TensorFlow Tutorial

Strata NYC 2016

Your guides

Martin Wicke
@martin_wicke

Josh Gordon
@random_forests

Slides and code

goo.gl/nrdsxM

Welcome and Logistics

- We will present the tutorial in Jupyter notebooks.
- To run them on your machine, you will need a working TensorFlow installation (v0.10+)
- Follow the instructions at **goo.gl/nrdsxM** to install TensorFlow and Jupyter on your laptop

Outline

Part 1) Concepts

- Tensors, graphs, sessions
- Inference, loss, training

Exercises

- Fibonacci sequence
- Linear regression
- Taxis
- Linear vs Nonlinear Classifiers

Part 2) Images

- Classifying images
- Image Retraining

Exercises

- Deep MNIST
- Convolutional MNIST

Outline cont’d

Part 3) Text

- Representing words

Exercises

- Word embeddings
- DBpedia

Part 4) Free play

- Community contributions

Experiments

- Deep Dream
- Neural Style

What’s Machine Learning?

Images: Wikimedia

AlphaGo

BBC News Sport Weather iPlayer TV Radio More Search

NEWS Find local news

Home UK World Business Politics Tech Science Health Education Entertainment & Arts More

Technology

Google achieves AI 'breakthrough' at Go

An artificial intelligence program developed by Google beats Europe's top player at the ancient Chinese game of Go, about a decade earlier than expected.

27 January 2016 | Technology

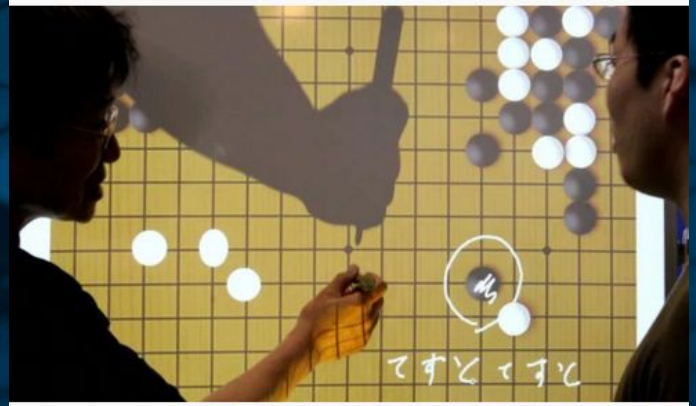
How did they do it?

What is the game Go?

Facebook trains AI to beat humans at Go

Google's AI just cracked the game that supposedly no computer could beat

By Mike Murphy | January 27, 2016



Going up. (Reuters/Kiyoshi Ota)

Computers have slowly started to encroach on activities we previously believed only the brilliantly sophisticated human brain could handle. IBM's Deep Blue supercomputer beat Grand Master Garry Kasparov at chess in 1997, and in 2011 IBM's Watson beat former human winners at the quiz game *Jeopardy*. But the ancient board game Go has long been one of the major goals of artificial intelligence research. It's understood to be one of the most difficult games for computers to handle due to the sheer number of possible moves a player can make at any given point. Until now, that is.

Open Source Models

github.com/tensorflow/models

Inception

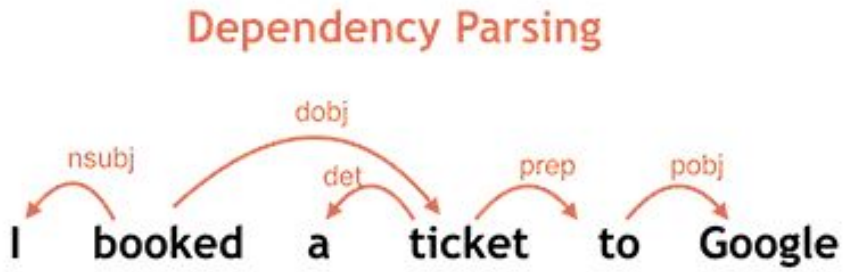
An Alaskan Malamute (left) and a Siberian Husky (right). Images from Wikipedia.

https://research.googleblog.com/2016/08/improving-inception-and-image.html

Show and Tell

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

Parsey McParseface



https://research.googleblog.com/2016/05/announcing-syntaxnet-worlds-most.html

Text Summarization

Original text

- *Alice and Bob took the train to visit the zoo. They saw a **baby giraffe, a lion, and a flock of colorful tropical birds**.*

Abstractive summary

- *Alice and Bob visited the zoo and saw **animals and birds**.*

https://research.googleblog.com/2016/08/text-summarization-with-tensorflow.html

How Can You Get Started?

Three ways, with varying complexity:

1. Use a Cloud-based API (e.g., Vision)

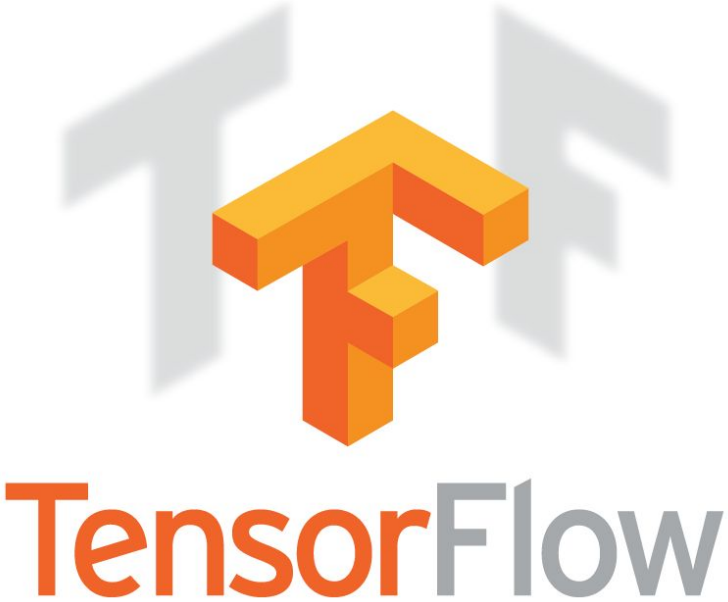
2. Use an existing model architecture, and retrain it or fine tune on your dataset (e.g., Inception)

3. Develop your own machine learning models for new problems

More flexible, but more effort

Install TensorFlow

goo.gl/nrdsxM

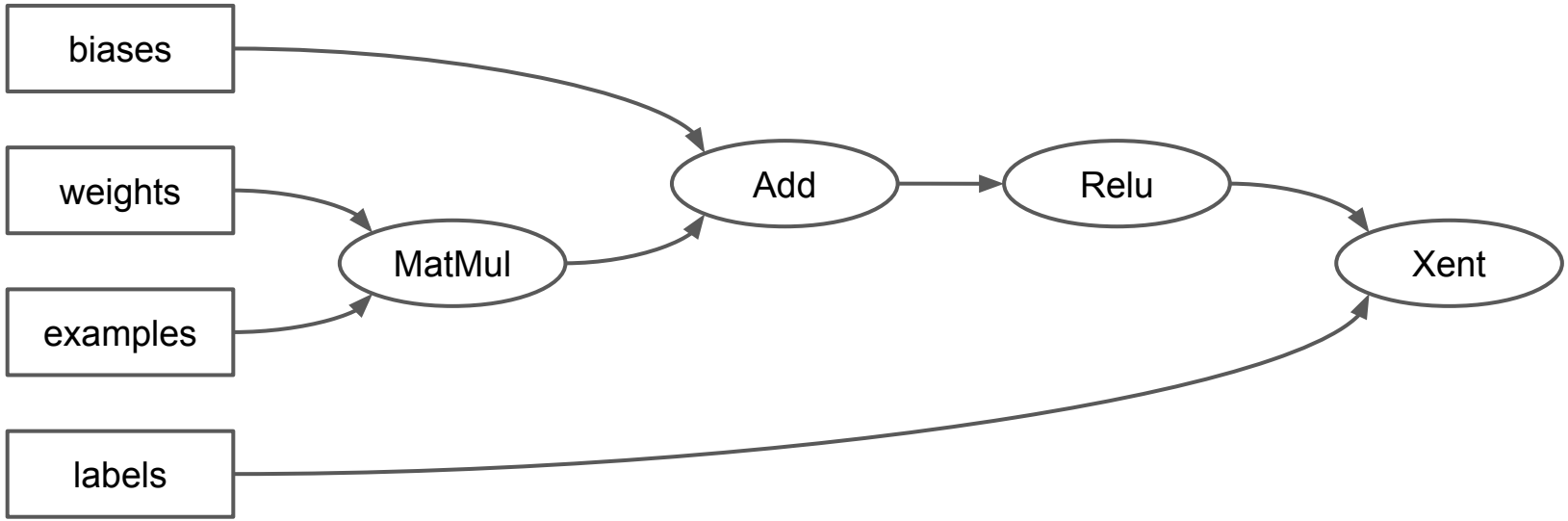


- **Open source** Machine Learning library
- Especially useful for **Deep Learning**
- For research **and** production
- **Apache 2.0** license

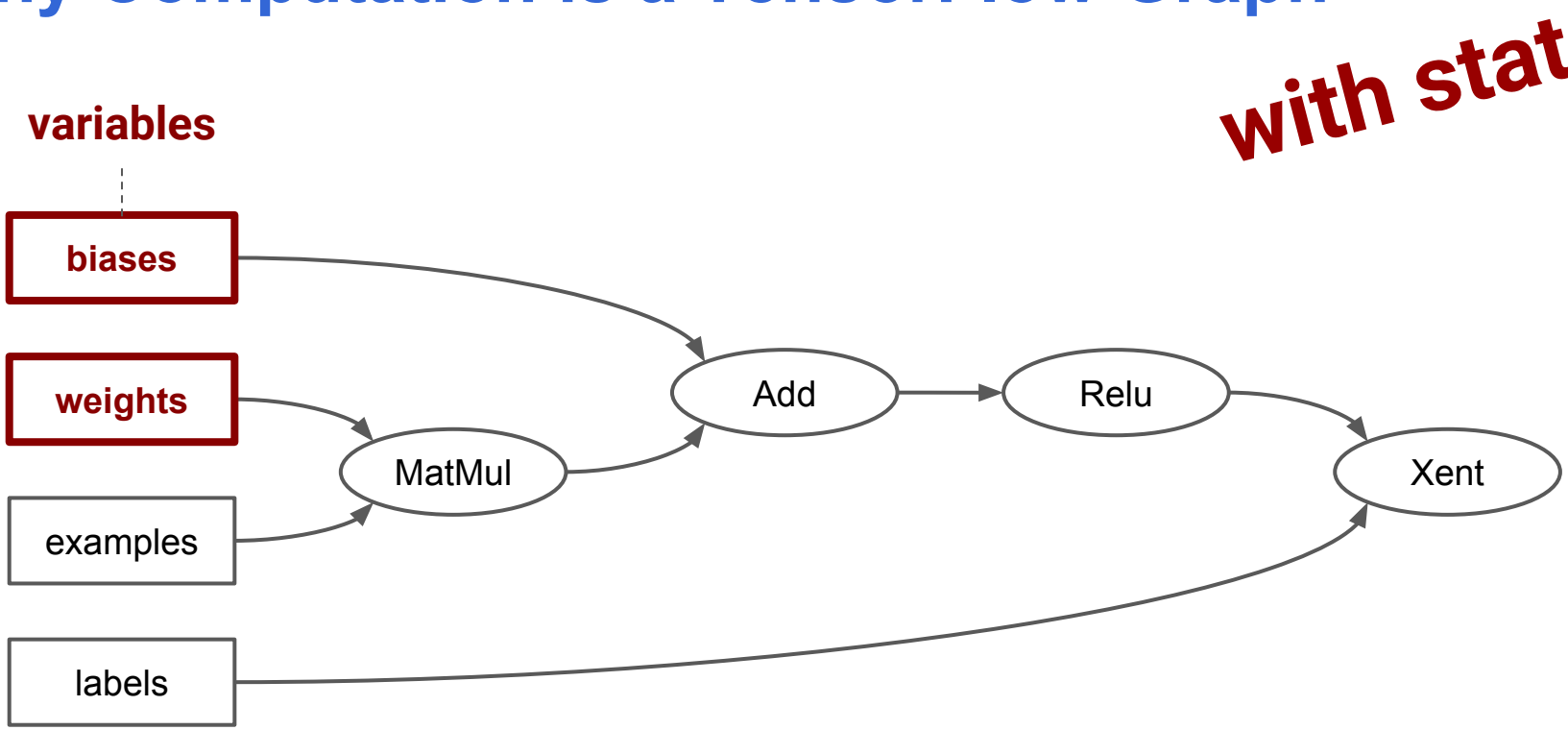
A multidimensional array.

