

Intro to TensorFlow

Your guides



Josh Gordon
[@random_forests](https://twitter.com/random_forests)



Wolff Dobson
[@greenexecutive](https://twitter.com/greenexecutive)

Welcome & Logistics

- Thanks for coming!
- Today we'll get started with TensorFlow
- Format is a mix of talks and Jupyter notebooks
- Content for beginners and experienced developers

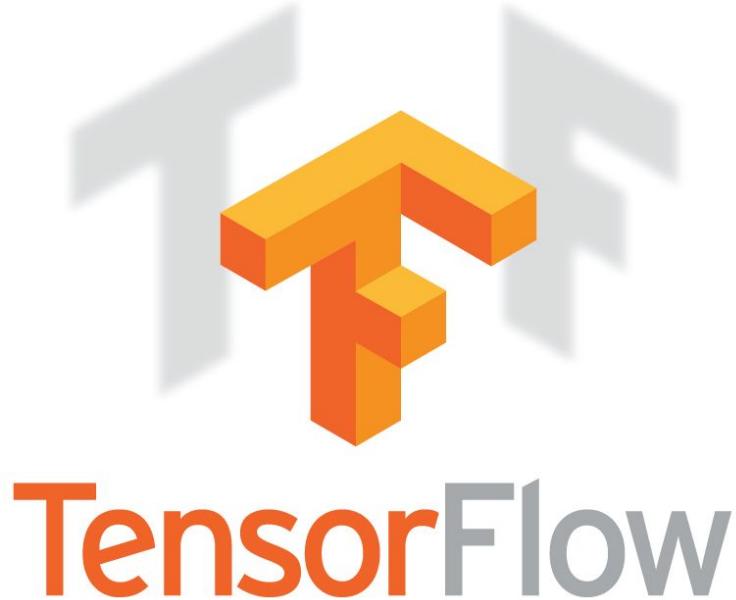
These slides + code
goo.gl/nrdsxM

What can we do better?
goo.gl/8ReQxN

Outline

- **Pretrained models**
 - Inception & TensorFlow for Poets
- **Higher-level APIs**
 - mnist with tf.contrib.learn
- **Lower-level APIs**
 - Fibonacci sequence, Linear Regression, and Deep Neural Networks
- **Bonus material**
 - Deep Dream, Neural Style

Getting Started

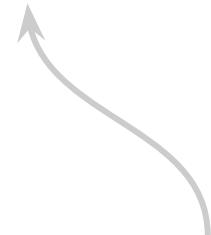


- Fast, flexible, and scalable open-source machine learning library
- One system for research and production
- Runs on CPU, GPU, TPU, and Mobile
- Apache 2.0 license

A multidimensional array.



TensorFlow



A graph of operations.

Deep Learning

Current state of the art in:

- **Image:** *recognition, captioning, enhancement, completion*
- **Language:** *translation, parsing, summarization*
- **Speech:** *recognition, generation*
- **Games:** *AlphaGo, Atari*
- And much more.



Open Source Models

github.com/tensorflow/models

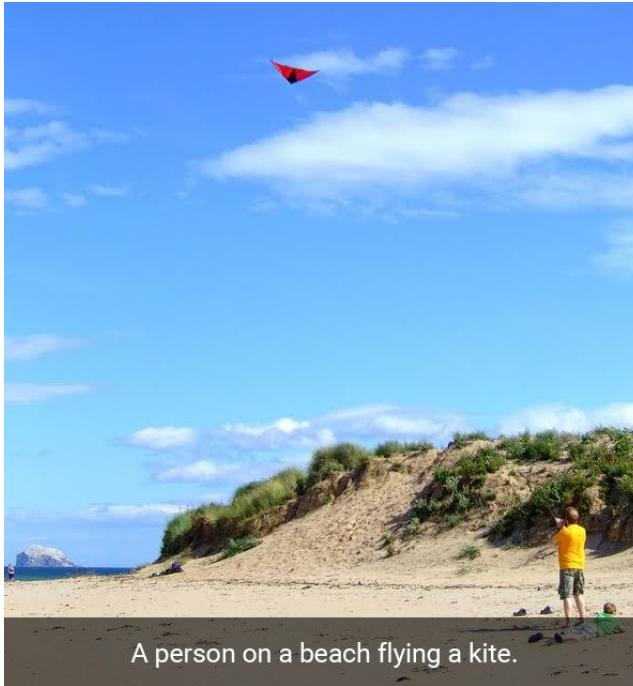
Inception



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

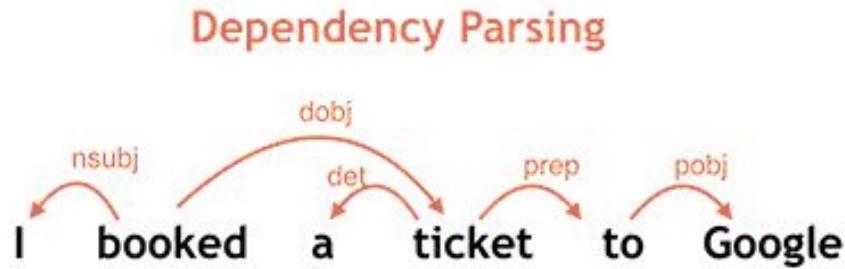
<https://research.googleblog.com/2016/08/improving-inception-and-image.html>

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

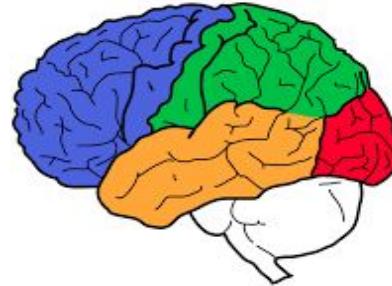
- *Alice and Bob took the train to visit the zoo. They saw a **baby giraffe**, a **lion**, and a **flock of colorful tropical birds**.*

