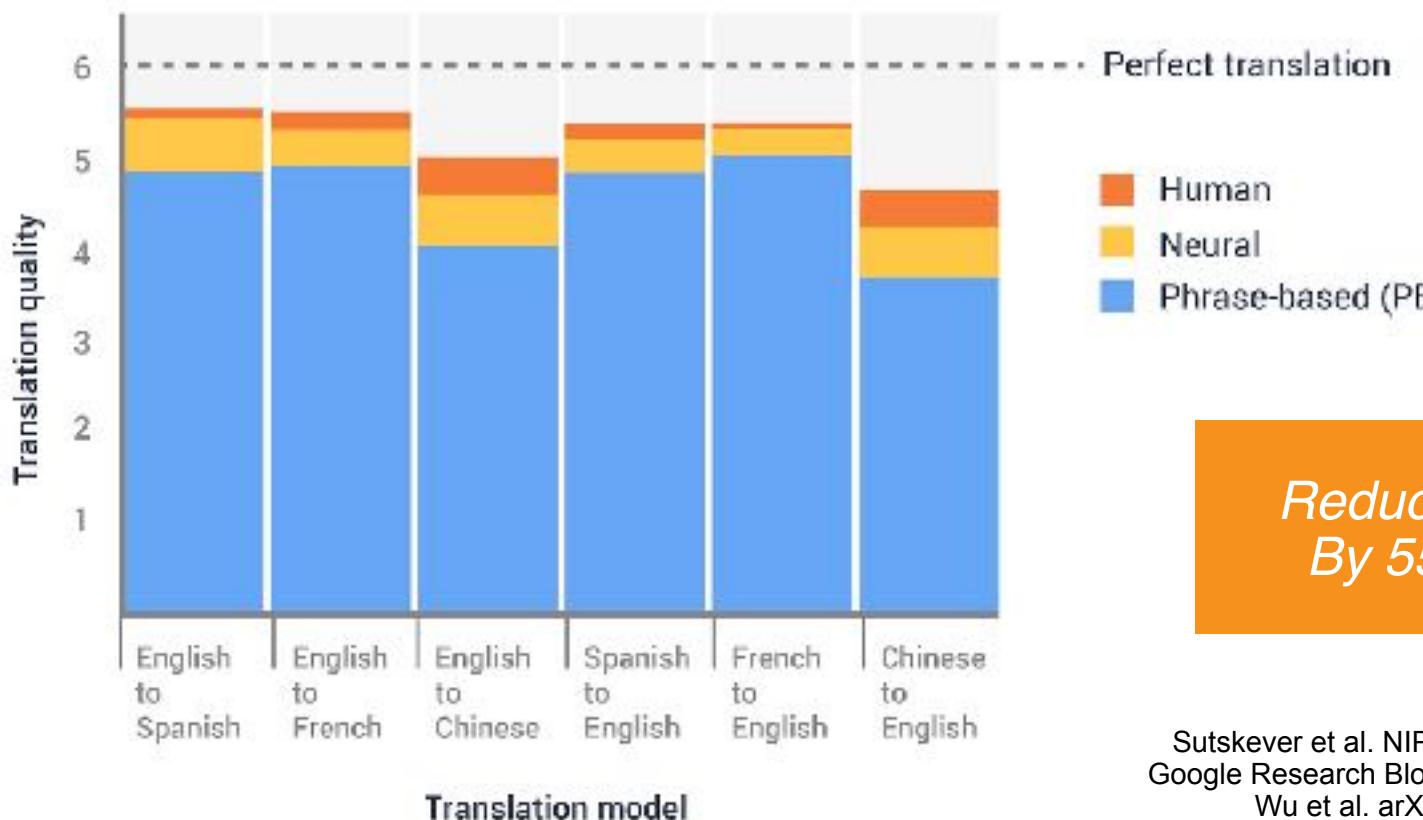
# Use in South East Asia



# What is TensorFlow being used for?



# Neural Machine Translation



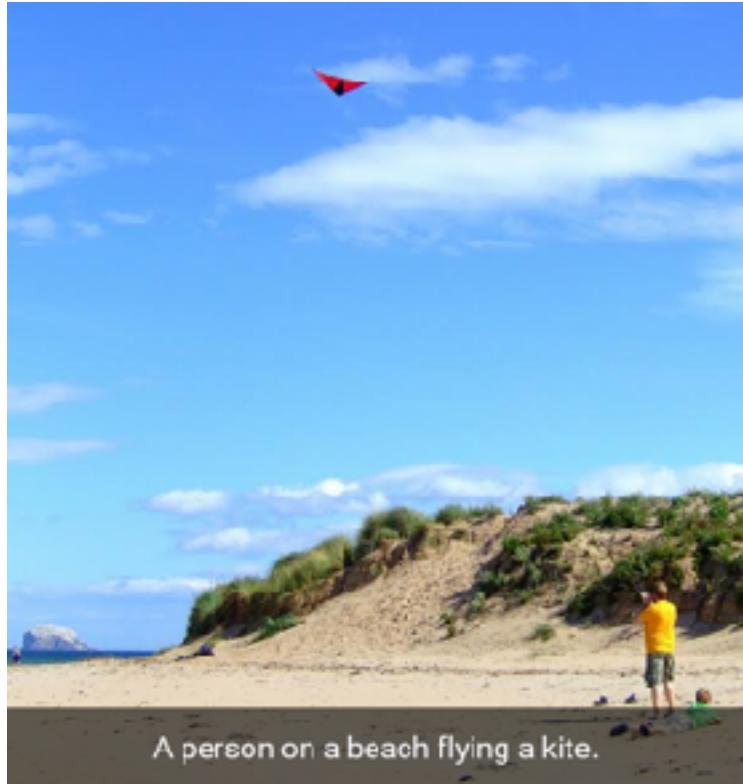
*Reduces Errors  
By 55%-85%*

Sutskever et al. NIPS, Dec 2014  
Google Research Blog, Sept 2016  
Wu et al. arXiv, Sept 2016

# Show and Tell



A blue and yellow train traveling down train tracks.