A graph of operations.

Any Computation is a TensorFlow Graph

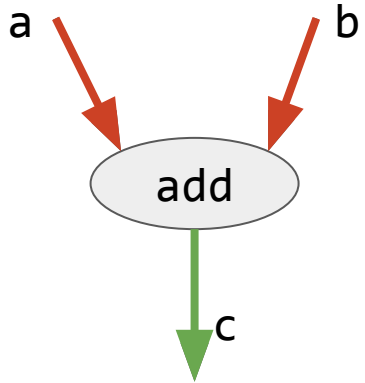


Any Computation is a TensorFlow Graph



Build a graph; then run it.

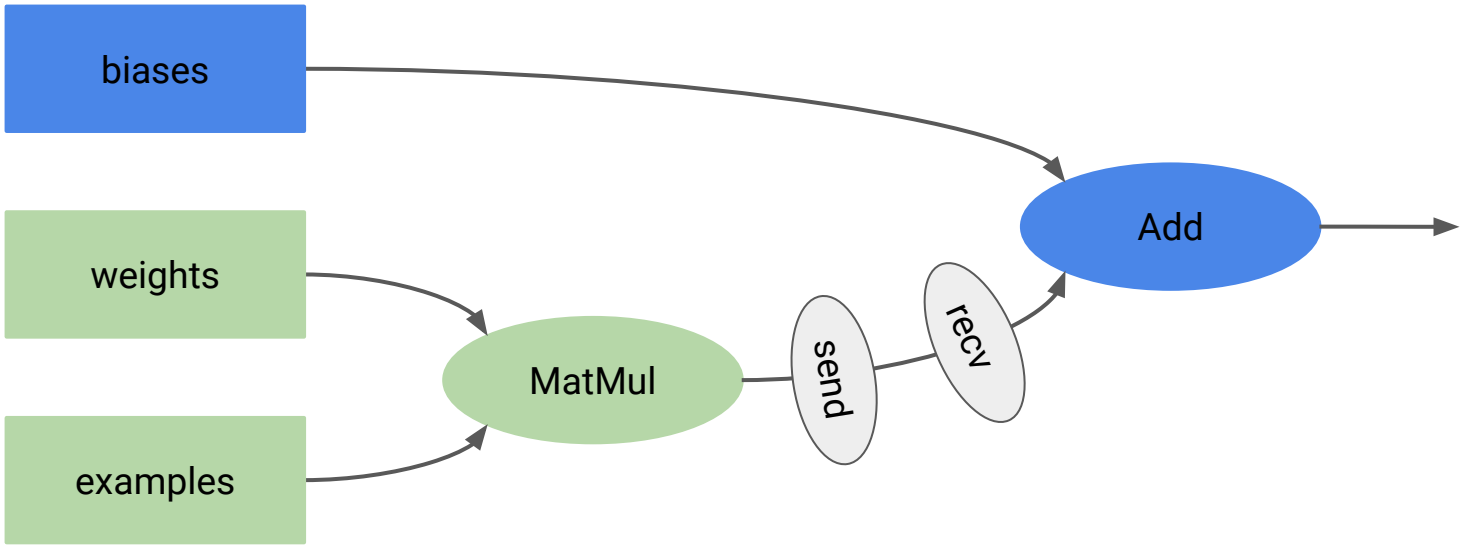
```
...
c = tf.add(a, b)


...

session = tf.Session()
value_of_c = session.run(c, {a=1, b=2})
```



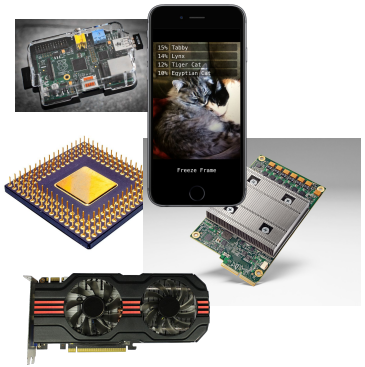
But why?

But, why?

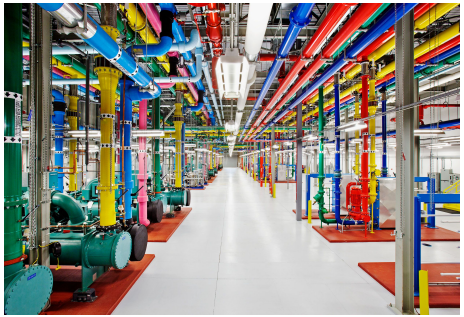
Graphs can be processed, compiled, remotely executed, assigned to devices.



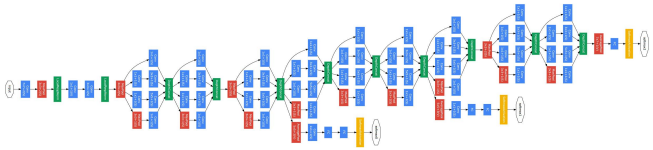
Three sources of complexity



Heterogenous systems

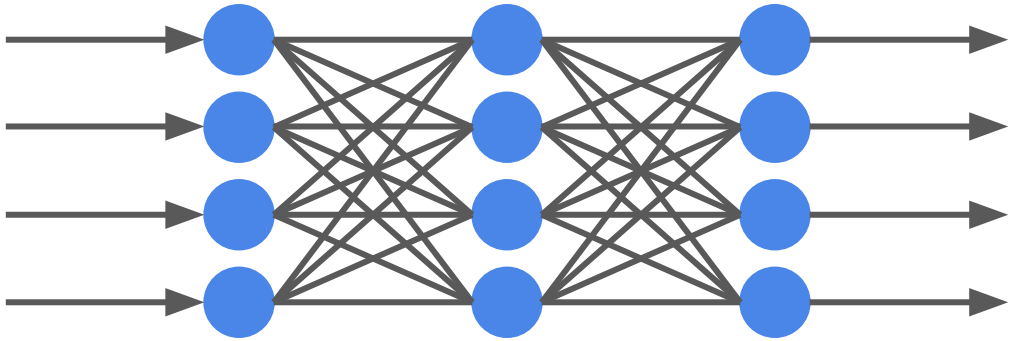


Distributed systems

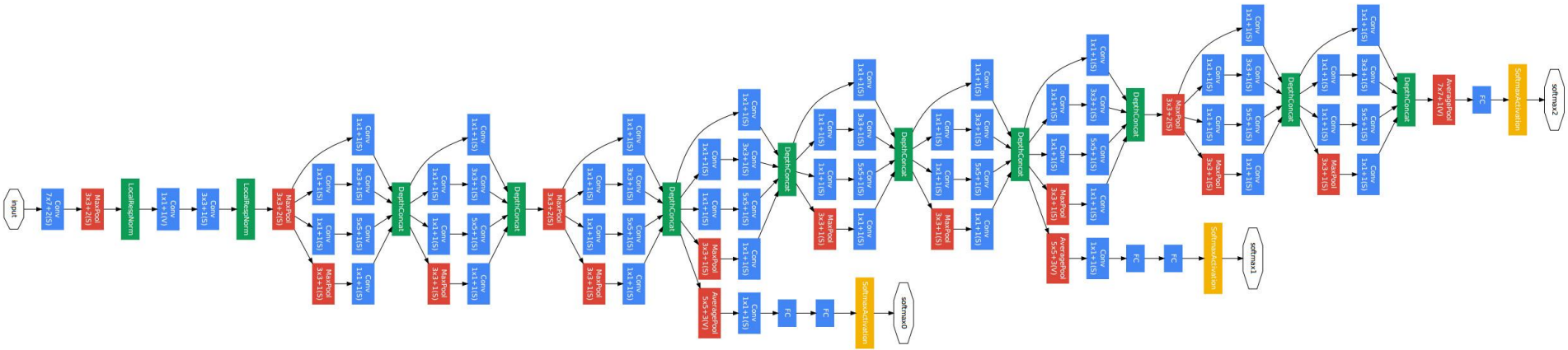


Large models

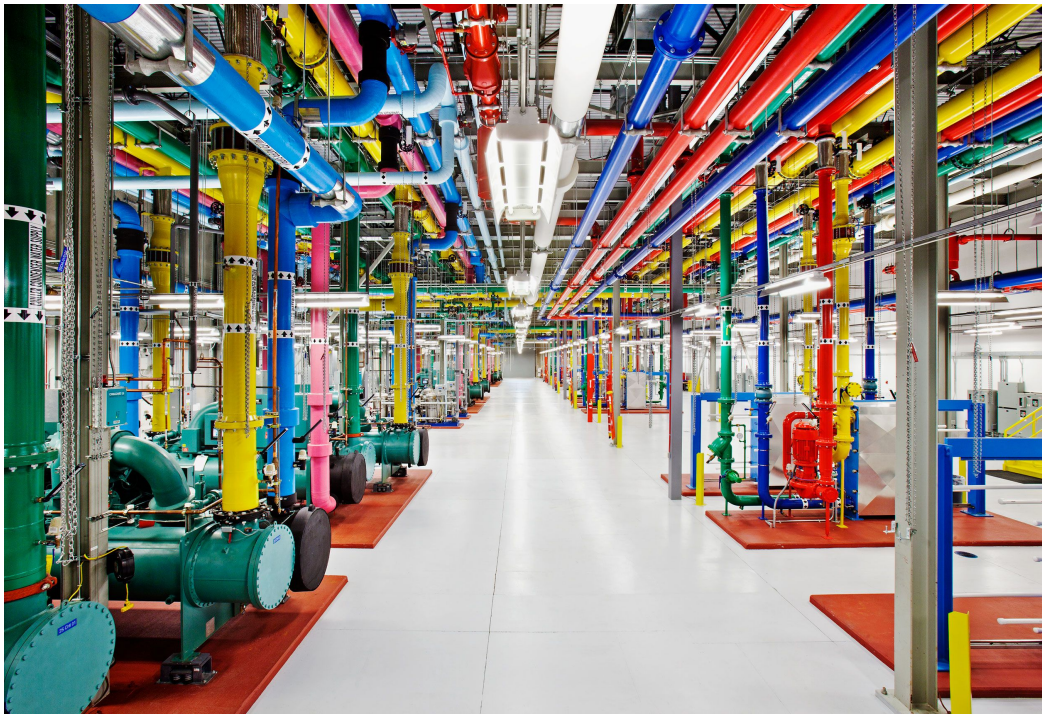
Neural networks



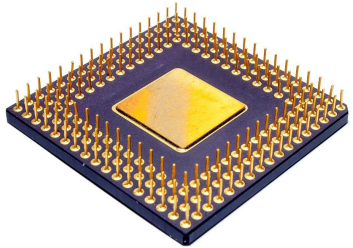
Inception (2015)