Abstractive summary

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

“Universal” Machine Learning

*Speech
Text
Search
Queries
Images
Videos
Labels
Entities
Words
Audio
Features*



*Speech
Text
Search
Queries
Images
Videos
Labels
Entities
Words
Audio
Features*

Install TensorFlow 0_test_install.ipynb

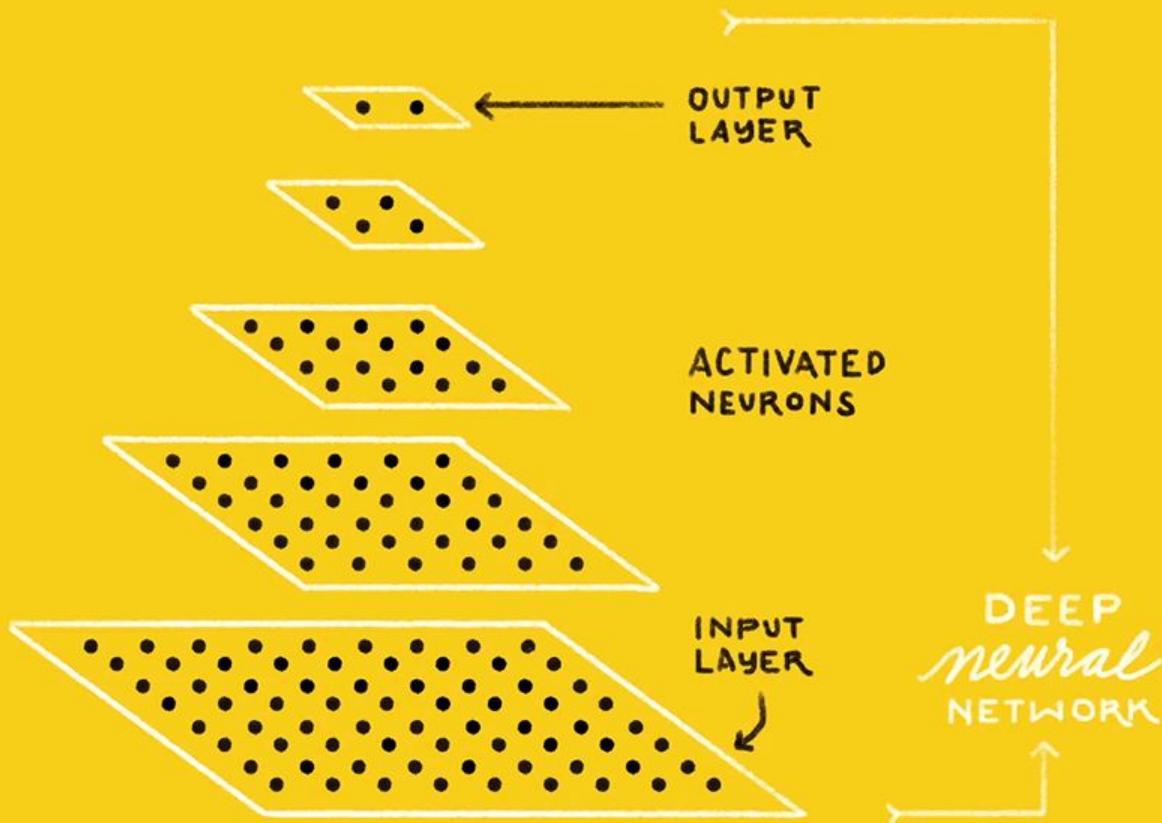
goo.gl/nrdsxM

Demo: working with Jupyter notebooks

IS THIS A
CAT or DOG?



CAT DOG



Demo: TensorFlow Playground

playground.tensorflow.org

Stanford's CS231n

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 annotate 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](#)

[parameteric approach, bias trick, hinge loss, cross-entropy loss, L2 regularization, web demo](#)

[Optimization: Stochastic Gradient Descent](#)

[optimization landscapes, local search, learning rate, analytic/numerical gradient](#)

[Backpropagation Intuitions](#)

Neural Networks in TensorFlow

Options

1. Use a pre-trained model
2. Use a higher-level API that builds on top of TensorFlow
3. Define and train your network directly in TensorFlow



More control,
more effort

1) Pretrained Models

Remember Inception?

- Load the graph and learned weights
- Use the model



Images via [Wikipedia](#)

www.tensorflow.org/tutorials/image_recognition/index.html

Demo: classifying images with Inception

Inception

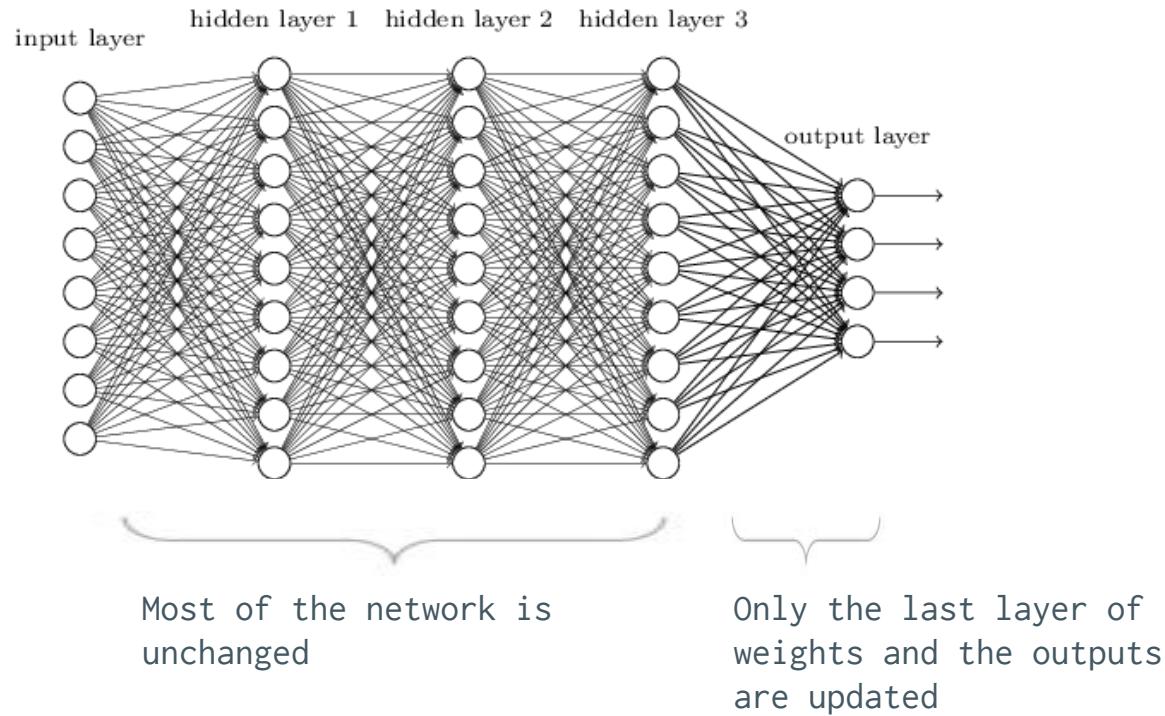
- 1,000 categories: pandas, cats, bears
- Performs about as well as people on this task
- Training took ~2 weeks on a single machine
- Open sourced **implementation** (so you can train on your own data) and a **pretrained model** (so you can experiment right away)

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

But...

- Animals are great.
- What if you want to train an image classifier on your own data?

Transfer Learning



Advantages

- Minutes vs weeks
- Handful of training data
- Reuse learned features
- Two ways to do this
 - a. Directly in TensorFlow ([tutorial](#))
 - b. TensorFlow for Poets

TensorFlow for Poets

The screenshot shows the TensorFlow For Poets codelab interface. On the left, a vertical navigation bar lists steps from 1 to 9: Introduction, Setting Up, Installing and Running the TensorFlow Docker Image, Retrieving the images, (Re)training Inception, Using the Retrained Model, Optional Step: Trying Other Hyperparameters, Optional Step: Training on Your Own Categories, and Next Steps. Step 1 is highlighted. The main content area is titled "1. Introduction". It contains text about TensorFlow being an open source library for numerical computation, specializing in machine learning applications. It mentions that in this codelab, you will learn how to install and run TensorFlow on a single machine, and will train a simple classifier to classify images of flowers. Below this, there's a section titled "What are we going to be building?" which describes using transfer learning to retrain an Inception v3 network on a dataset of flower images. It also lists what you will learn and what you need.