A person on a beach flying a kite.

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



# Mobile Deployment



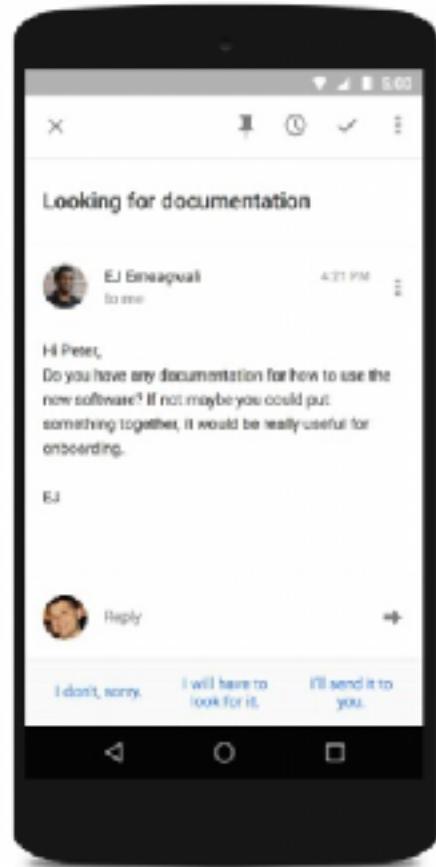


# Smart Reply

*April 1, 2009: April Fool's Day joke*

*Nov 5, 2015: Launched Real Product*

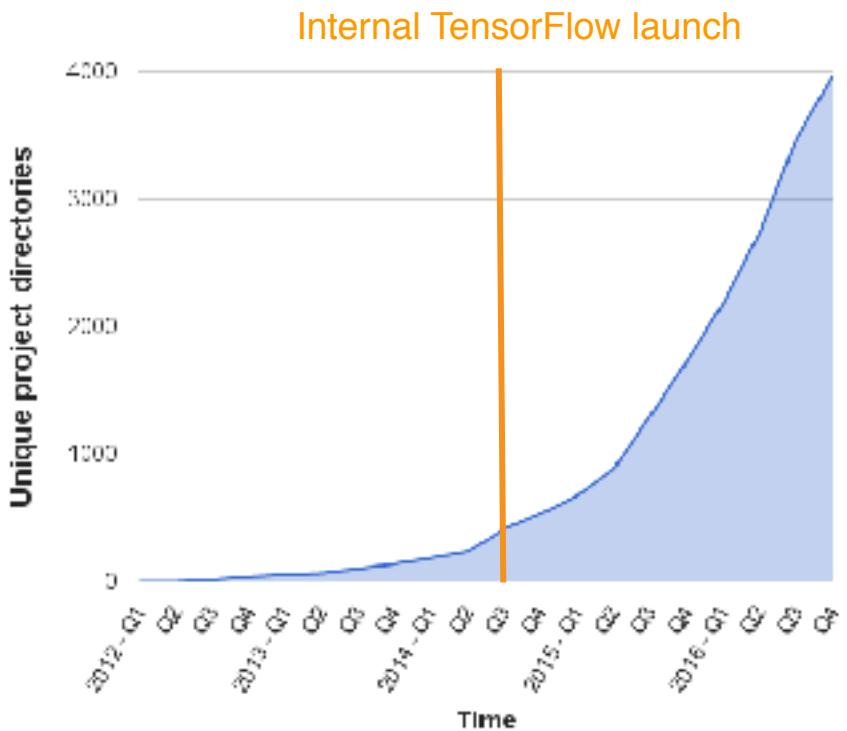
*Feb 1, 2016: >10% of mobile Inbox replies*



Research at Google



# # of Google directories containing model description files



## Production use in many areas:

- Search
- Gmail
- Translate
- Maps
- Android
- Photos
- Speech
- YouTube
- Play
- ... many others ...

## Research use for:

100s of projects and papers

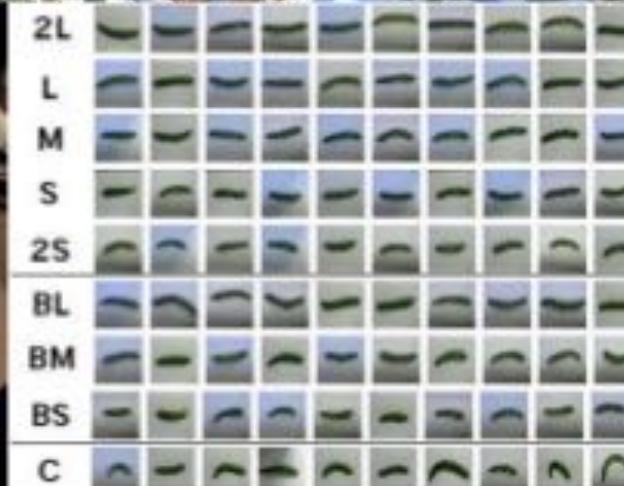




TensorFlow is for **everyone**



# TensorFlow powered Cucumber Sorter





<https://cloud.google.com/blog/big-data/2016/08/how-a-japanese-cucumber-farmer-is-using-deep-learning-and-tensorflow>