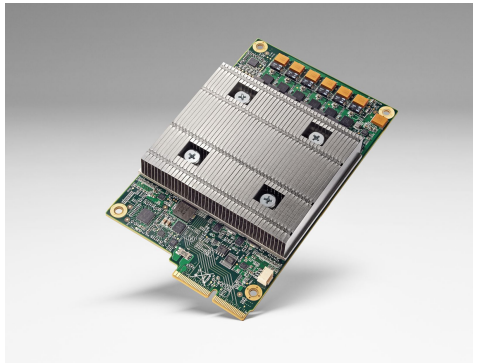
Distributed systems



Heterogenous systems

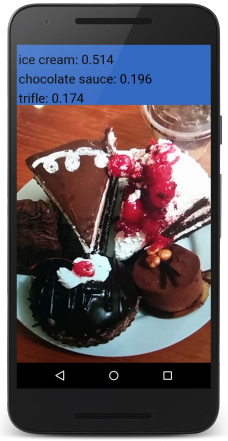


CPU

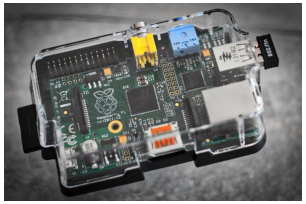


TPU

Android



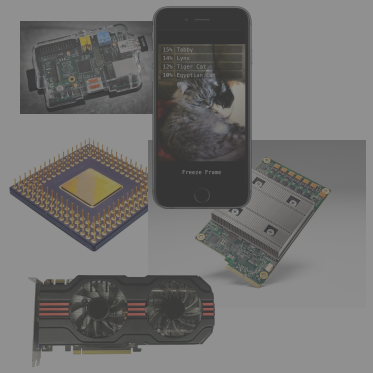
iOS



Raspberry Pi

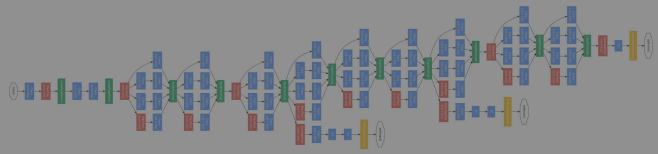
GPU

TensorFlow handles complexity for you ...