Codelab - goo.gl/xGsB9d



Video - goo.gl/KewA03

Credit: Pete Warden - thank you!

Monet - Sunflowers: 0.992



Claude Monet - Bouquet of Sunflowers

Images from the Metropolitan Museum of Art (with permission)

2) Higher-level APIs

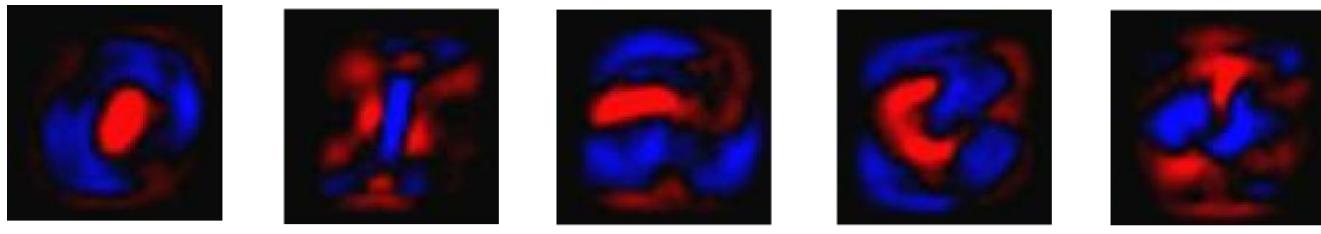
2) Higher-level APIs

- tf.contrib.learn
 - a. Implements a sklearn-inspired API
 - b. Creates the graph for you behind the scenes

MNIST: 3_mnist_high_level.ipynb
goo.gl/nrdsxM

“The only benchmark that matters”

goo.gl/nrdsxM



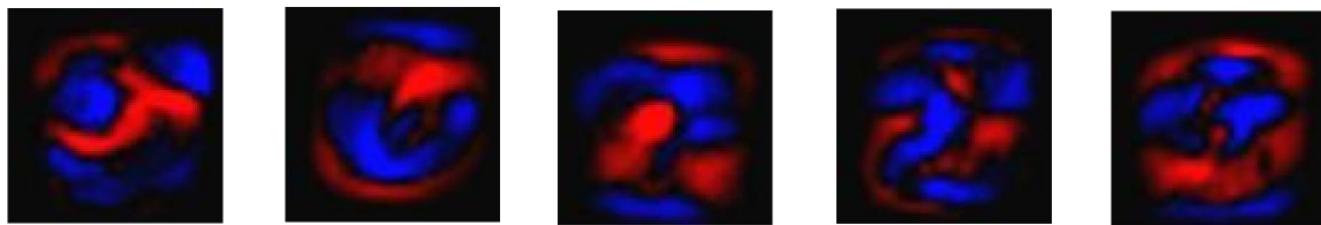
0

1

2

3

4



5

6

7

8

9

Want to learn more?

tf.contrib.learn tutorials

- [Quickstart](#)
- [Linear models](#)
- [Wide and Deep](#)

3) Lower-level APIs

Under the hood

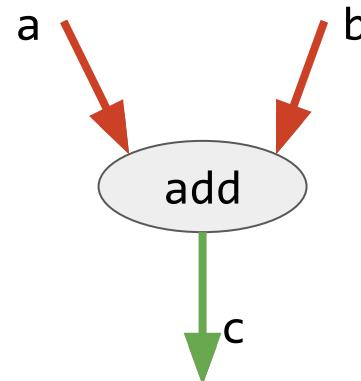
Part 1: Graphs and Sessions

Build a graph; then run it.

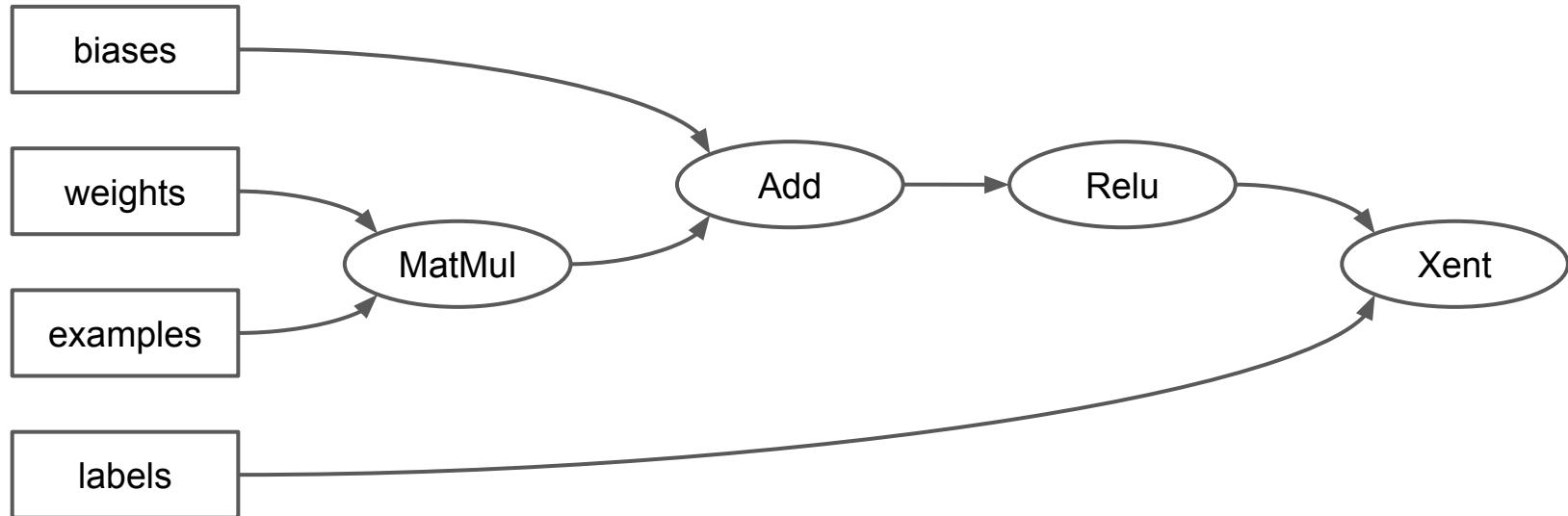
```
...  
c = tf.add(a, b)
```

```
...
```