# Getting Started

Key parts of the TensorFlow API



# TensorFlow pre v1.0



Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS



Layers

← Build Models

Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS

XLA

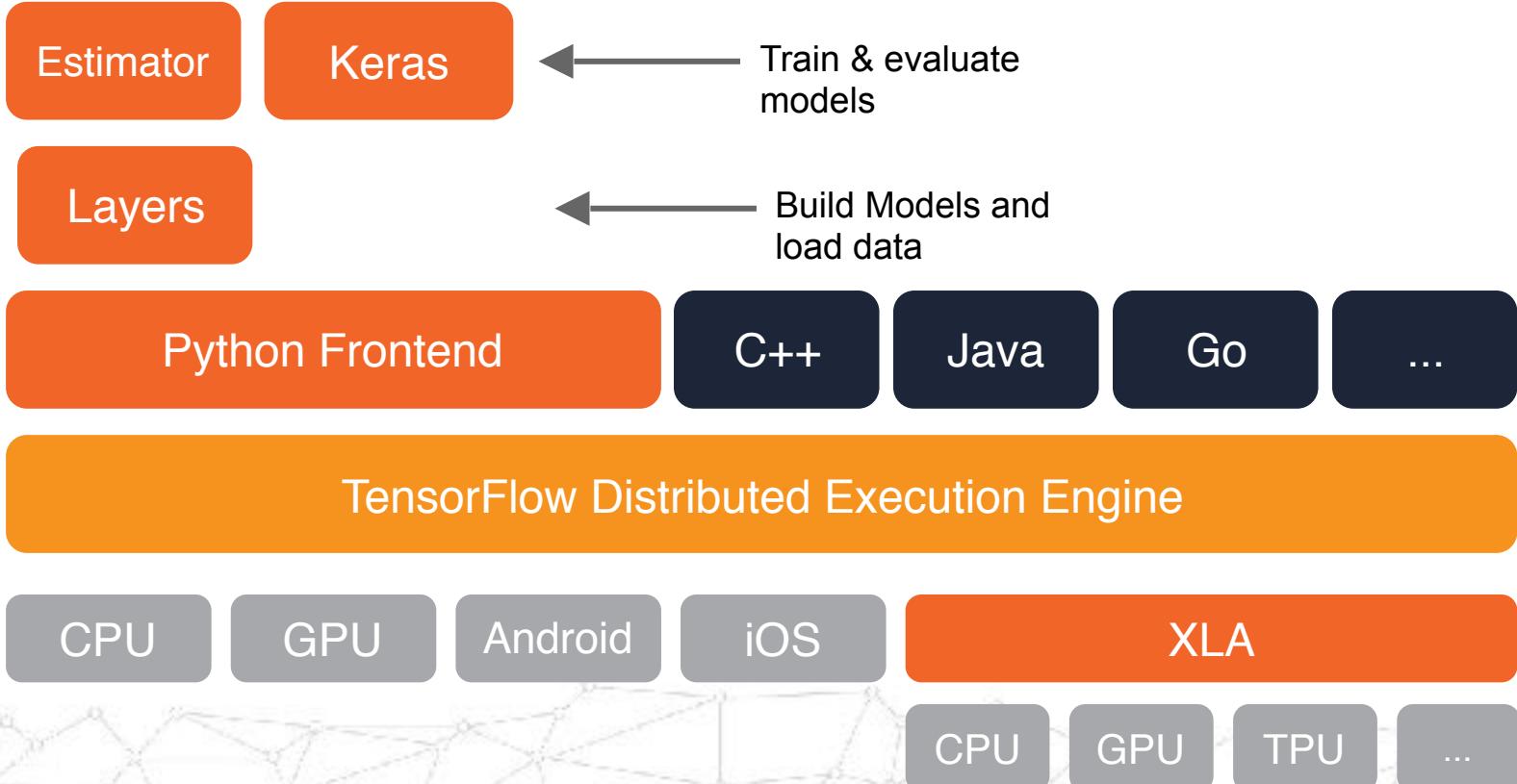
CPU

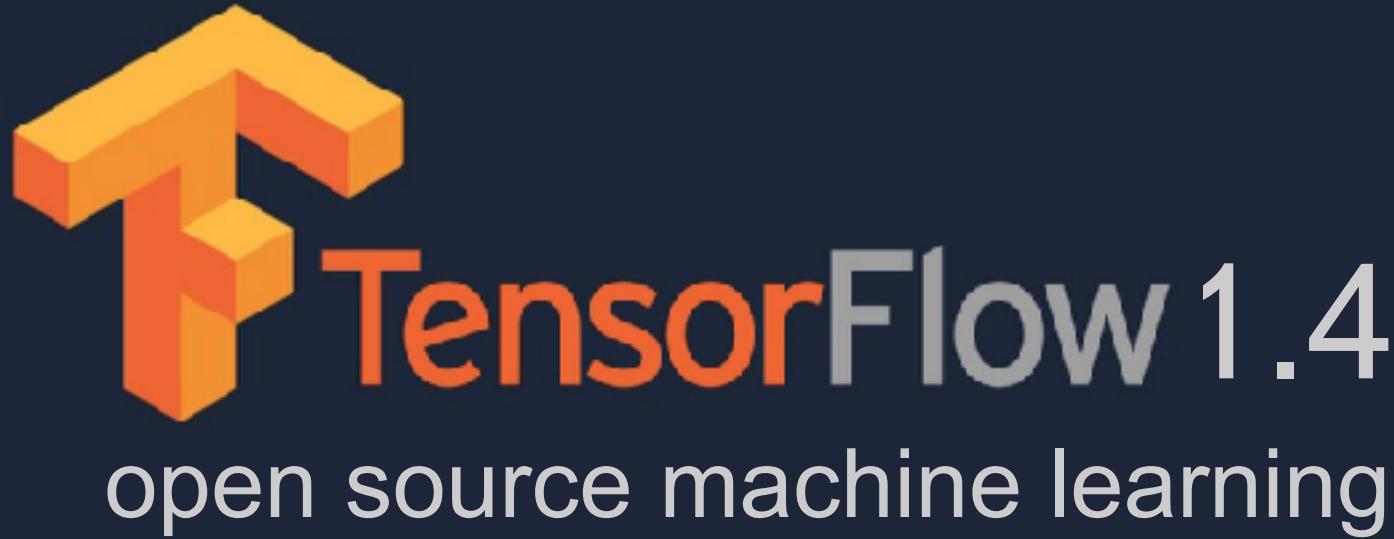
GPU

TPU

...