Heterogenous systems

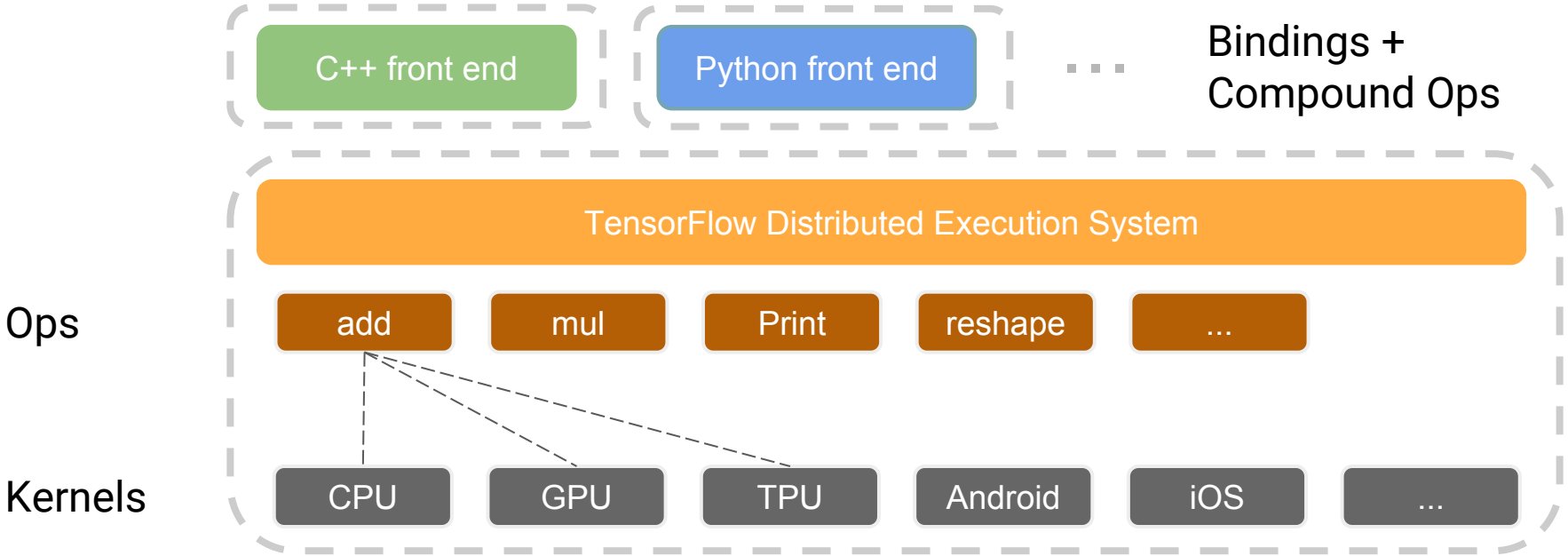
Distributed systems



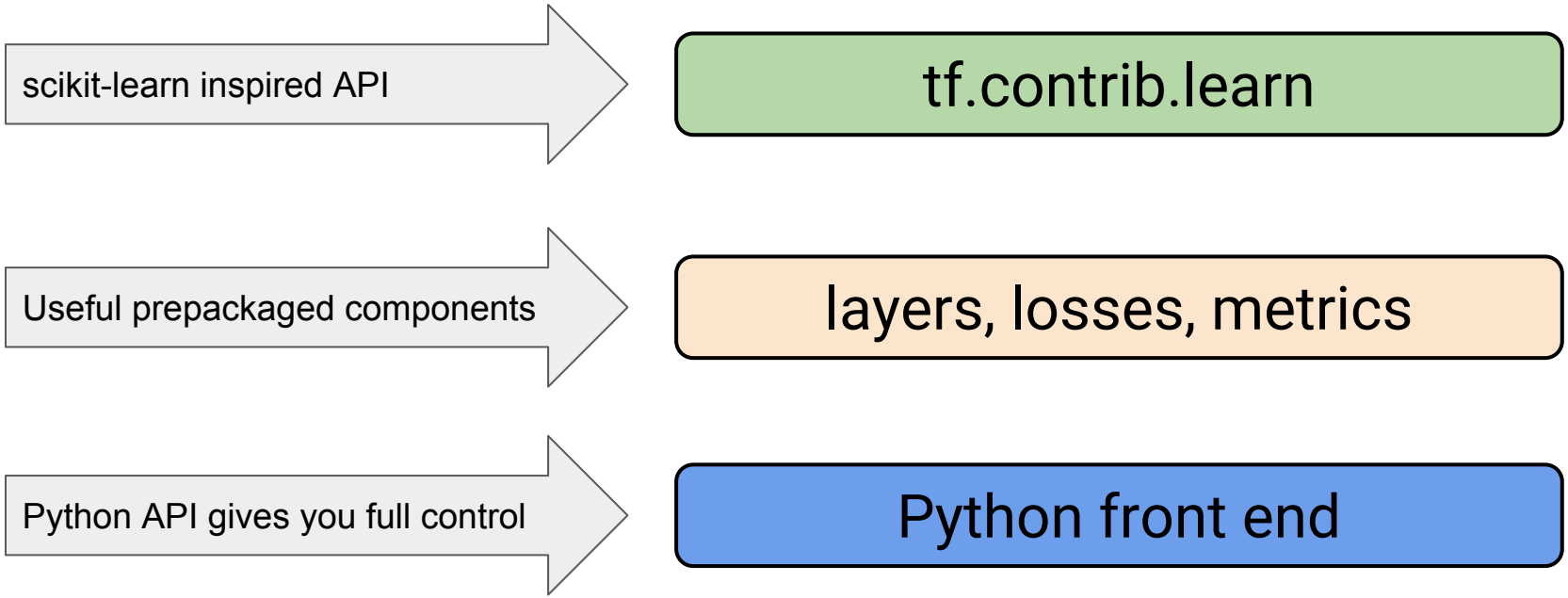
Complex models

… so you can focus on your ideas

Architecture



Toolkit hierarchy

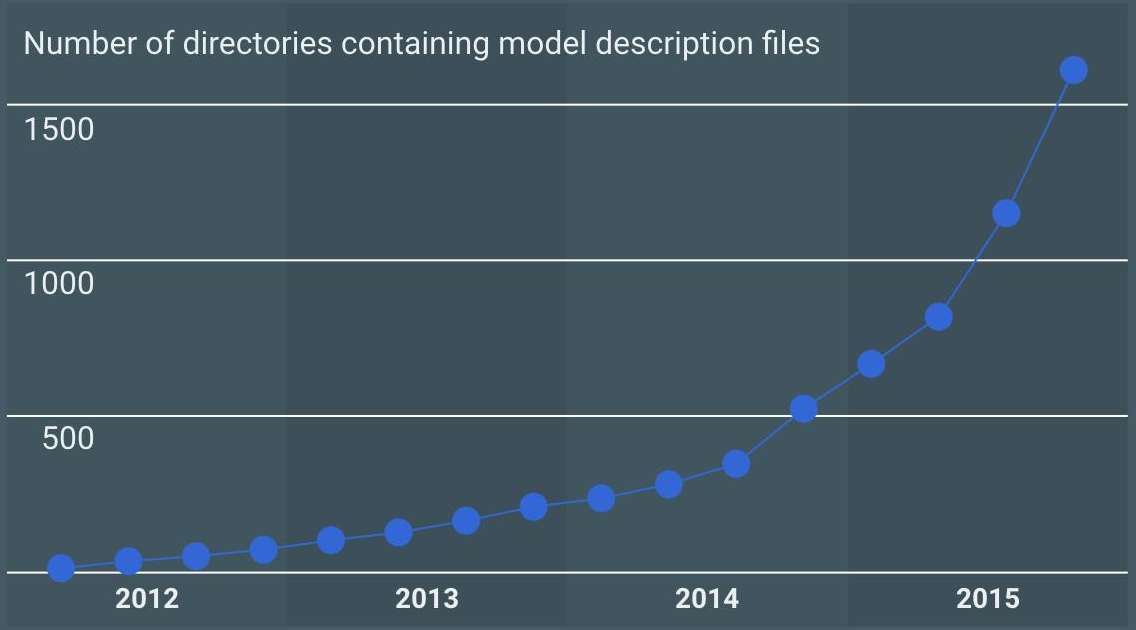


Community contributions

Lab #1 Warm up
goo.gl/nrdsxM

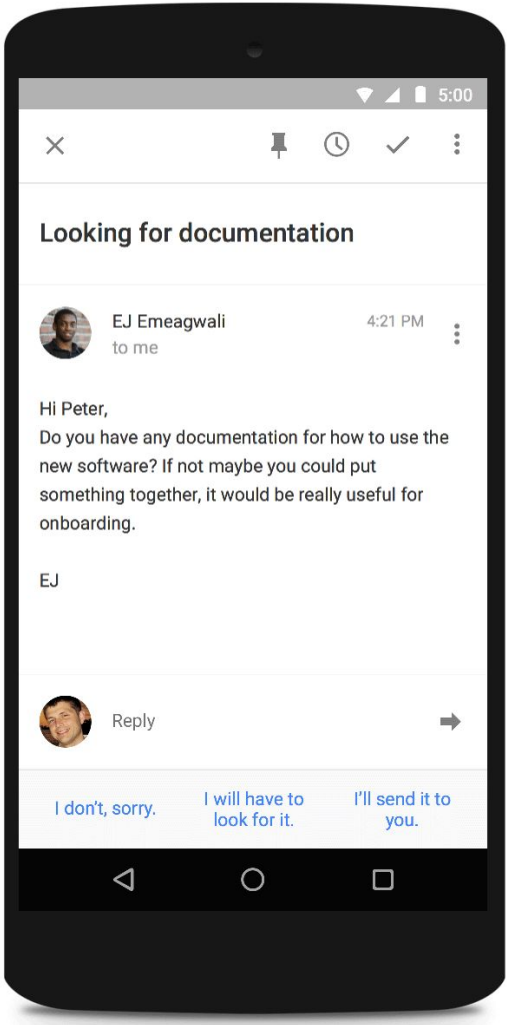
TensorFlow at Google

Growing use of deep learning at Google



Across many areas

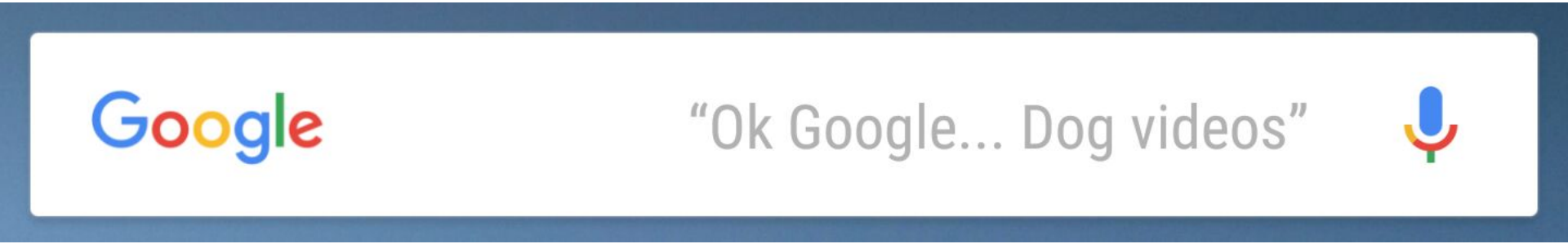
- AlphaGo
- Apps
- Maps
- Photos
- Gmail
- Speech
- Android
- YouTube
- Translation
- Robotics Research
- Image Understanding
- Natural Language Understanding
- Drug Discovery



Smart reply in Inbox by Gmail

10%

of all responses sent on mobile

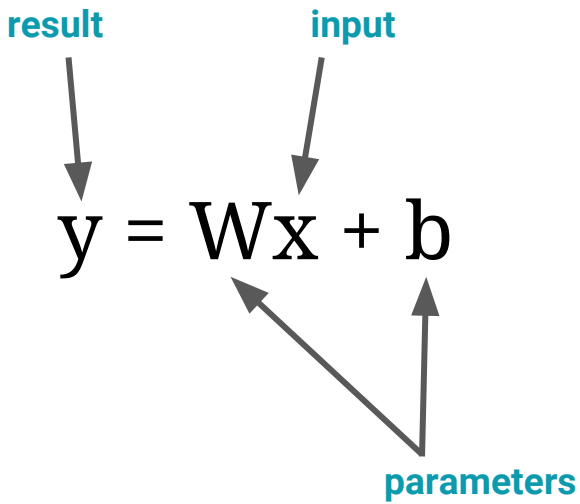

Google
“Ok Google... Dog videos”

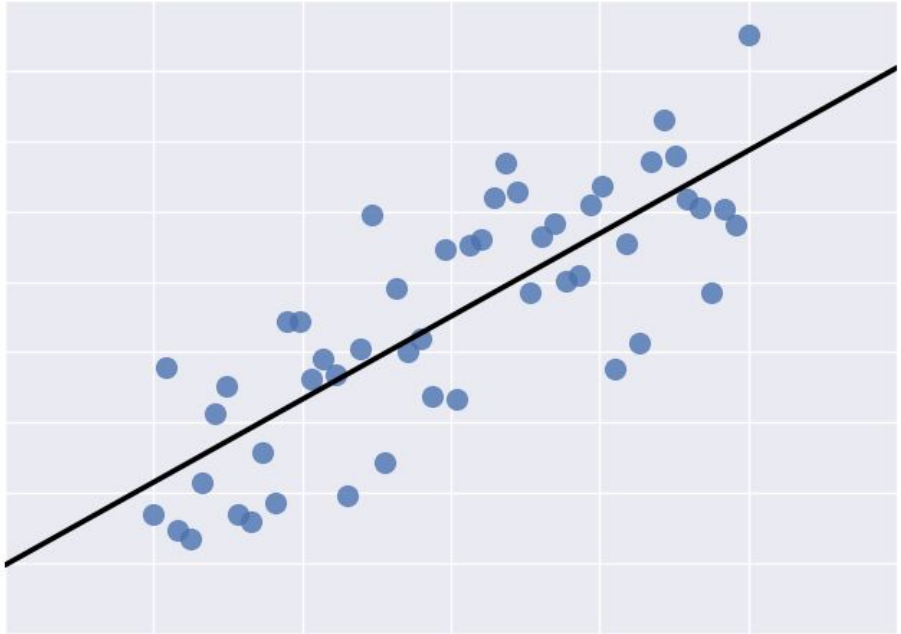

9:50
Dogs
Feb 28
Feb 26


EXIT

Linear Regression

Linear Regression





What are we trying to do?

Mystery equation: `y = 0.1 * x + 0.3 + noise`

Model: `y = W * x + b`

Objective: Given enough (`x`, `y`) value samples, figure out the value of `W` and `b`.

Pattern

1. **Inference** (“code to predict y”)
2. **Loss** (“code to quantify how close our prediction was”)
3. **Eval** (“check your accuracy”)
4. **Train** (“code to update our variables to improve the next prediction”)

Inference

```
import tensorflow as tf
W = tf.get_variable(shape=[], name='W')
b = tf.get_variable(shape=[], name='b')
x = tf.placeholder(shape=[None],
                   dtype=tf.float32,
                   name='x')
y = tf.matmul(W, x) + b

with tf.Session() as sess:

  sess.run(tf.initialize_all_variables())

  print(sess.run(y, feed_dict={x: x_in}))
```

Build the graph

Prepare execution environment

Initialize variables

Run the computation (usually often)

Variables Must be Initialized

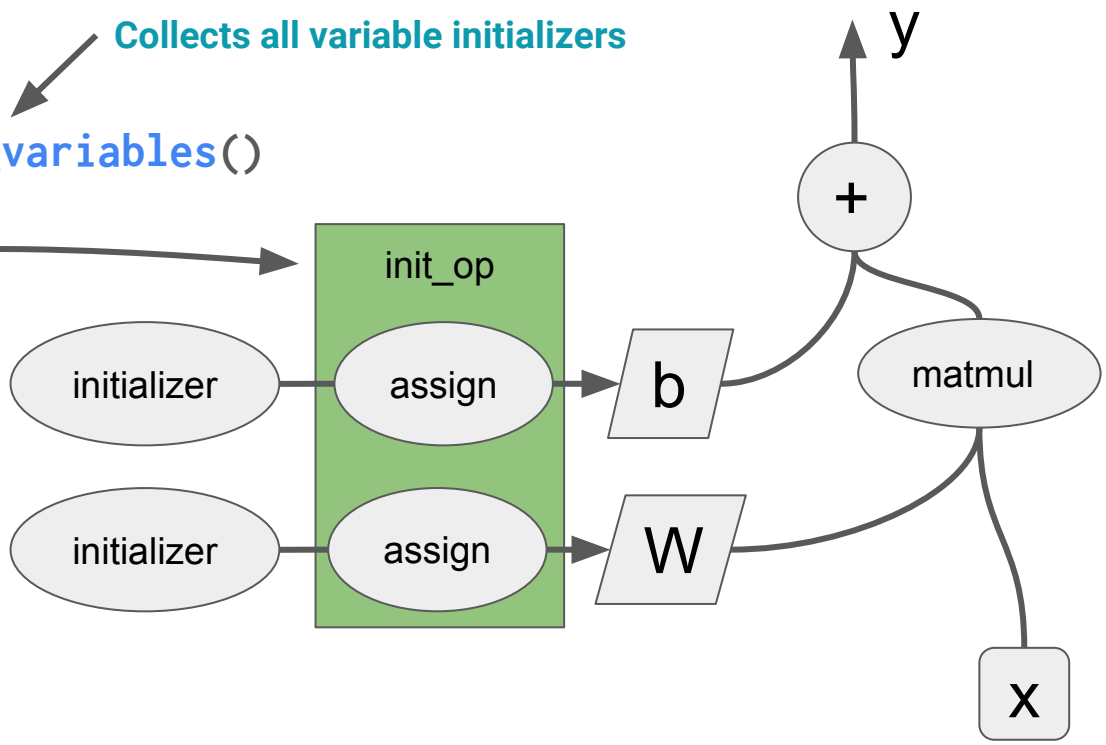
Collects all variable initializers

```
init_op = tf.initialize_all_variables()
```

Makes an execution environment

```
sess = tf.Session()

sess.run(init_op)
```

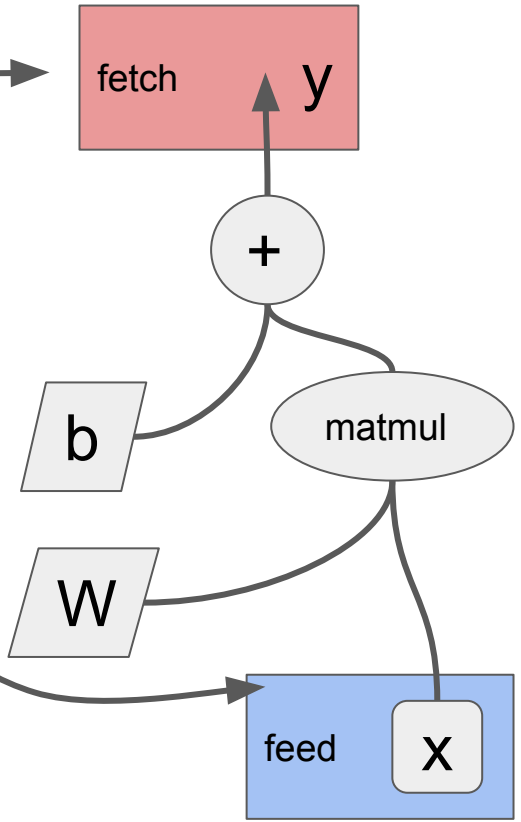
Actually initialize the variables



Running the Computation

```
x_in = [3]

sess.run(y, feed_dict={x: x_in})
```

- Only what's used to compute a fetch will be evaluated
- All Tensors can be fed, but all placeholders must be fed



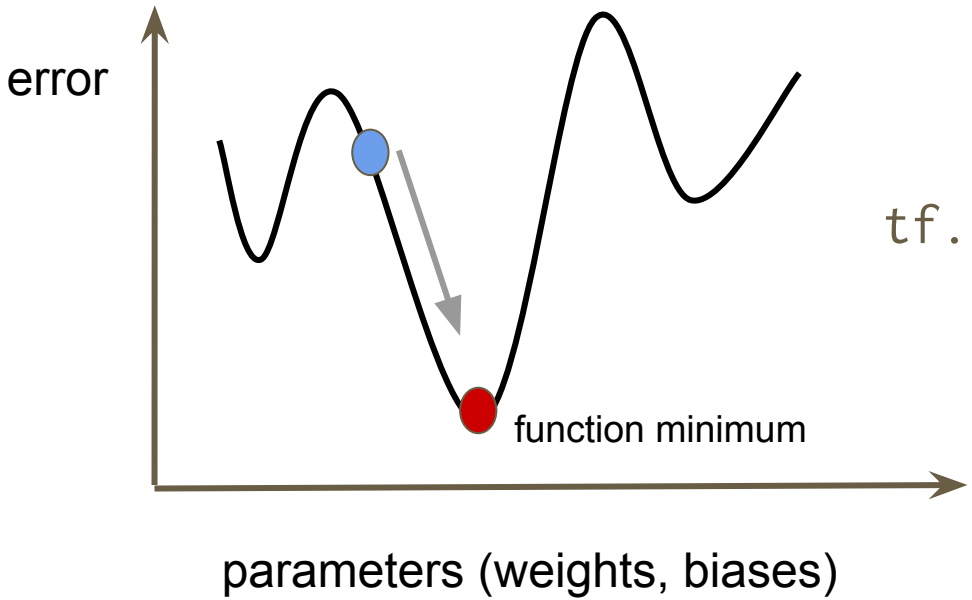
Define a Loss

Given x, y_{label}, compute a loss, for instance:

$$L = (y - y_{label})^2$$

```
# Create an operation that calculates loss.
loss = tf.reduce_mean(tf.square(y - y_data))
```