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

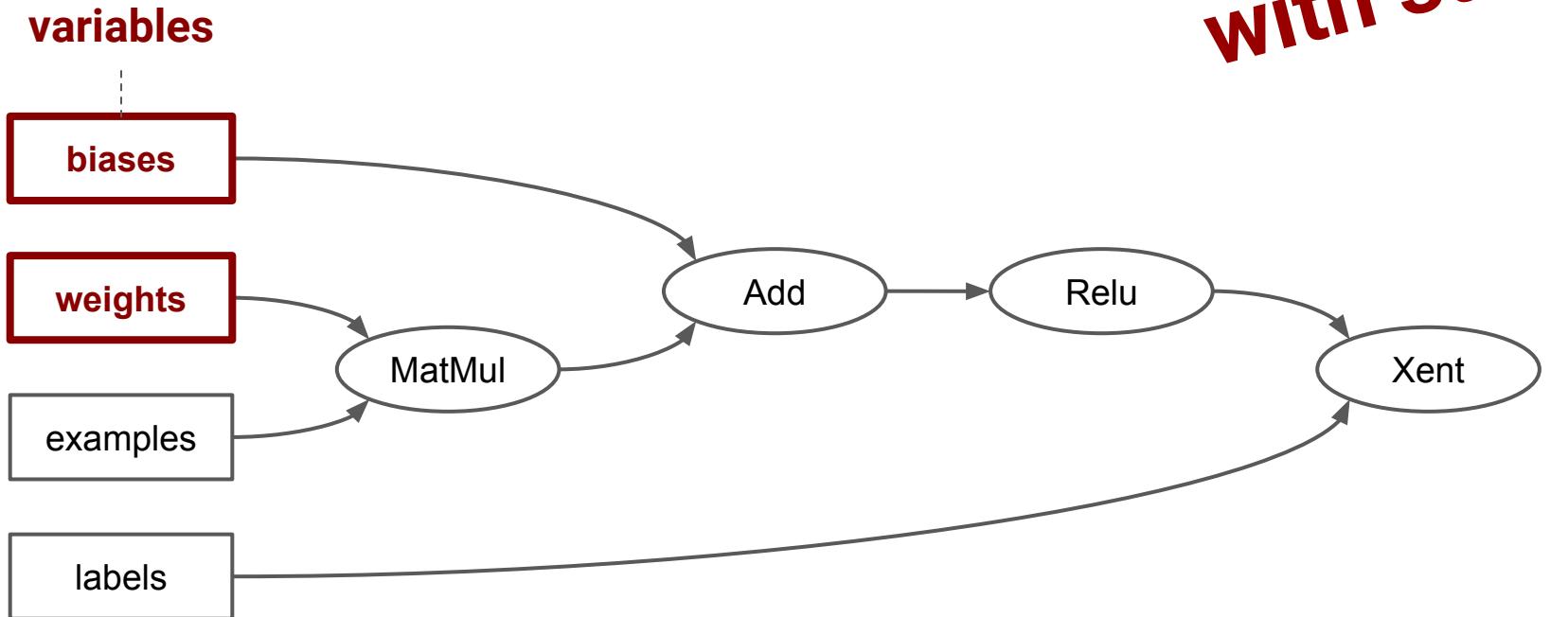


Any Computation is a TensorFlow Graph



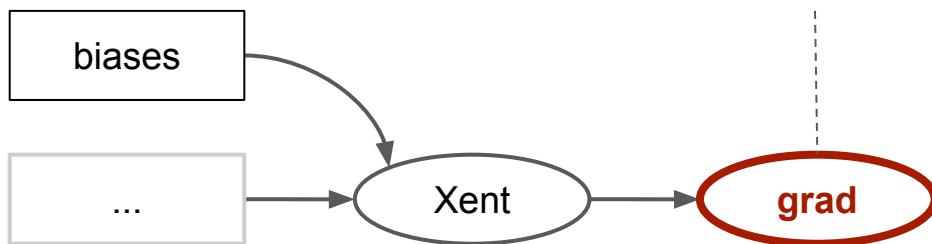
Any Computation is a TensorFlow Graph

with state



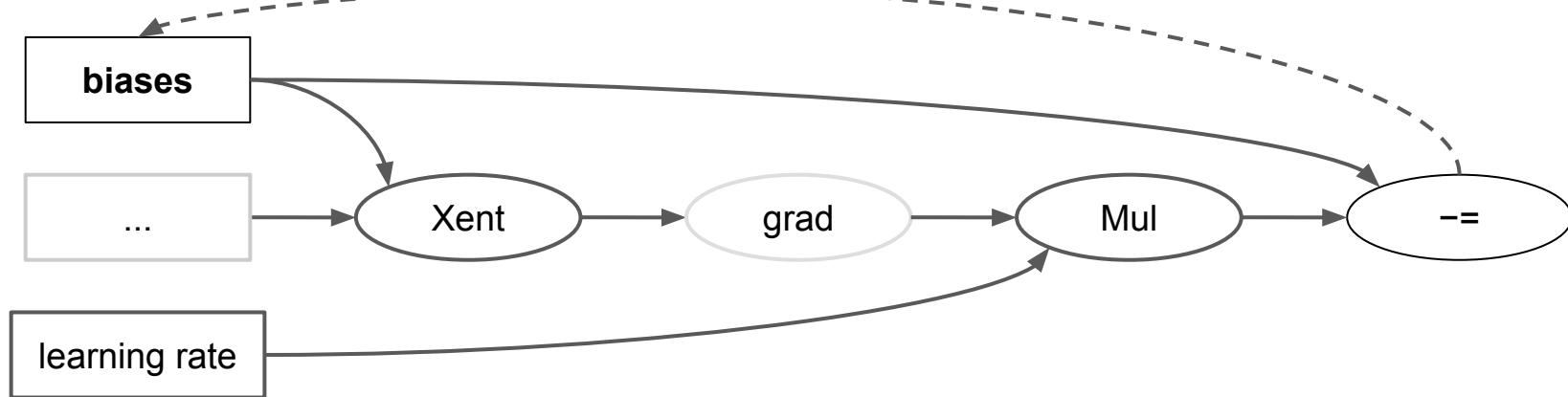
Automatic Differentiation

Automatically add ops which
compute gradients for variables

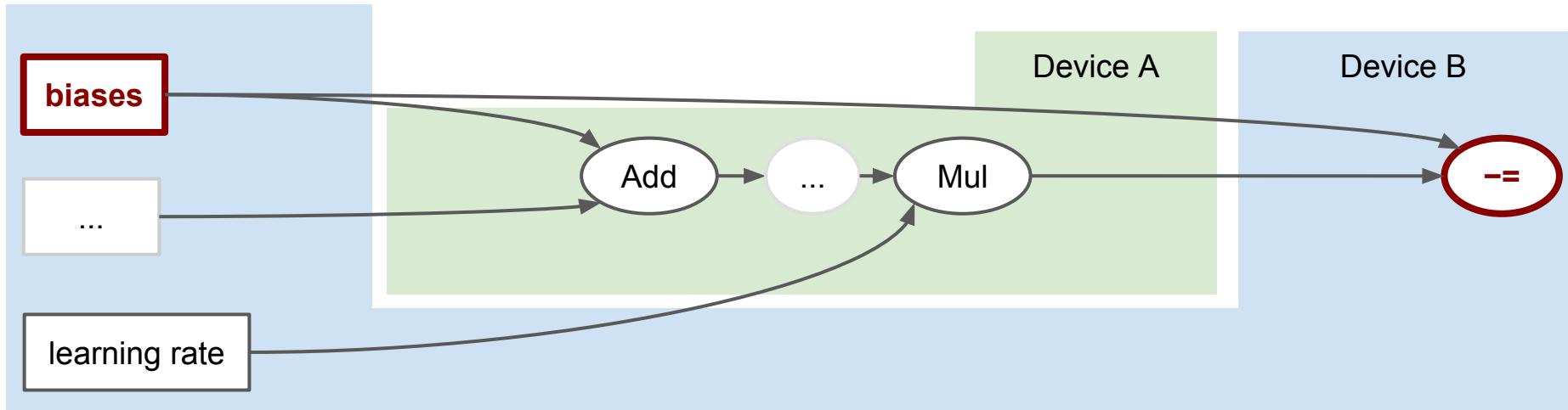


Automatic Differentiation

Simple gradient descent:

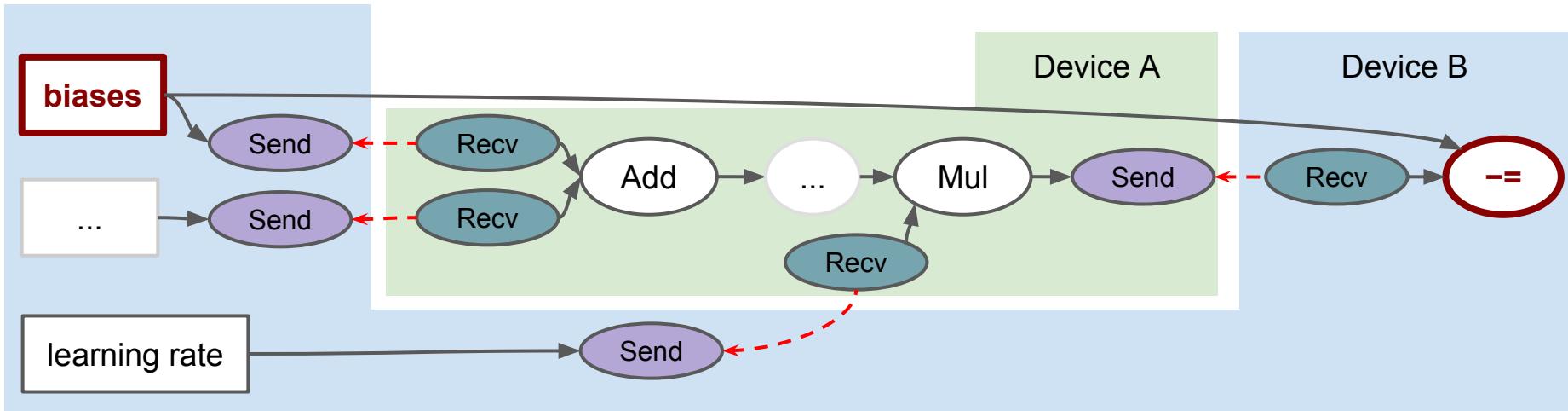


Graphs can be distributed and assigned to devices



Devices: Processes, Machines, CPUs, GPUs, TPUs, etc

Send and Receive Nodes



Devices: Processes, Machines, CPUs, GPUs, TPUs, etc

Fibonacci Sequence: 1_warm_up.ipynb
goo.gl/nrdsxM

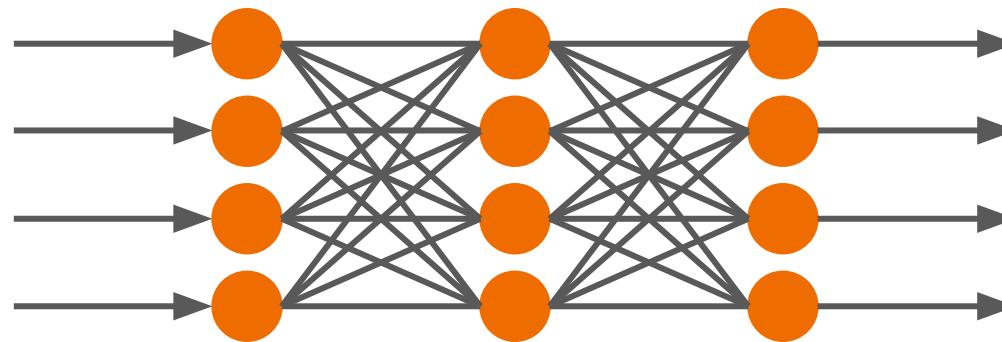
Or, if you prefer:

TensorFlow for Poets
goo.gl/xGsB9d

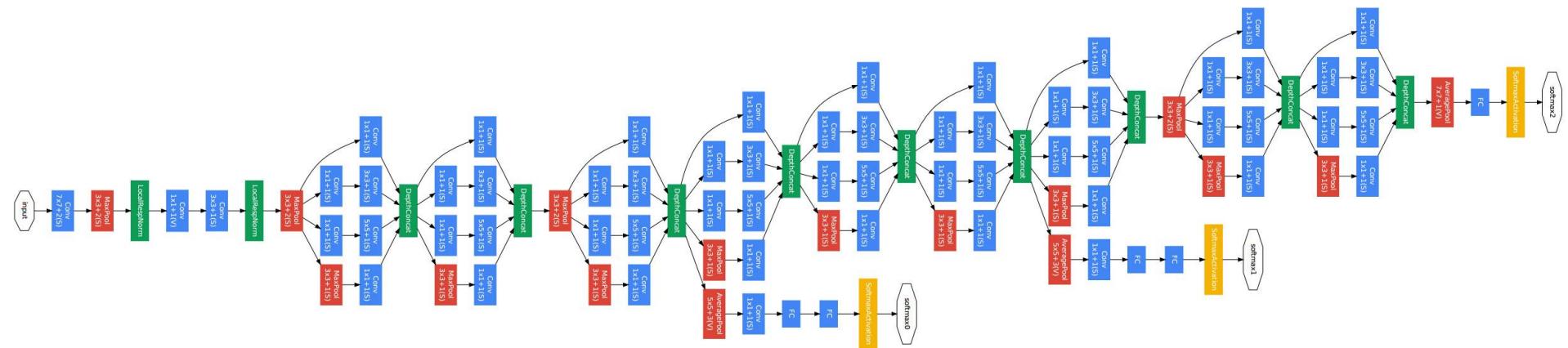
The background features a stylized landscape graphic composed of a grid of white lines on a yellow-orange gradient background. The grid forms rolling hills and mountains, creating a sense of depth and perspective.

But why?