Canned Estimators

Models in a box

Estimator

Keras

Train & evaluate  
models

Layers

Datasets

Build Models and  
load data

Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS

XLA

CPU

GPU

TPU

...



# TensorFlow v1.4



Canned Estimators

Models in a box

Estimator

Keras

Train & evaluate  
models

Layers

Datasets

Build Models and  
load data

Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS

XLA

CPU

GPU

TPU

...



## Flexible: High level APIs

### Low-Level Python API

```
import numpy as np
import tensorflow as tf

W = tf.Variable([.3], tf.float32)
b = tf.Variable([-3], tf.float32)
x = tf.placeholder(tf.float32)
linear_model = W * x + b
y = tf.placeholder(tf.float32)
loss = tf.reduce_sum(tf.square(linear_model - y))
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)
x_train = [1,2,3,4]
y_train = [0,-1,-2,-3]
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
    sess.run(train, {x:x_train, y:y_train})
curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x:x_train, y:y_train})
```

### High-Level Python API

```
import tensorflow as tf
import numpy as np

features = [tf.contrib.layers.real_valued_column("x", dimension=1)]
estimator = tf.contrib.learn.LinearRegressor(feature_columns=features)
x = np.array([1., 2., 3., 4.])
y = np.array([0., -1., -2., -3.])
input_fn = tf.contrib.learn.io.numpy_input_fn({"x":x}, y, batch_size=4,
                                              num_epochs=1000)
estimator.fit(input_fn=input_fn, steps=1000)
estimator.evaluate(input_fn=input_fn)
```

## Flexible: High level APIs

### Low-Level Python API

```
import numpy as np
import tensorflow as tf

W = ...
b = ...
x = ...
line = ...
y = ...
loss = ...
op = ...
train = ...

x_train = [1,2,3,4]
y_train = [0,-1,-2,-3]
init = tf.global_variables_initializer()
sess = tf.Session()
sess.run(init) # reset values to wrong
for i in range(1000):
    sess.run(train, {x:x_train, y:y_train})
curr_W, curr_b, curr_loss = sess.run([W, b, loss], {x:x_train, y:y_train})
```

15 lines

### High-Level Python API

```
import tensorflow as tf
import numpy as np

features = [tf.contrib.learn.infer_real_valued_columns_from_input]
estimator = tf.contrib.learn.LinearRegressor(feature_columns=features)
x = np.array([1., 2., 3., 4.])
y = np.array([0., -1., -2., -3.])
input_fn = tf.contrib.learn.io.numpy_input_fn(x, y, batch_size=4, num_epochs=1000)

estimator.fit(input_fn=input_fn, steps=1000)
estimator.evaluate(input_fn=input_fn, steps=1)
```

7 lines

Canned Estimators

Estimator

Keras Model

Layers

Datasets

Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS

XLA

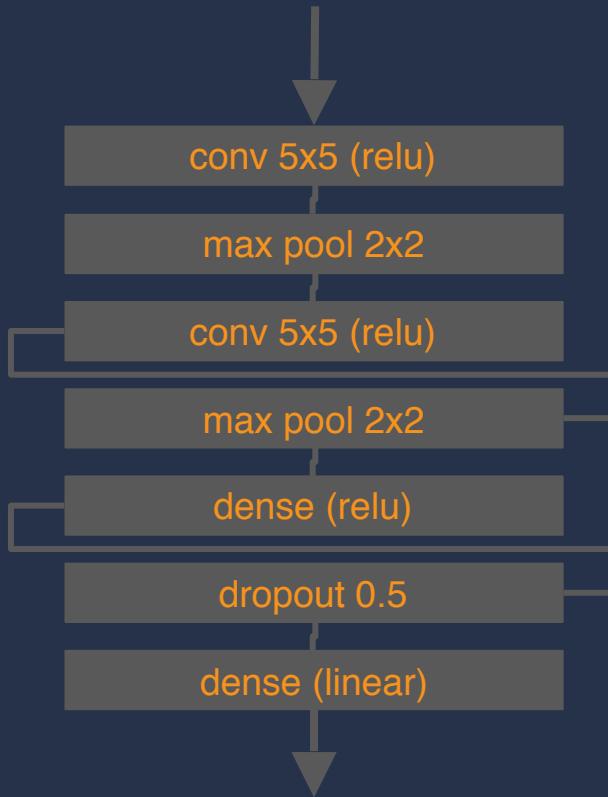
CP  
U

GP  
U

TPU

...







conv 5x5 (relu)

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

max pool 2x2

conv 5x5 (relu)

max pool 2x2

dense (relu)

dropout 0.5

dense (linear)





conv 5x5 (relu)

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

max pool 2x2

`x = tf.layers.max_pooling2d(x, kernel_size=[2,2], ...)`

conv 5x5 (relu)

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

max pool 2x2

`x = tf.layers.max_pooling2d(x, kernel_size=[2,2], ...)`

dense (relu)

`x = tf.layers.dense(x, activation_fn=tf.nn.relu)`

dropout 0.5

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

dense (linear)