Minimize loss: optimizers



`tf.train.GradientDescentOptimizer`

Train

Feed (x, y_{label}) pairs and adjust W and b to decrease the loss.

$$W \leftarrow W - \eta \, (dL/dW)$$

$$b \leftarrow b - \eta \, (dL/db)$$

TensorFlow computes gradients automatically

```
# Create an optimizer

optimizer = tf.train.GradientDescentOptimizer(0.5)

# Create an operation that minimizes loss.

train = optimizer.minimize(loss)
```

Learning rate

Building graphs looks *mostly* like numpy

With special functions for deep learning

Numpy	TensorFlow
add mul matmul … sum …	add mul matmul … reduce_sum … sigmoid relu …

2_linear_regression_model.ipynb

1. Import libraries (cell 1.1)
2. Create input data (cell 1.2)
3. Build inference graph (cell 1.3)
    a. Create Variables to hold weights and biases.
    b. Create Operations that produce logistic outputs.
4. Build training graph (cell 1.4)
    a. Loss
    b. Optimizer
    c. Train_op: Operation that minimizes Loss
5. Create session and run initialization (cell 1.5)
6. Perform training (cell 1.6)

Exercises to run

1. Run all the cells
2. Uncomment 1.5 and see the graph
3. Uncomment 1.8 and compare values
4. Try to train a different linear function
    a. If you can’t visualize the values, try changing pylab.ylim()
5. Try a different initialization function.
    a. tf.random_normal
6. Try a different optimizer
    a. tf.AdagradOptimizer

Lab #2 Linear Regression
goo.gl/nrdsxM

tf.learn

That seemed complicated...

```
import tensorflow as tf

R = tf.contrib.learn.LinearRegressor(feature_columns=[
    tf.contrib.layers.real_valued_column('', dimension=5)])

R.fit(x=data[:,0:5], y=target, batch_size=100, max_steps=1000)

R.evaluate(x=eval_data[:,0:5], y=target_eval

R.predict(x=np.asarray([1.5, 3.4, 1.6, 1.9, 2.0]))
```

tf.learn

- High level API for TensorFlow
- Implements scikit-learn API
- One-liners create the graph for us behind the scenes.

```
classifier = learn.DNNClassifier(idden_units=[10,20,10],n_classes=3)
```

Linear Regression with Taxi Data

Problem

- Predict tip amount based on distance traveled (and other features)

Solution

- Prepare a numpy ndarray.
- Each column represents a feature
- Last column is the target
- Use TF.Learn to train a Linear and Deep Regressor; compare the results

Lab #2b TF.Learn
goo.gl/nrdsxM

Linear vs Nonlinear classifiers

Tinker With a **Neural Network** Right Here in Your Browser.
Don't Worry, You Can't Break It. We Promise.

Iterations: 000,000

Learning rate: 0.03

Activation: Tanh

Regularization: None

Regularization rate: 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?

X_1

X_2

X_1^2

X_2^2

X_1X_2

$\sin(X_1)$

2 HIDDEN LAYERS

4 neurons

2 neurons

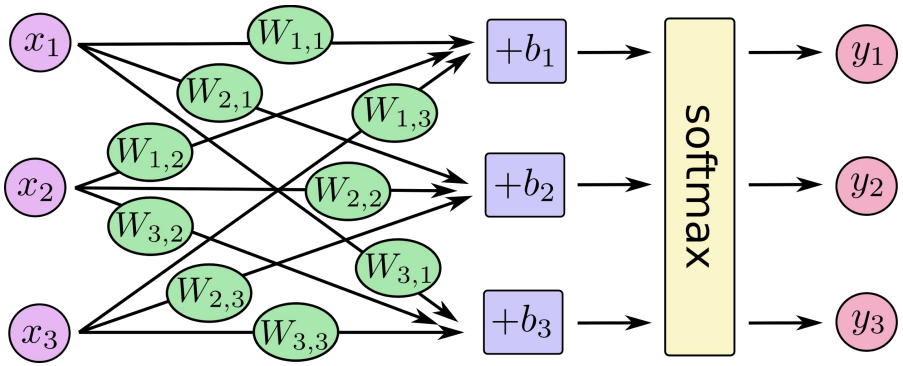
*This is the output from one **neuron**. Hover to see it larger.*

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

OUTPUT

Test loss 0.513

Training loss 0.507



$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \end{bmatrix} = \text{softmax}\left(\begin{matrix} W_{1,1}x_1 + W_{1,2}x_2 + W_{1,3}x_3 + b_1 \\ W_{2,1}x_1 + W_{2,2}x_2 + W_{2,3}x_3 + b_2 \\ W_{3,1}x_1 + W_{3,2}x_2 + W_{3,3}x_3 + b_3 \end{matrix} \right)$$

IS THIS A

CAT or DOG?

CAT DOG

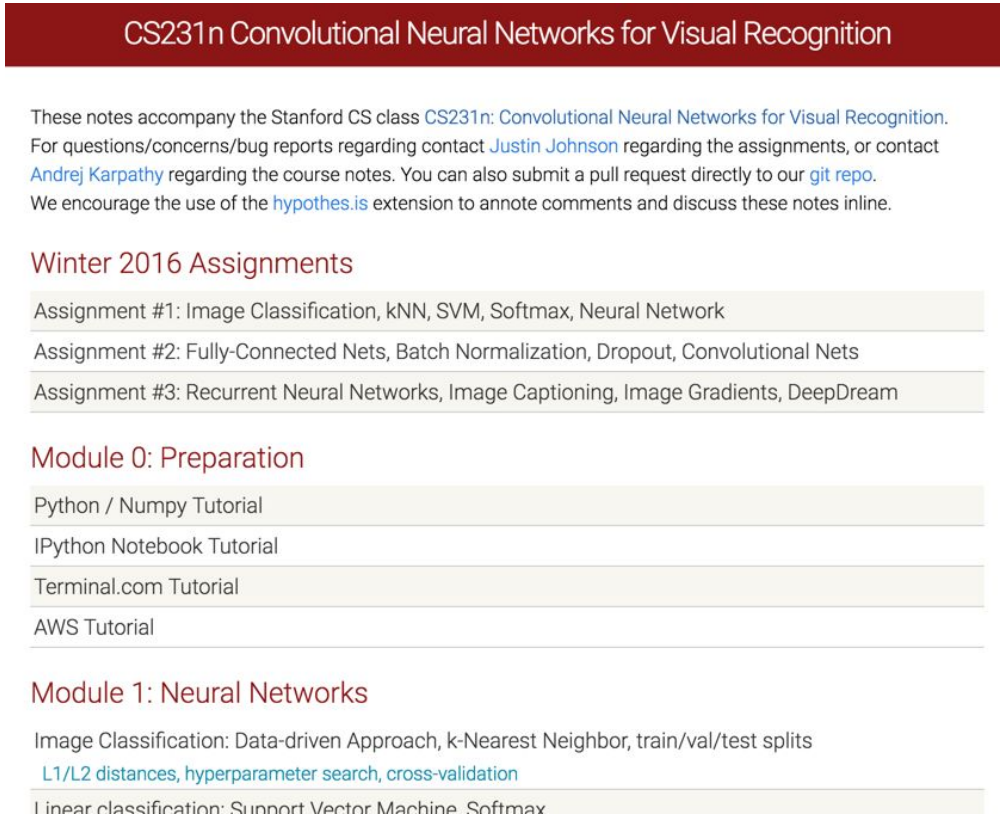
OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP *neural* NETWORK

Stanford's cs231n
Convolutional Neural Networks for Visual Recognition



CS231n Convolutional Neural Networks for Visual Recognition

These notes accompany the Stanford CS class CS231n: Convolutional Neural Networks for Visual Recognition.
For questions/concerns/bug reports regarding contact Justin Johnson regarding the assignments, or contact Andrej Karpathy regarding the course notes. You can also submit a pull request directly to our git repo.
We encourage the use of the hypothes.is extension to annote comments and discuss these notes inline.

Winter 2016 Assignments

Assignment #1: Image Classification, kNN, SVM, Softmax, Neural Network

Assignment #2: Fully-Connected Nets, Batch Normalization, Dropout, Convolutional Nets

Assignment #3: Recurrent Neural Networks, Image Captioning, Image Gradients, DeepDream

Module 0: Preparation

Python / Numpy Tutorial

IPython Notebook Tutorial

Terminal.com Tutorial

AWS Tutorial

Module 1: Neural Networks

Image Classification: Data-driven Approach, k-Nearest Neighbor, train/val/test splits

L1/L2 distances, hyperparameter search, cross-validation

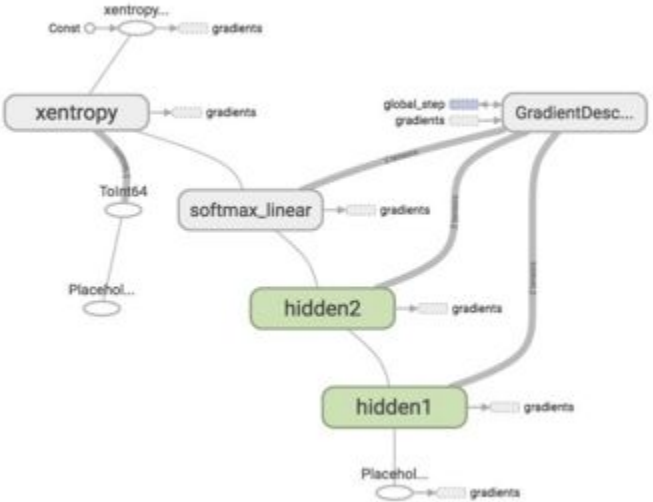
Linear classification: Support Vector Machine, Softmax

Lab #3 Comparing Classifers

goo.gl/nrdsxM

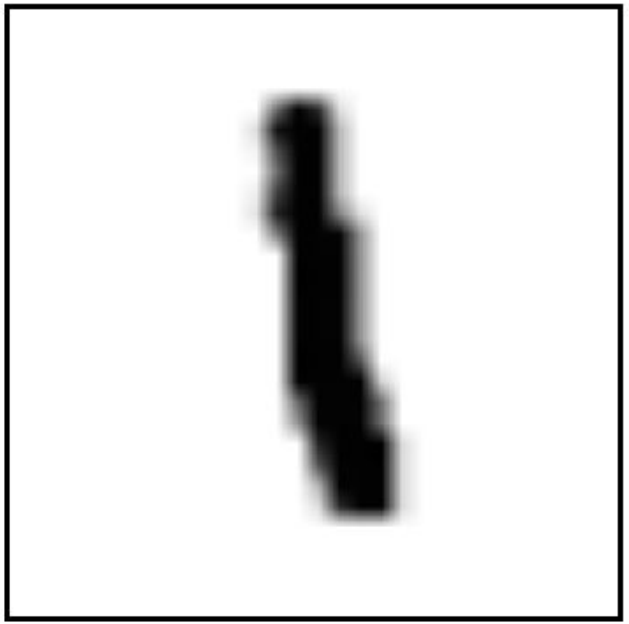
MNIST

What are we trying to do?