Neural Networks, yesterday



Inception (2015)



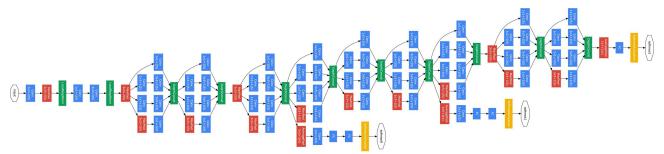
Machine Learning gets complex quickly



Heterogenous
systems

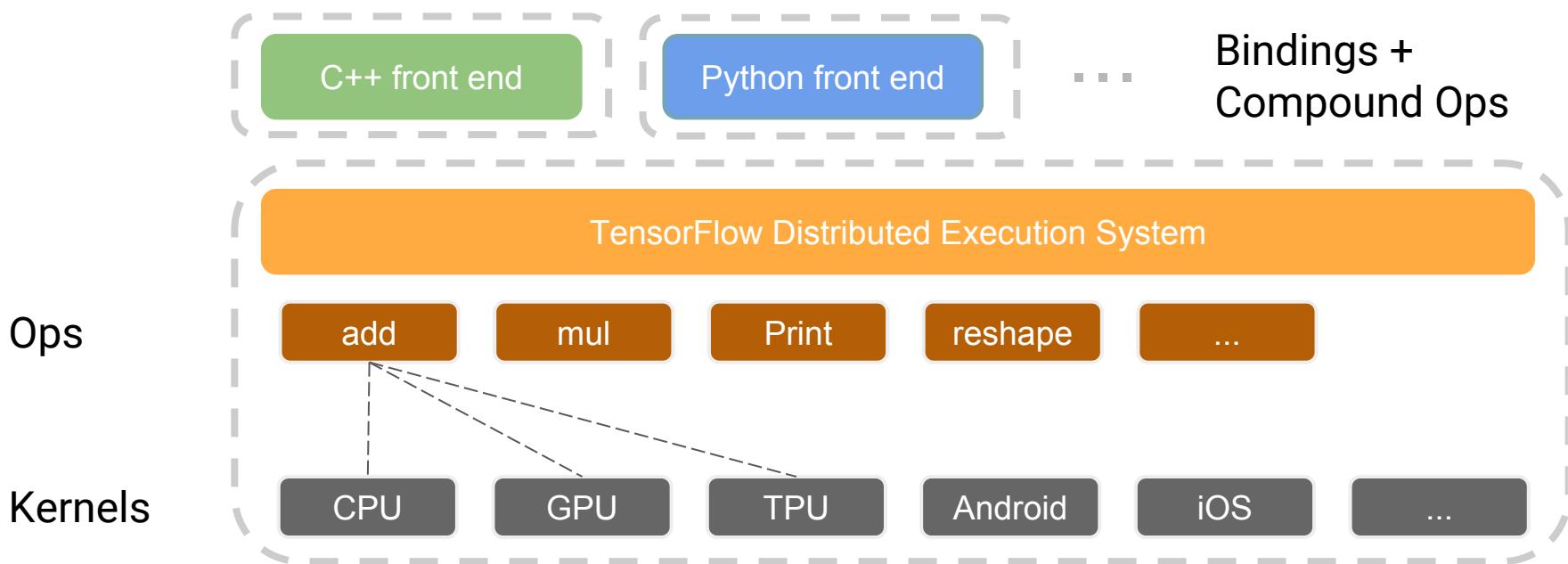


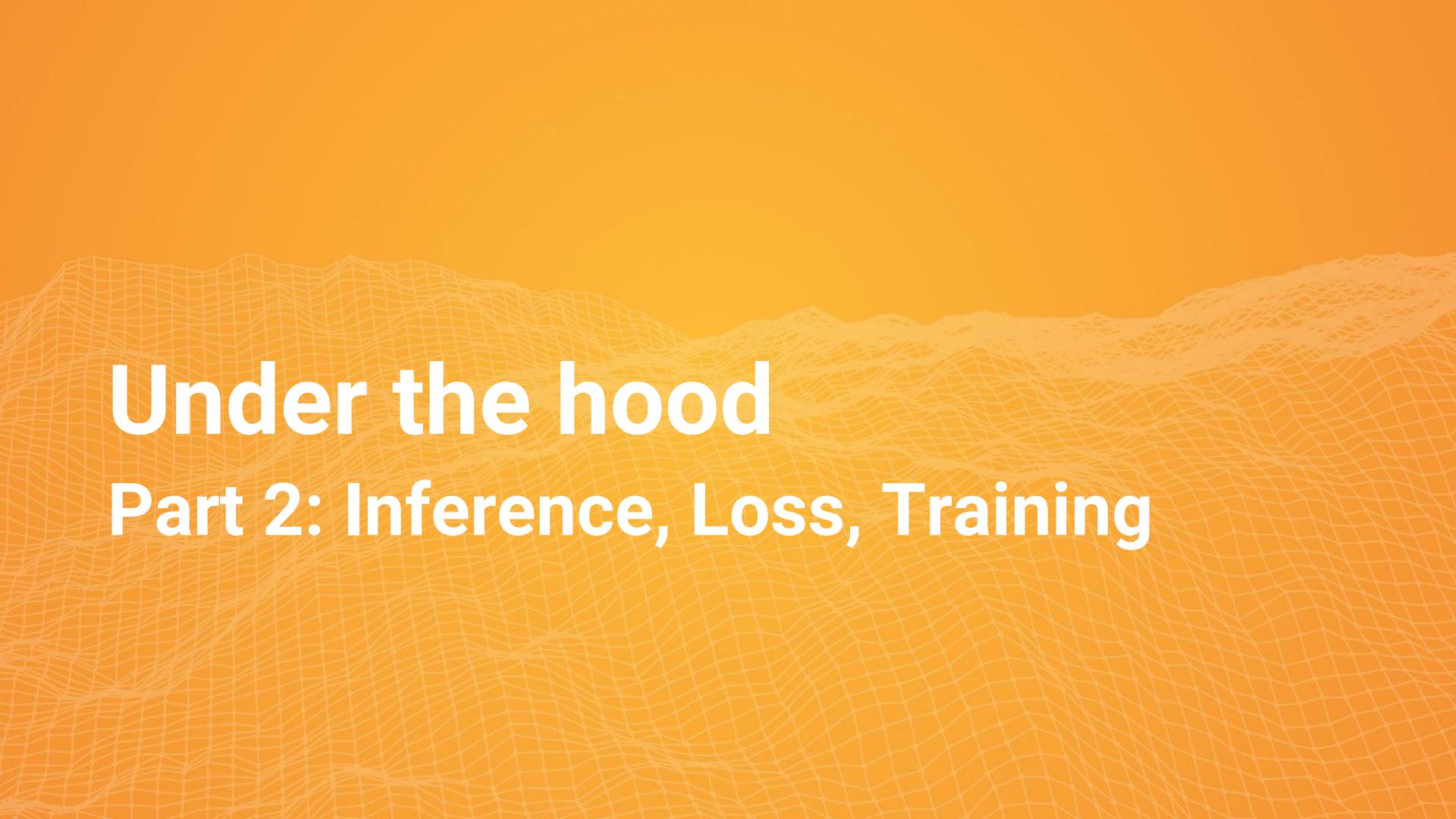
Distributed
systems



Modeling complexity

Architecture



The background of the slide features a warm orange gradient. Overlaid on this are several white wireframe mesh representations of 3D surfaces, resembling undulating hills or ripples in space-time.