`x = tf.layers.dense(x)`



Canned Estimators

Estimator

Keras  
Model

Layers

Datasets

Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS

XLA

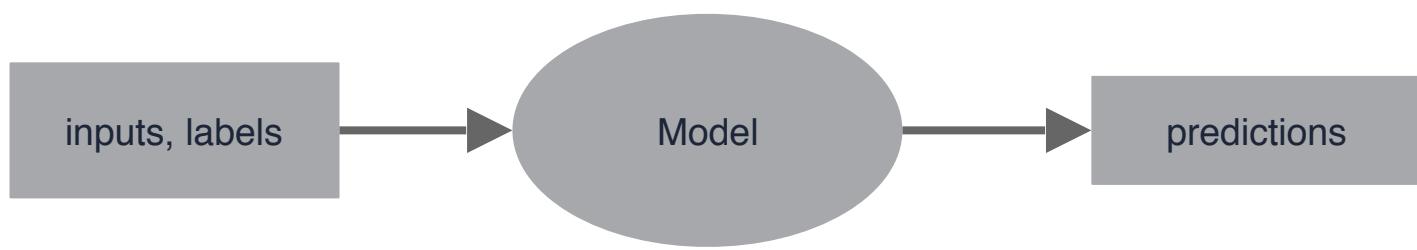
CP  
U

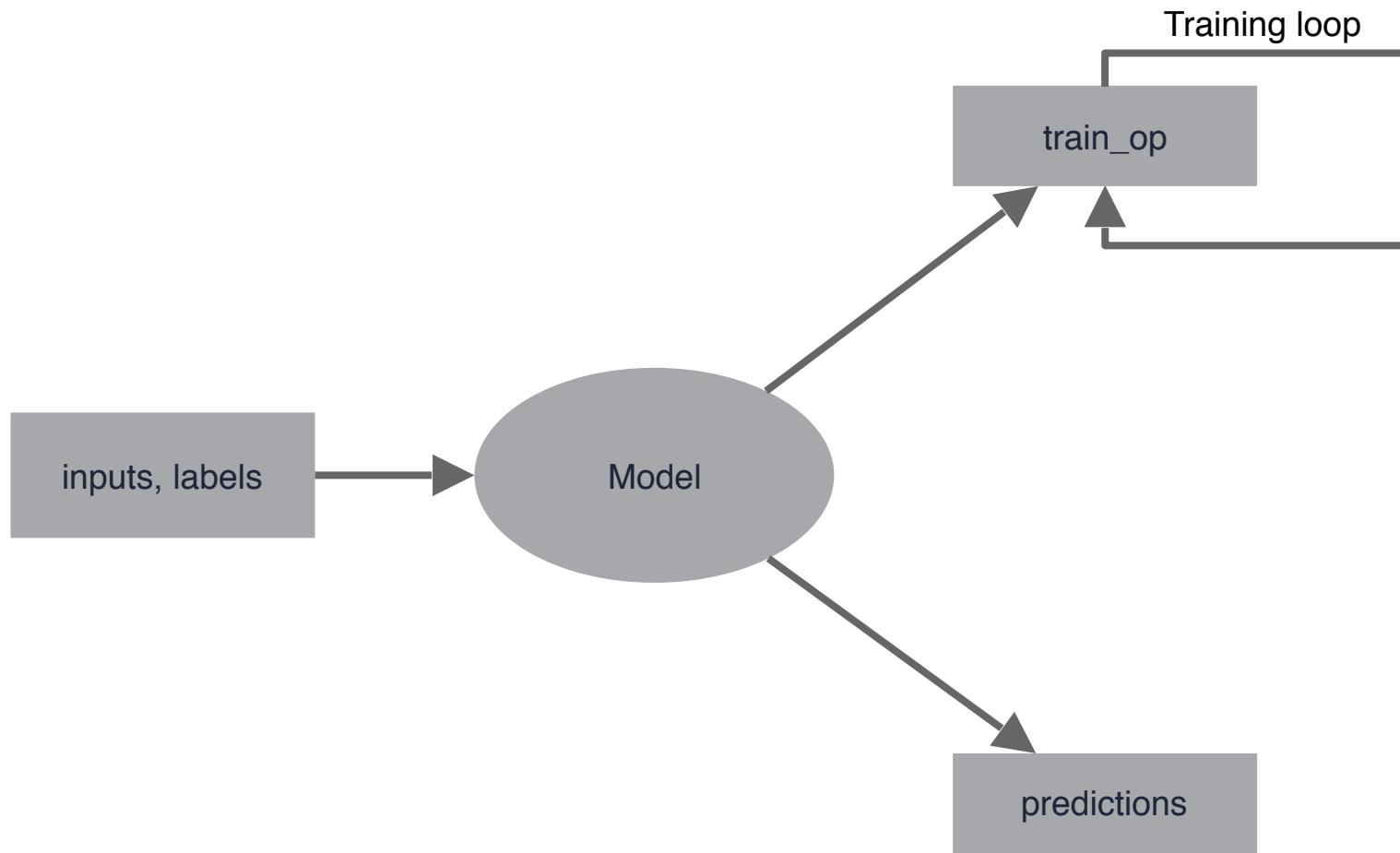
GP  
U

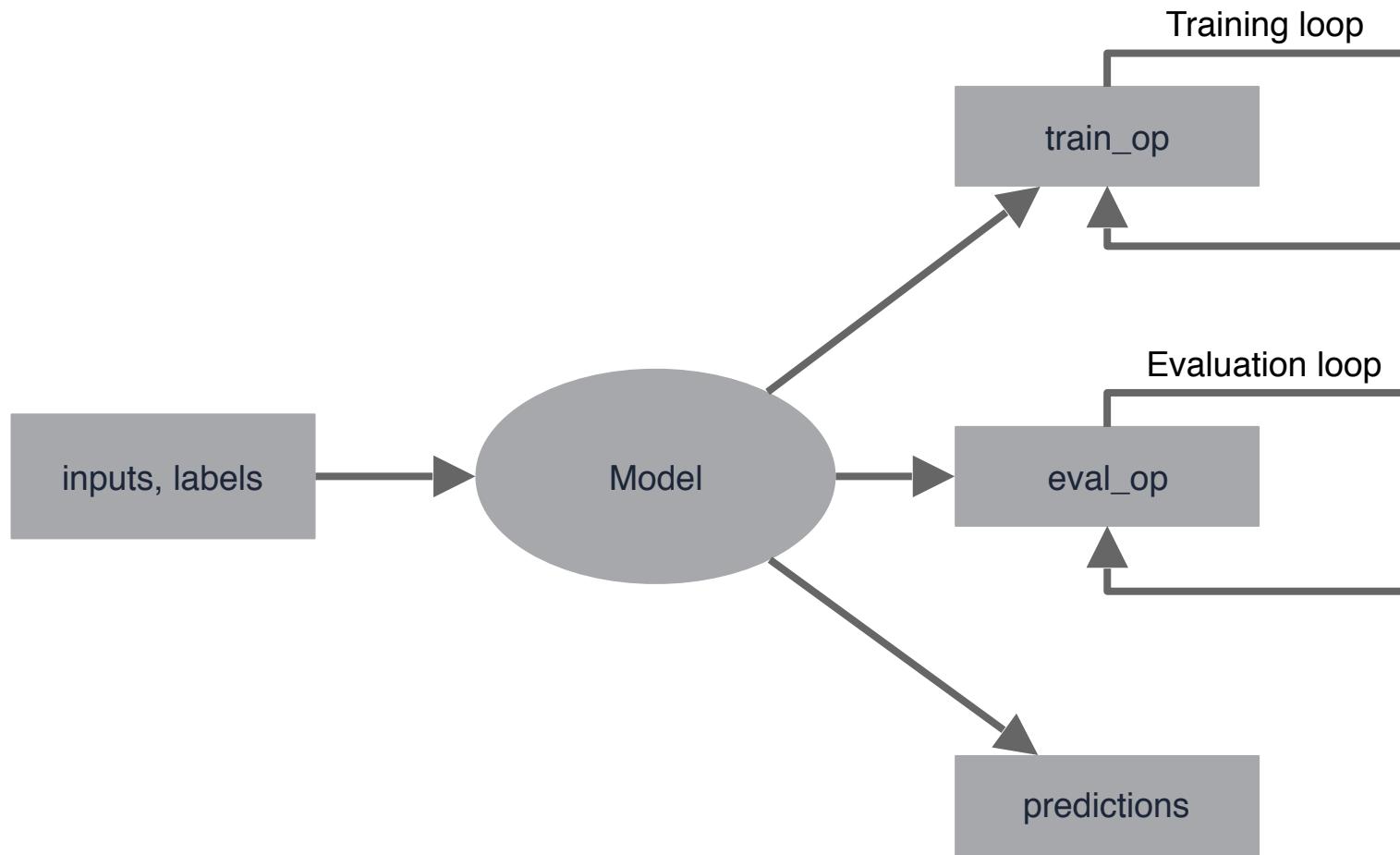
TPU

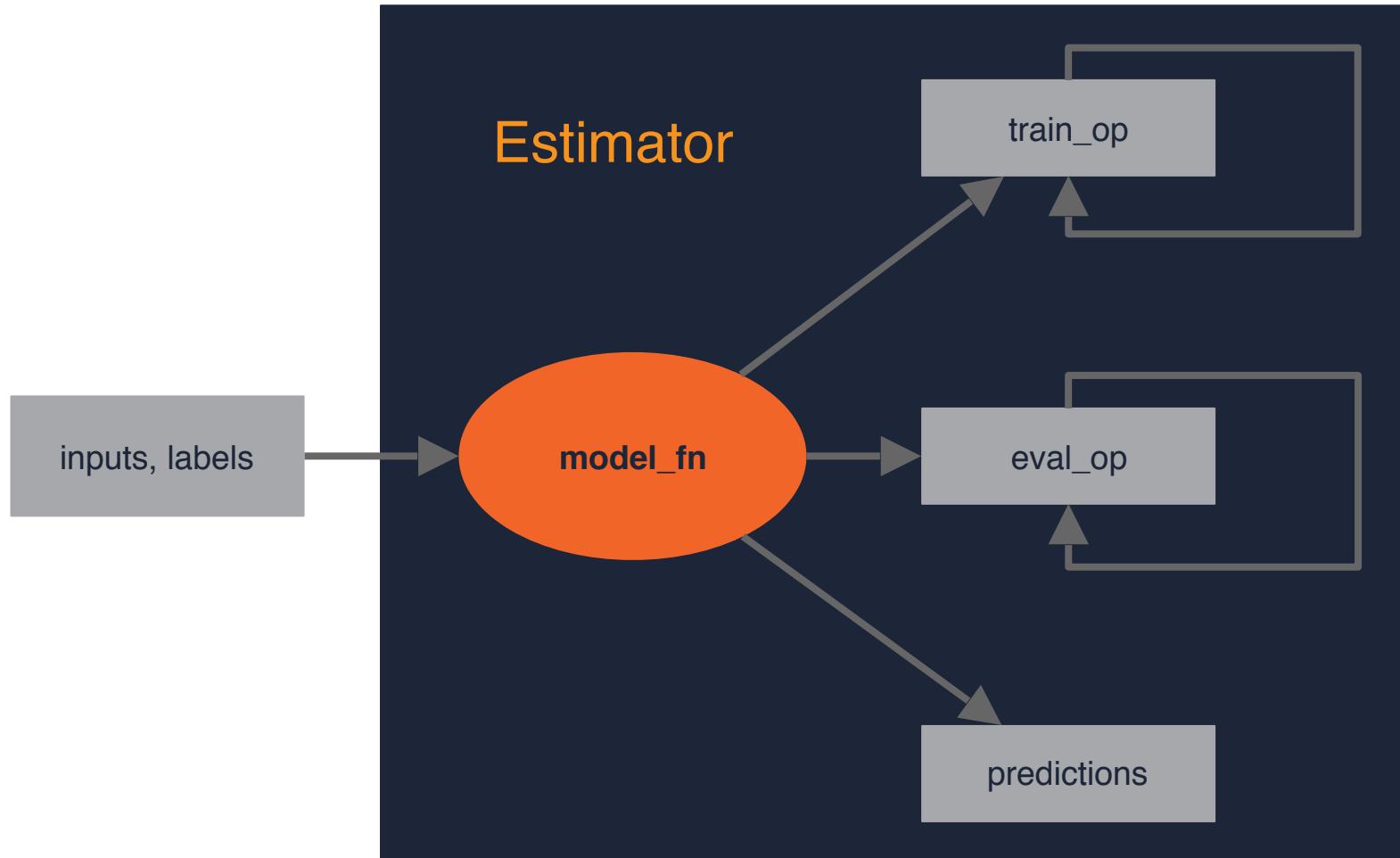
...