Objective: Given enough images and labels, figure out the weights and biases so the model can correctly identify digits.

What we see



What the computer “sees”

$$\approx$$

$$\begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .6 & .8 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .7 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & .5 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .4 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & .7 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .9 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & .3 & 1 & .1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{bmatrix}$$

3_mnist.ipynb

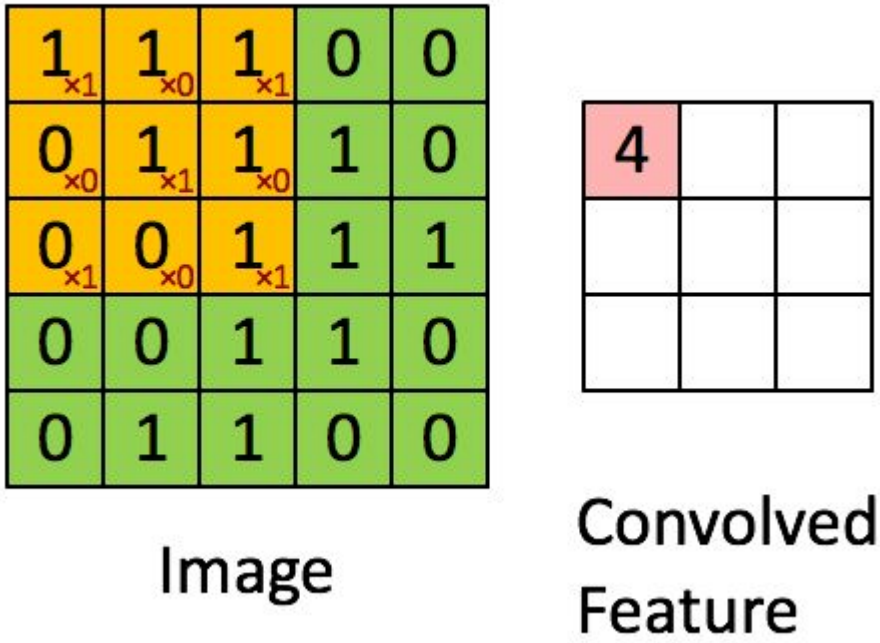
1. Import libraries (cell 2.1)
2. Define a set of constants for convenience (cell 2.2)
3. Get input data (cell 2.3)
4. Build inference graph (cell 2.4)
5. Build training graph (cell 2.5)
6. Build the complete graph for training, and saving checkpoints (cell 2.6)
7. Run training for MAX_STEPS and save checkpoint at the end (cell 2.7)
8. Run evaluation based on the saved checkpoint (cell 2.8)

Exercises to run

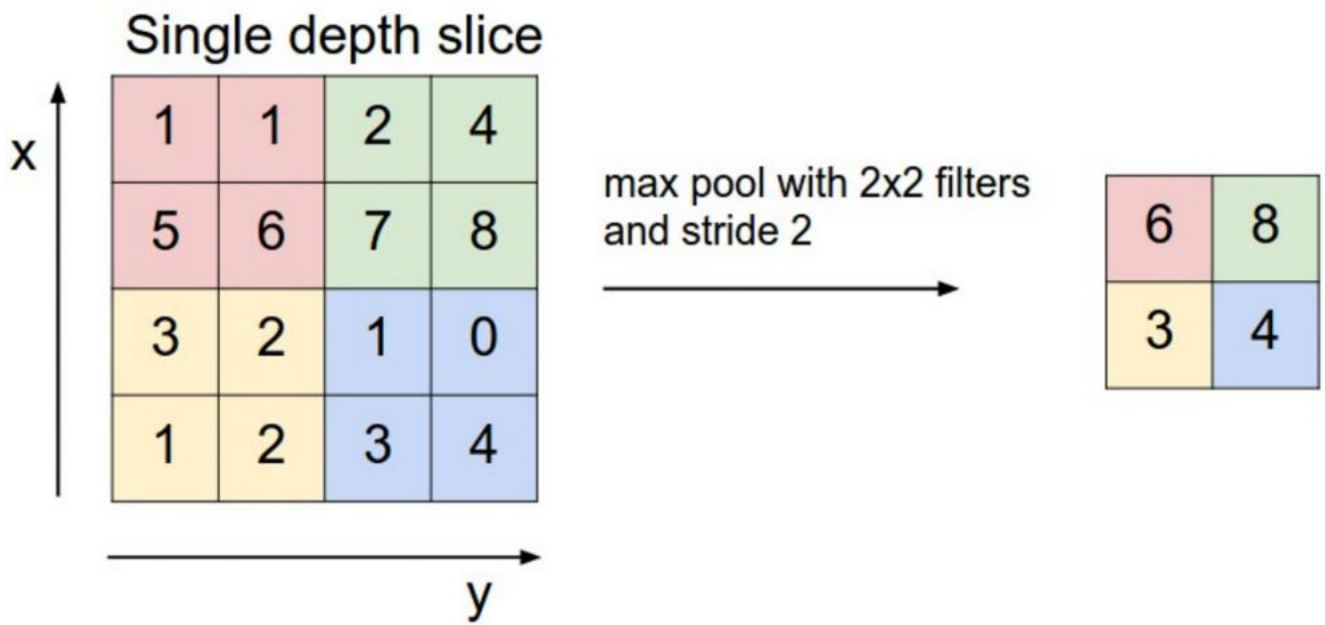
1. Run all the cells
2. Uncomment the graph writing to see what the graph looks like.
3. Print out loss value every 100 steps. What do you notice?
4. Save checkpoints every 500 steps.
5. Run evaluation with different checkpoints. What do you notice?
6. Run evaluation on the complete validation set.
7. Build evaluation graph from scratch instead of importing from meta graph.
8. Experiment with different optimizers

Lab #4 Deep MNIST
goo.gl/nrdsxM

Convolutional Neural Networks



Convolution with 3×3 Filter. Source:
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

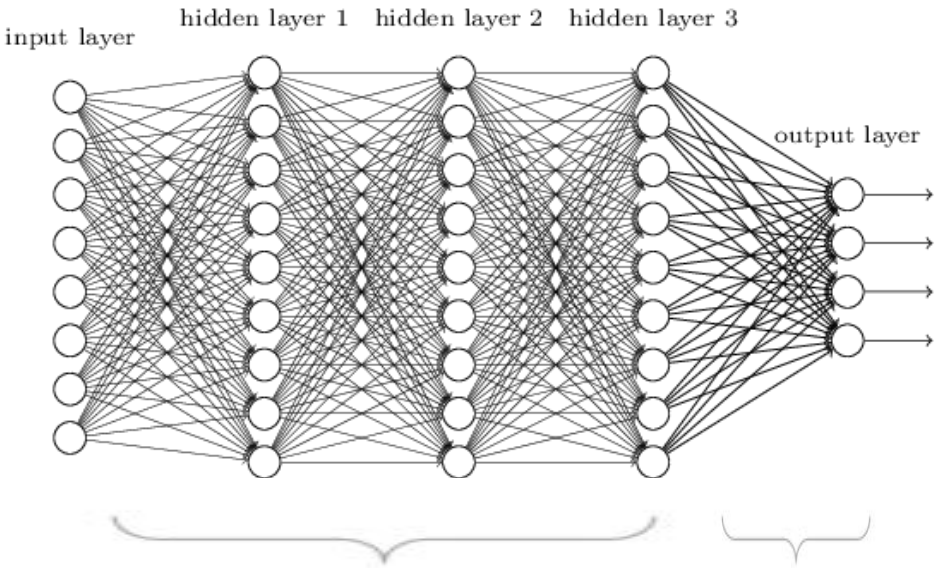


Max pooling in CNN. Source: http://cs231n.github.io/convolutional-networks/#pool, via http://www.wildml.com/2015/11/understanding-convolutional-neural-networks-for-nlp/

Lab #5
Convolutional MNIST at TensorFlow.org

Image Retraining

Image Retraining

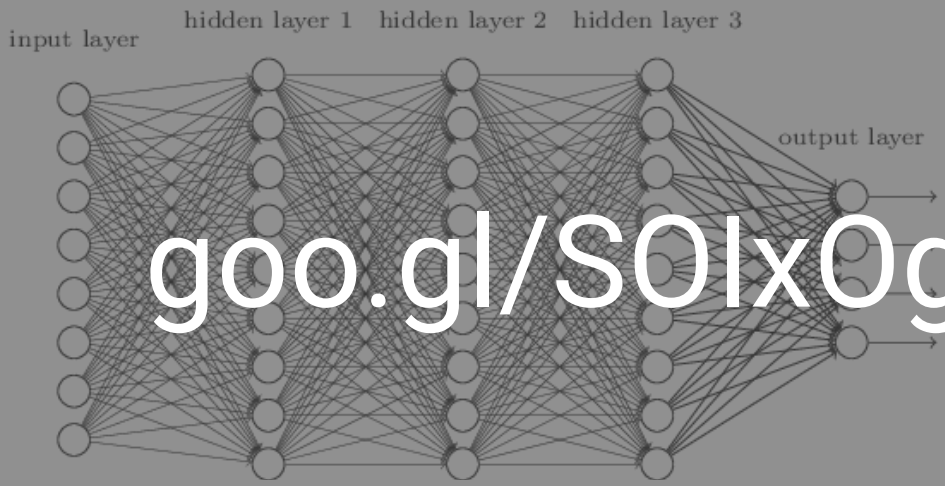


Most of the network is unchanged

Only the last layer of weights and the outputs are updated

Image: neuralnetworksanddeeplearning.com

Image Retraining

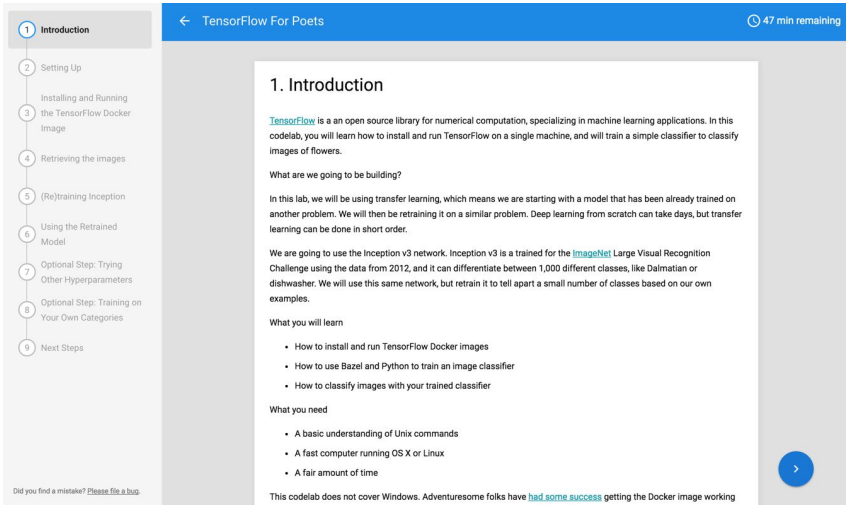


Most of the network is unchanged

Only the last layer of weights and the outputs are updated

Image: neuralnetworksanddeeplearning.com

TensorFlow for Poets

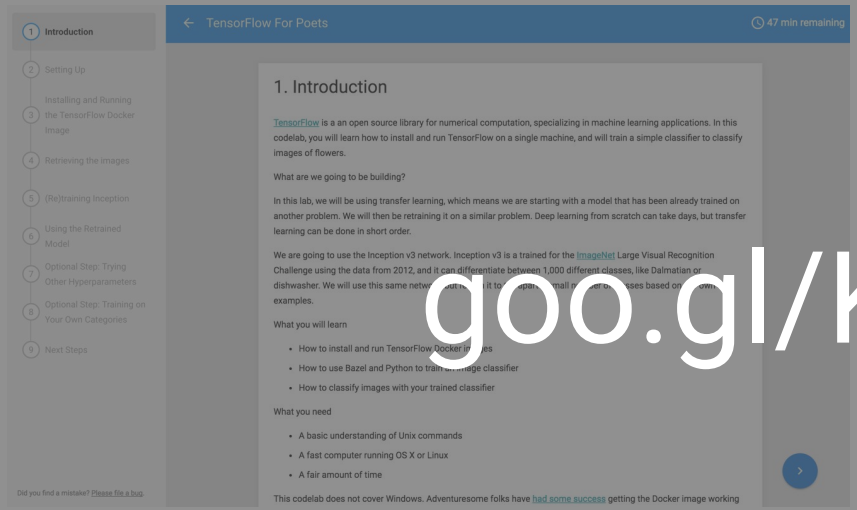


Codelab
goo.gl/xGsB9d

Video
Machine Learning Recipes #6

Pete Warden - https://twitter.com/petewarden

TensorFlow for Poets



goo.gl/KewA03

Codelab
goo.gl/xGsB9d

Video
[Machine Learning Recipes #6]

Pete Warden - https://twitter.com/petewarden

Mobile TensorFlow

Mobile TensorFlow

https://www.tensorflow.org/mobile.html

TensorFlow™ GET STARTED TUTORIALS HOW TO MOBILE API RESOURCES ABOUT

Mobile TensorFlow

TensorFlow was designed with mobile and embedded platforms in mind. We have sample code and build support you can try now for these platforms:

Android

iOS

Raspberry Pi

Many applications can benefit from on-device processing. Google Translate's instant visual translation is a great example. By running its processing locally, users get an incredibly responsive and interactive experience.