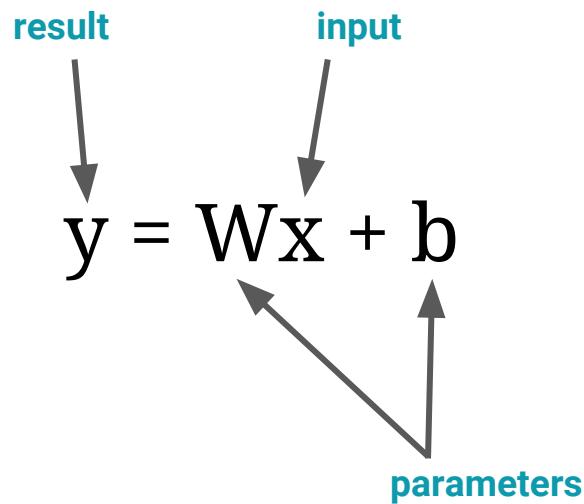
Under the hood

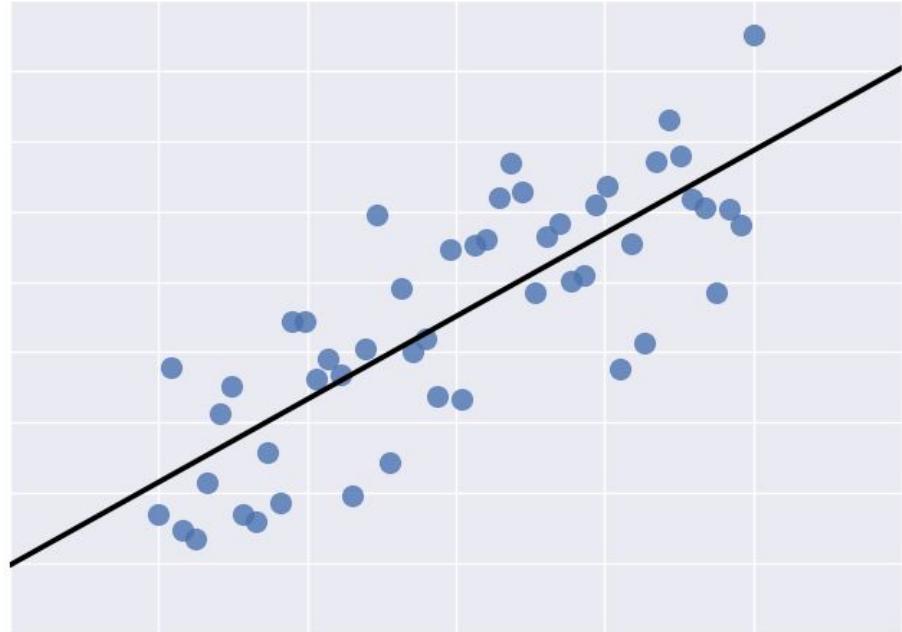
Part 2: Inference, Loss, Training

Linear Regression

$$y = Wx + b$$

result input
parameters





What are we trying to do?

Mystery equation: $y = 0.1 * x + 0.3 + \text{noise}$

Model: $y = W * x + b$

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

$y = Wx + b$ in TensorFlow

```
import tensorflow as tf
```

$y = Wx + b$ in TensorFlow

```
import tensorflow as tf

x = tf.placeholder(shape=[None],  
                   dtype=tf.float32, name="x")
```

$y = Wx + b$ in TensorFlow

```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                    dtype=tf.float32, name="x")

W = tf.get_variable(shape=[], name="W")
```

$y = Wx + b$ in TensorFlow

```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                    dtype=tf.float32, name="x")

W = tf.get_variable(shape=[], name="W")

b = tf.get_variable(shape=[], name="b")
```

$y = Wx + b$ in TensorFlow

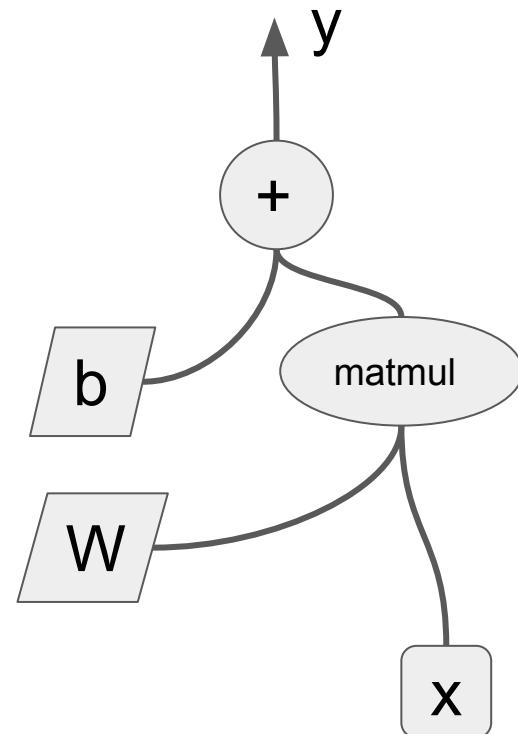
```
import tensorflow as tf

x = tf.placeholder(shape=[None],
                    dtype=tf.float32, name="x")

w = tf.get_variable(shape=[], name="w")

b = tf.get_variable(shape=[], name="b")

y = w * x + b
```