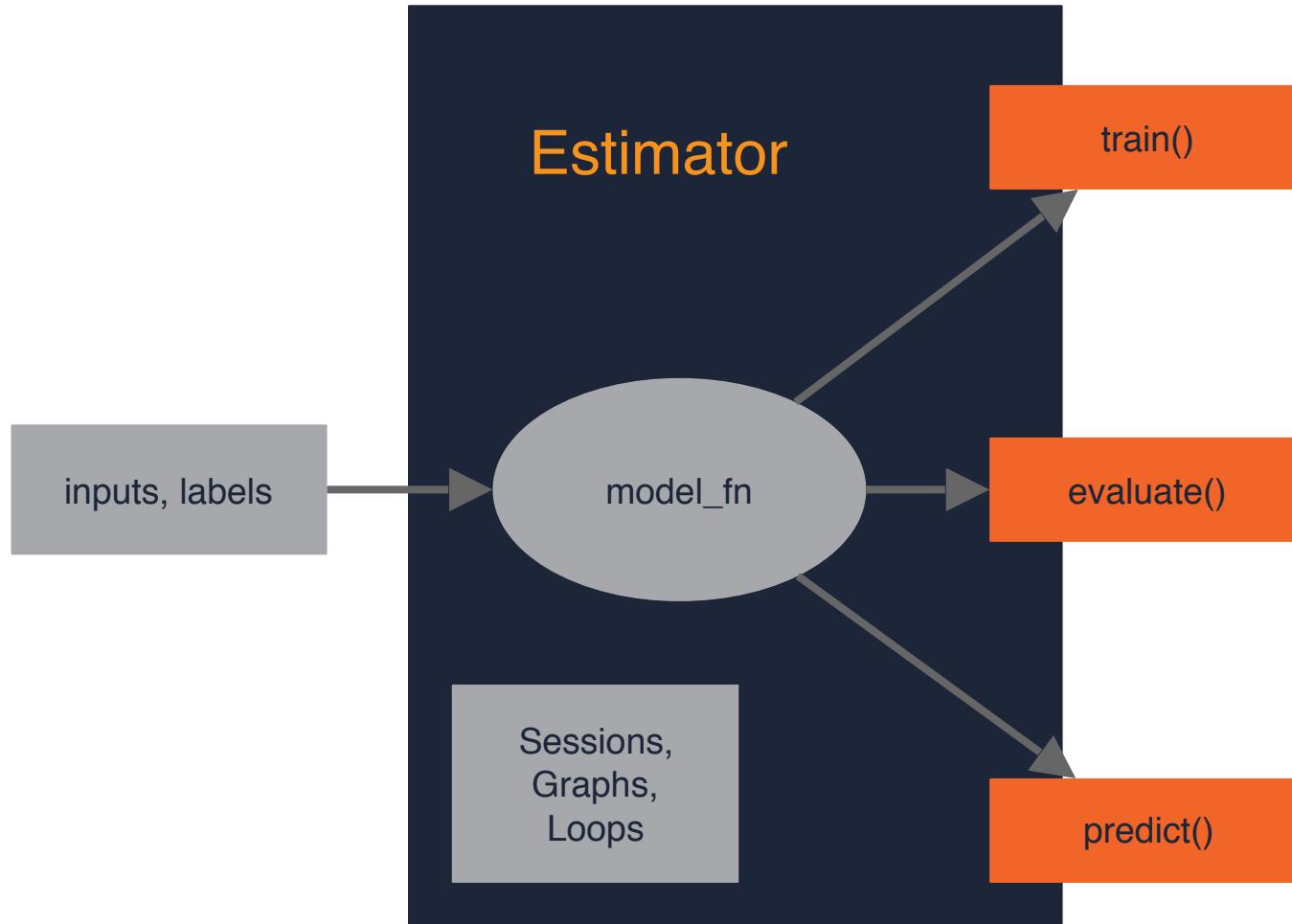


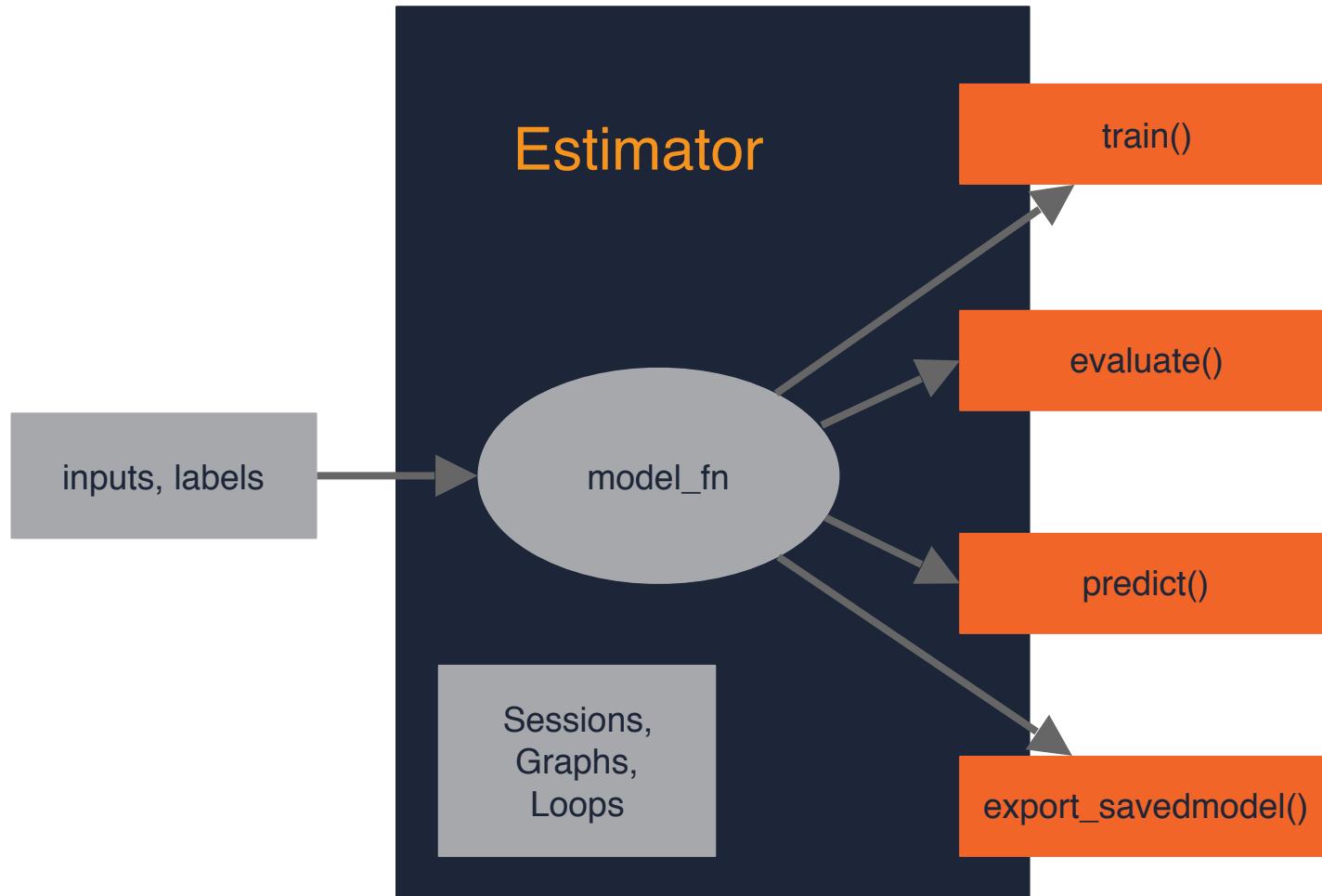












# Custom Estimators

- Input\_fn
- model\_fn
- train
- evaluate
- export\_savedmodel



Canned Estimators

# tf.keras

Estimator

Keras

Layers

Datasets

Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS

XLA

CPU

GPU

TPU

...



```
video = tf.keras.layers.Input(shape=(None, 150, 150, 3))
cnn = tf.keras.applications.InceptionV3(weights='imagenet',
                                         include_top=False,
                                         pool='avg')
cnn.trainable = False
```



Canned Estimators

# Canned Estimators

Estimator

Keras  
Model

Layers

Datasets

Python Frontend

C++

Java

Go

...

TensorFlow Distributed Execution Engine

CPU

GPU

Android

iOS

XLA

CP  
U

GP  
U

TPU

...



```
area = real_valued_column("square_foot"),
rooms = real_valued_column("num_rooms"),
zip_code = sparse_column_with_integerized_feature("zip_code", 100000)
```

```
regressor = DNNRegressor(  