Mobile TensorFlow makes sense when there is a poor or missing network connection, or where sending continuous data to a server would be too expensive. We are working to help developers make lean mobile apps using TensorFlow, both by continuing to reduce the code footprint, and supporting quantization and lower precision arithmetic that reduce model size.

Claude Monet - Bouquet of Sunflowers
Images from the Metropolitan Museum of Art (with permission)

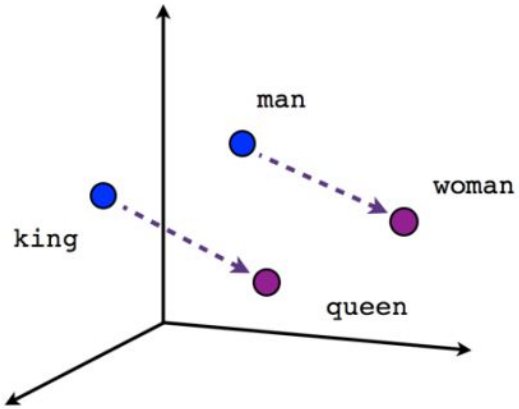
Image by @random_forests

Lab #6 TensorFlow for Poets

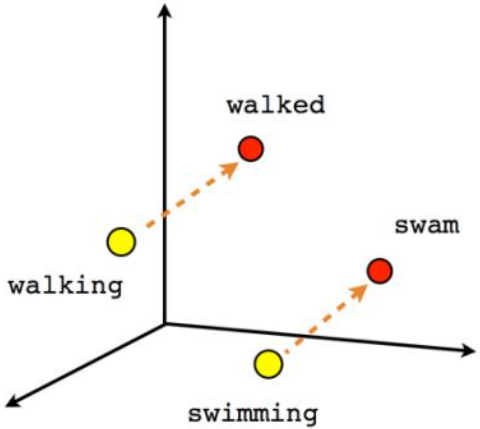
Word Embeddings

Nearest to government:

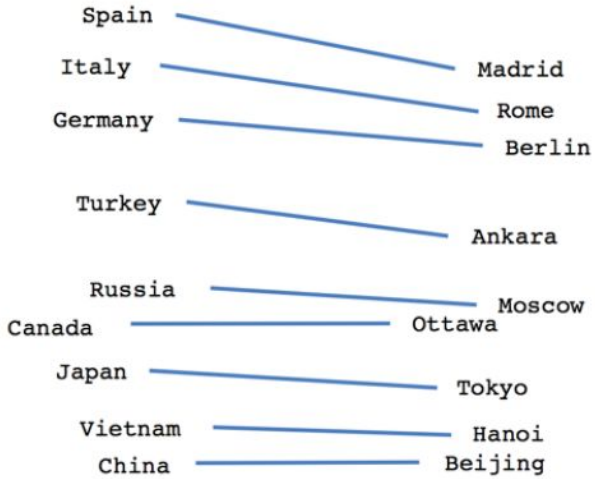
'governments', 'leadership',
'regime', 'crown', 'rule',
'leaders', 'parliament',
'elections', ...



Male-Female



Verb tense



Country-Capital

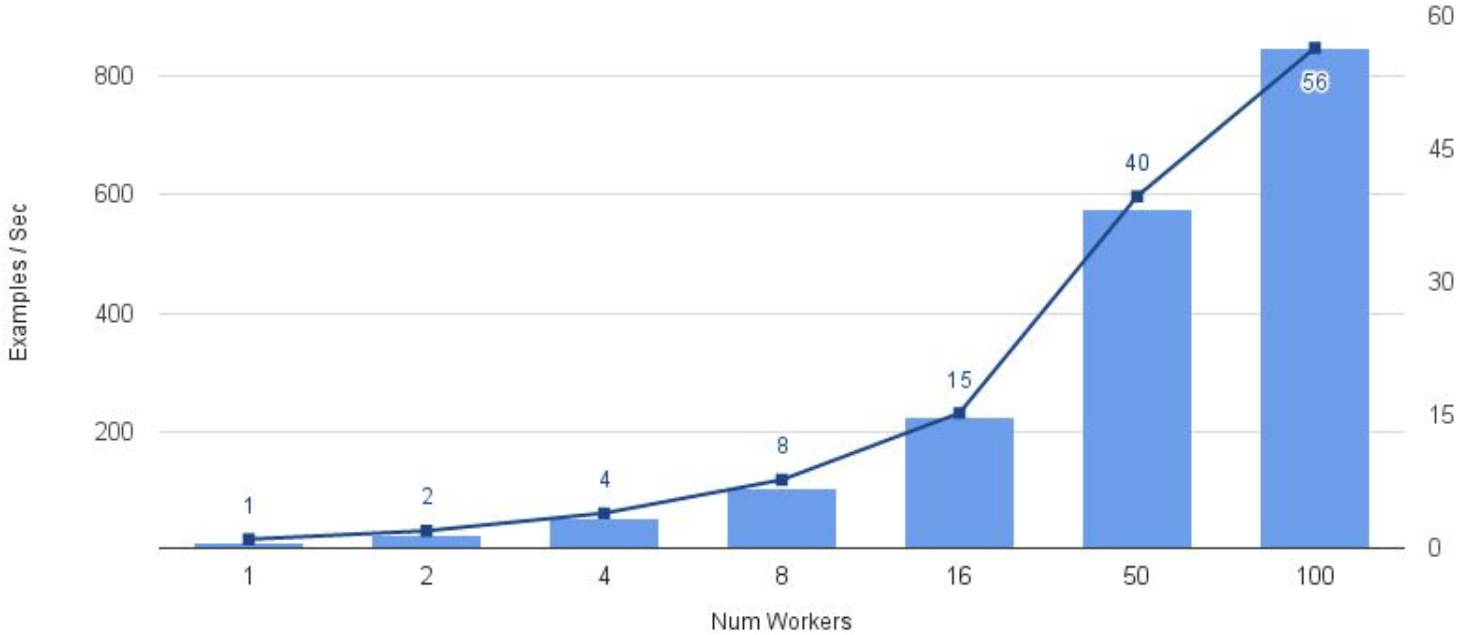
Lab #7 Word embeddings
goo.gl/nrdsxM
Also check tensorflow.org for analogies

Lab #8 Document classification

https://github.com/tensorflow/tensorflow/blob/master/tensorflow/examples/skflow/text_classification.py