    feature_columns=[area, rooms, embedding_column(zip_code, 8)],  
    hidden_units=[1024, 512, 256])
```

```
regressor.train(train_input_fn)
```

```
regressor.evaluate(eval_input_fn)
```



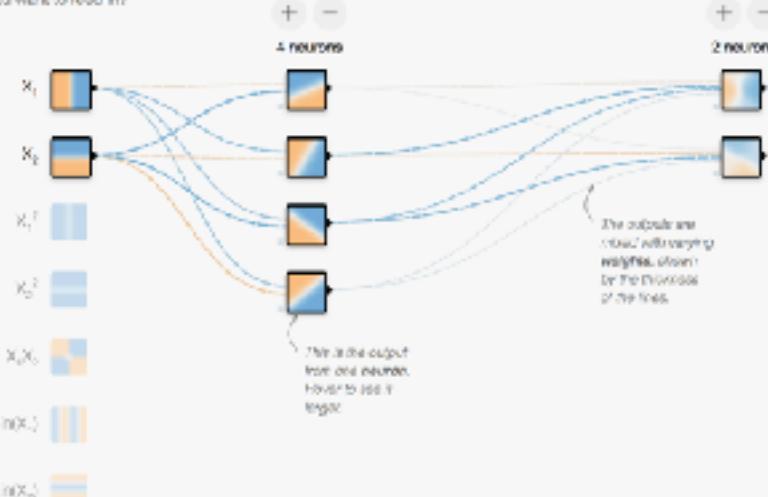
# TensorFlow v1.4

Code time!!



# http://playground.tensorflow.org

Epoch: 000,000      Learning rate: 0.001      Activation: Tanh      Regularization: None      Regularization ratio: 0      Problem type: Classification

**DATA**  
Which dataset do you want to use?  
  
  
  
  
Ratio of training to test data: 50%  
Noise: 0  
Batch size: 10  
**FEATURES**  
Which properties do you want to feed in?  


+ - 2 HIDDEN LAYERS

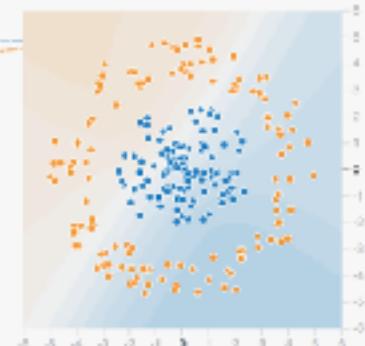
+ - 2 neurons

+ - 2 neurons

*This is the output from one neuron. How to see it right?*

*The outputs are mixed with varying weights, shown by the thickness of the lines.*

**OUTPUT**  
Test loss: 0.103  
Training loss: 0.009



Colors shows data, neuron and weight values.

Show test data     Discretize output