Distributed TensorFlow

TensorFlow



https://arxiv.org/abs/1604.00981

Image Model (Inception) Synchronous Training

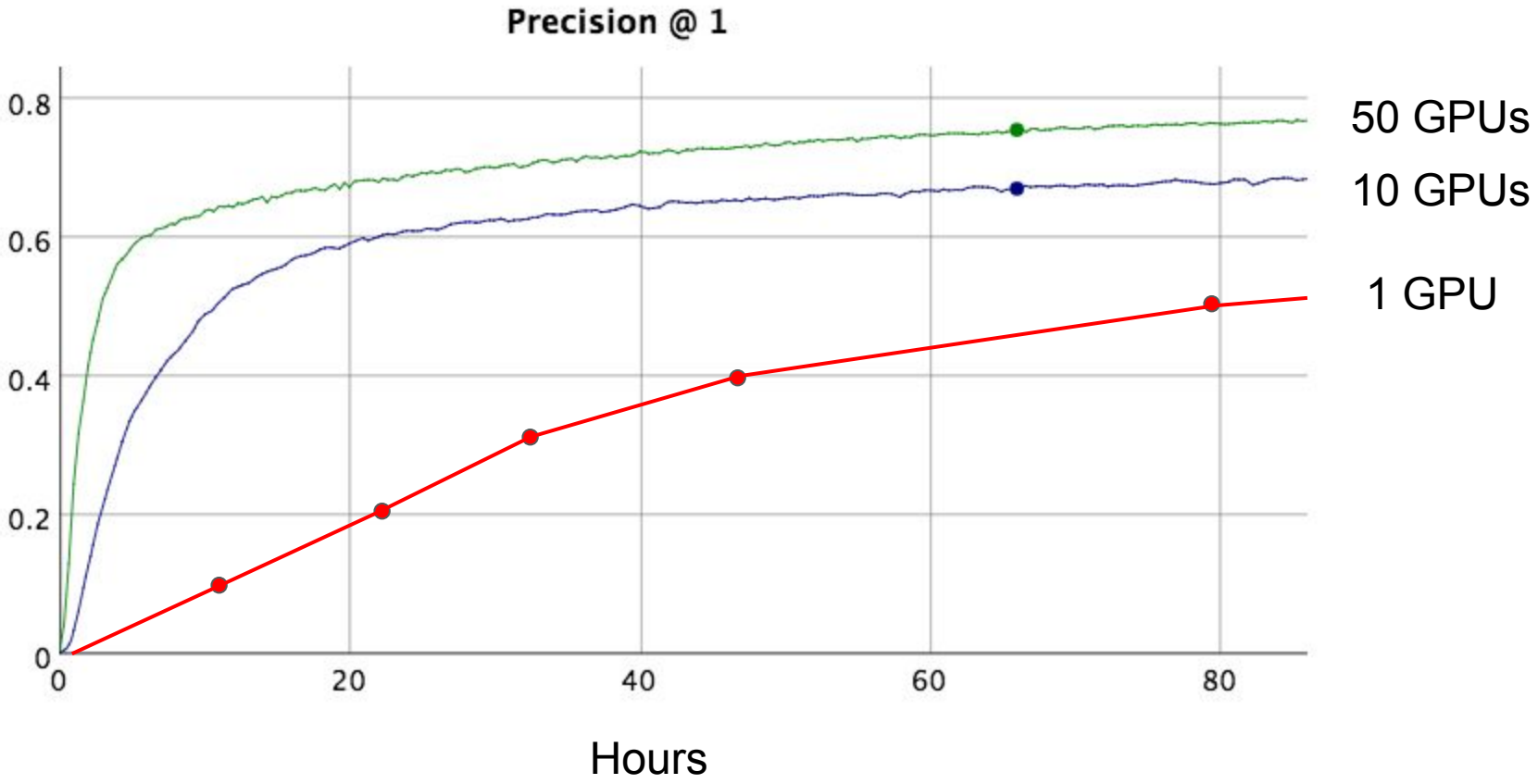


Image Model (Inception) Synchronous Training

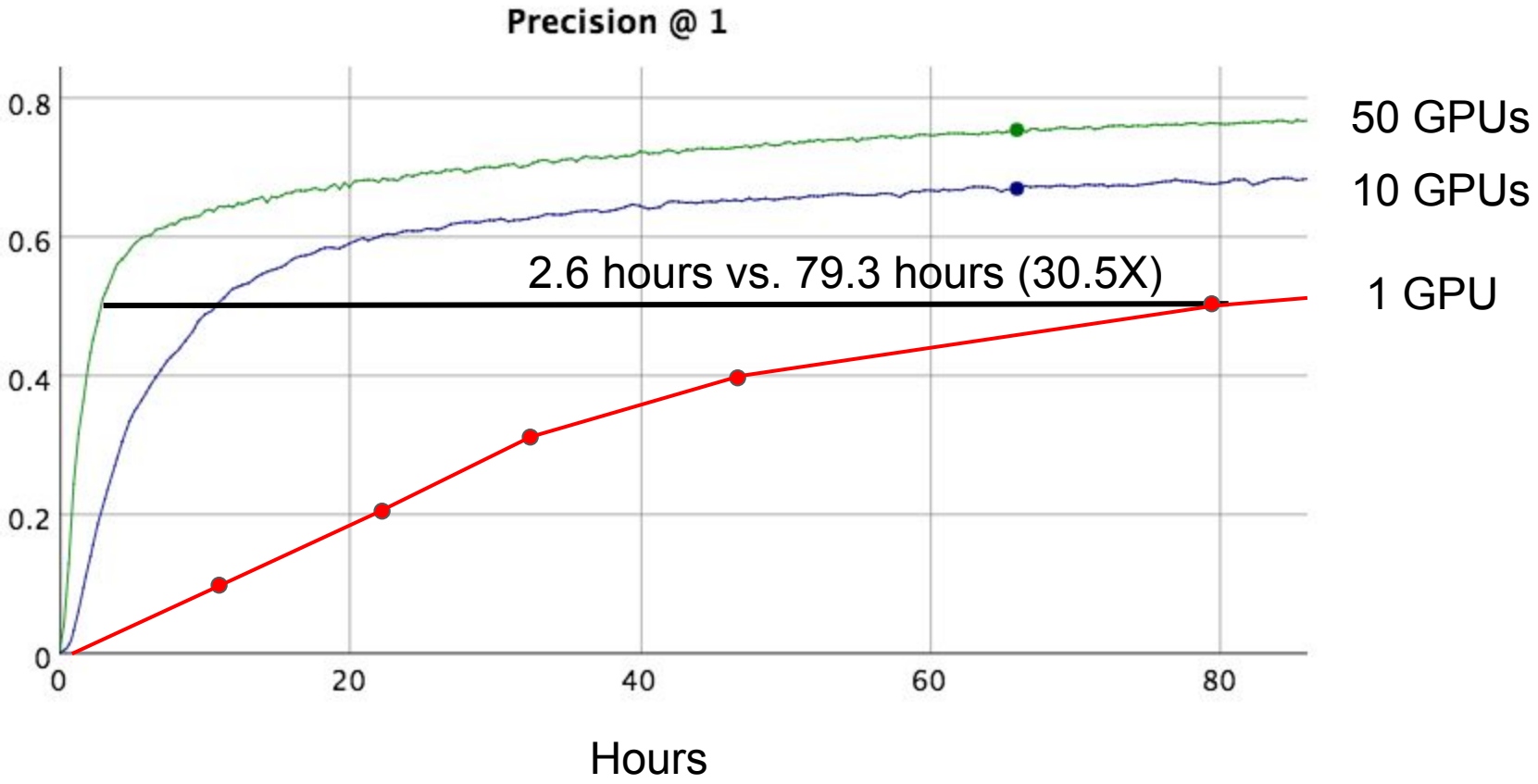
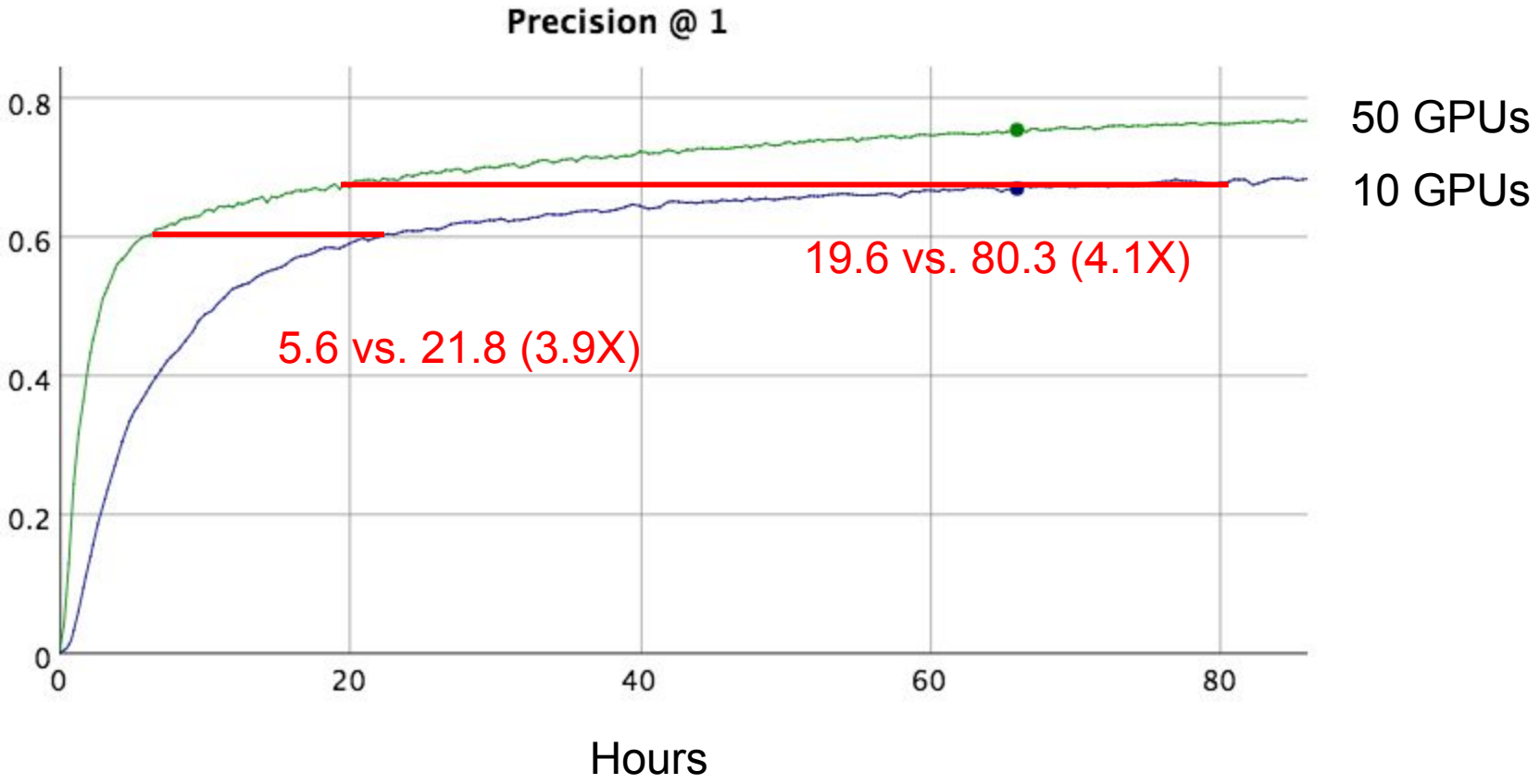


Image Model (Inception) Synchronous Training



Google Cloud Machine Learning Platform LIMITED PREVIEW

- Training and prediction at Scale
- Fully Managed Platform
- Powered by TensorFlow
- Seamless integration with Google Cloud Platform

Bottomline: TensorFlow is for Machine Learning -- from lab to production.

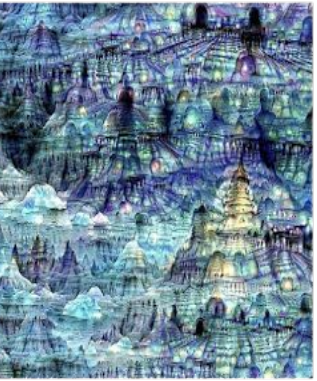
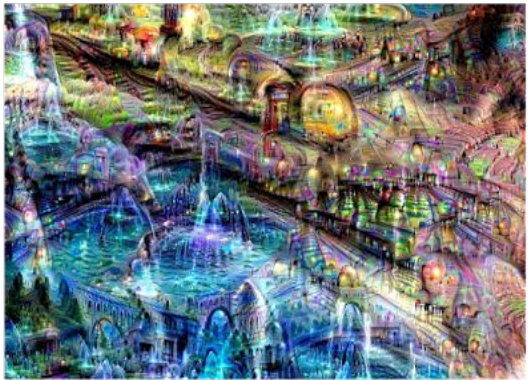
Fun experiments

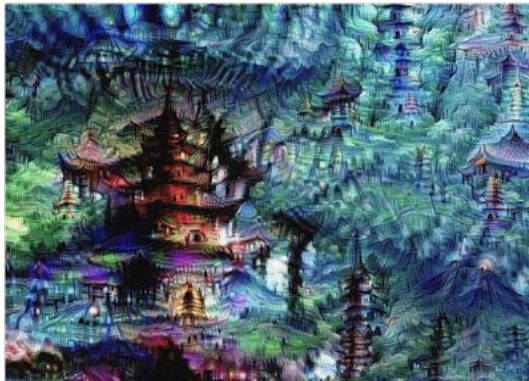
Image: Leonid Afremov – Rain Princess

Image: Stephen Wilkes, National Geographic


goo.gl/fyDxhC






goo.gl/OCYseb


goo.gl/67hfSo

Fun experiments

Deep Dream: goo.gl/OCYseb

Style transfer: goo.gl/fyDxhC

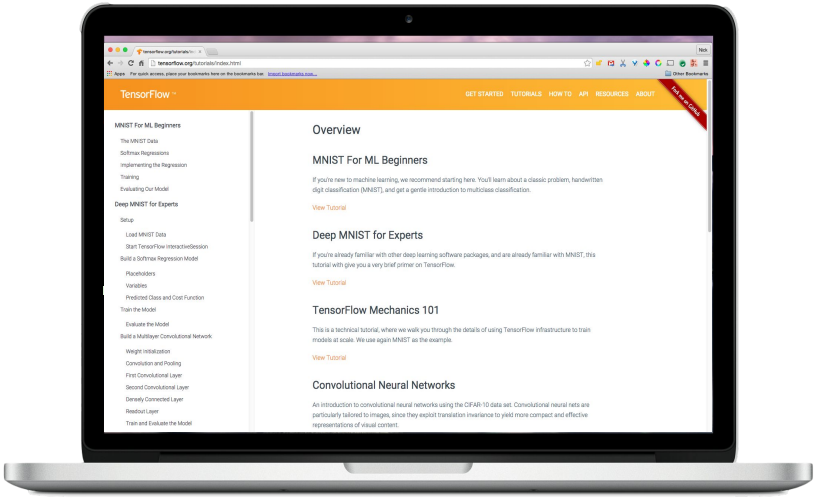
Image completion: goo.gl/67hfSo

Next steps

Tutorials and code
tensorflow.org

Want to learn more?
Udacity class goo.gl/iHssII

Totally new to ML?
Recipes goo.gl/KewA03



Thanks and have fun!

Martin Wicke
@martin_wicke

Josh Gordon
@random_forests