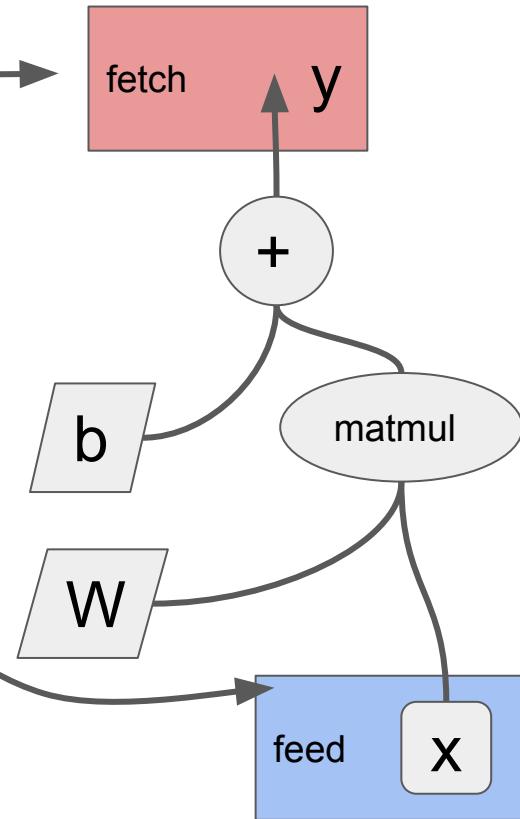


Running the Computation

`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



Putting it all together

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

with tf.Session() as sess:
    sess.run(tf.initialize_all_variables())
    print(sess.run(y, feed_dict={x: x_in}))
```

The diagram illustrates the execution flow of the provided TensorFlow code. It is divided into four main phases, each indicated by a curly brace:

- Build the graph**: This phase covers the declaration of tensors and variables. In the code, this includes the definition of `x`, `W`, and `b`.
- Prepare execution environment**: This phase covers the creation of a session. In the code, this is represented by the `with tf.Session() as sess:` block.
- Initialize variables**: This phase covers the initialization of variables. In the code, this is represented by the call to `sess.run(tf.initialize_all_variables())`.
- Run the computation (usually often)**: This phase covers the execution of operations. In the code, this is represented by the final `print` statement.

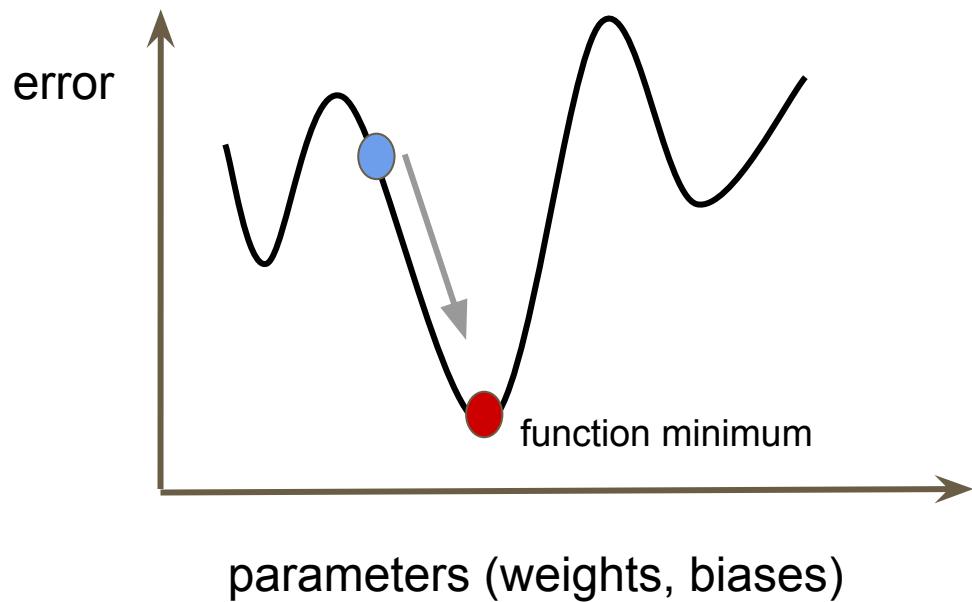
Define a Loss

Given y , y_{train} compute a loss, for instance:

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

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

Minimize loss: optimizers



`tf.train.AdadeltaOptimizer`

`tf.train.AdagradOptimizer`

`tf.train.AdagradDAOptimizer`

`tf.train.AdamOptimizer`

...

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

Putting it all together

```
loss = tf.reduce_mean(tf.square(y - y_train))  
optimizer = tf.train.GradientDescentOptimizer(0.5)  
train = optimizer.minimize(loss)  
  
with tf.Session() as sess:  
  
    sess.run(tf.initialize_all_variables())  
  
    for i in range(1000):  
  
        sess.run(train, feed_dict={x: x_in[i],  
                                  y_label: y_in[i]})
```

The diagram illustrates the components of the provided Python code. It uses three horizontal braces on the right side to group parts of the code:

- A brace groups the first three lines of code: `loss = ...`, `optimizer = ...`, and `train = ...`. This brace is labeled "Define a loss" in black text.
- A brace groups the line `sess.run(tf.initialize_all_variables())`. This brace is labeled "Create an optimizer" in blue text.
- A brace groups the entire loop structure starting with `with tf.Session() as sess:`. This brace is labeled "Op to minimize the loss" in green text.
- A brace groups the line `sess.run(tf.initialize_all_variables())`. This brace is labeled "Initialize variables" in black text.
- A brace groups the loop body, specifically the line `sess.run(train, feed_dict={x: x_in[i], y_label: y_in[i]})`. This brace is labeled "Iteratively run the training op" in orange text.

Linear Regression

2_linear_regression.ipynb

goo.gl/nrdsxM

Defining your own neural networks

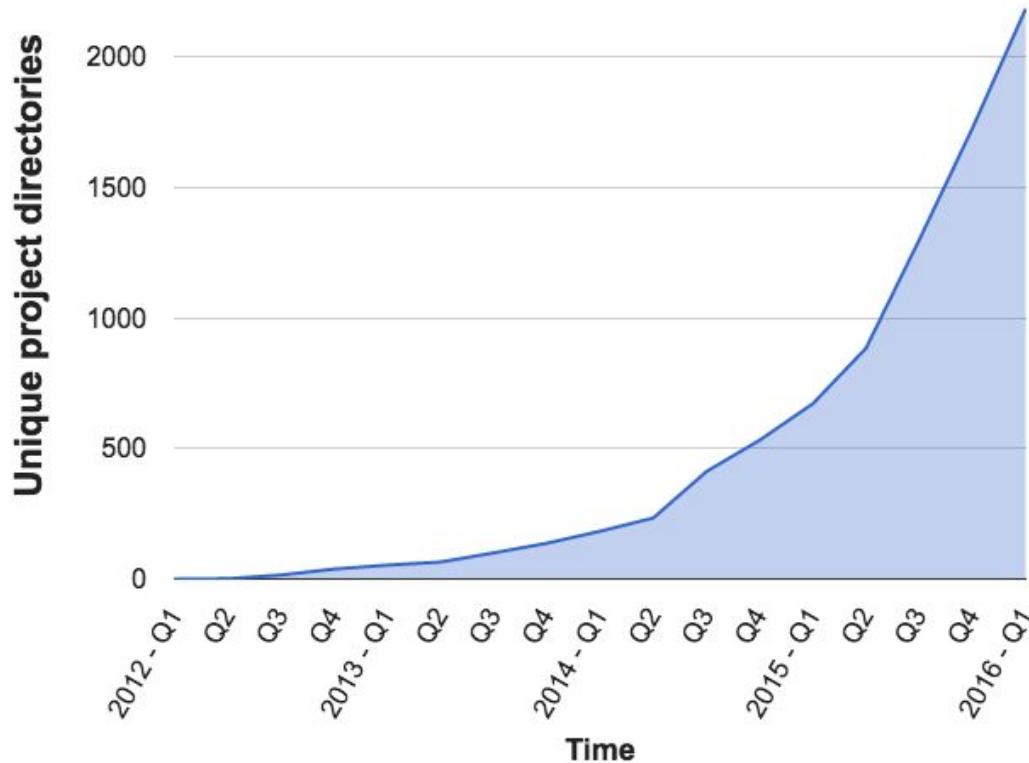
4_mnist_low_level.ipynb

goo.gl/nrdsxM

Deep Learning at Google

Growing Use of Deep Learning at Google

of directories containing model description files



AlphaGo

BBC News Sport Weather iPlayer TV Radio More Search

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

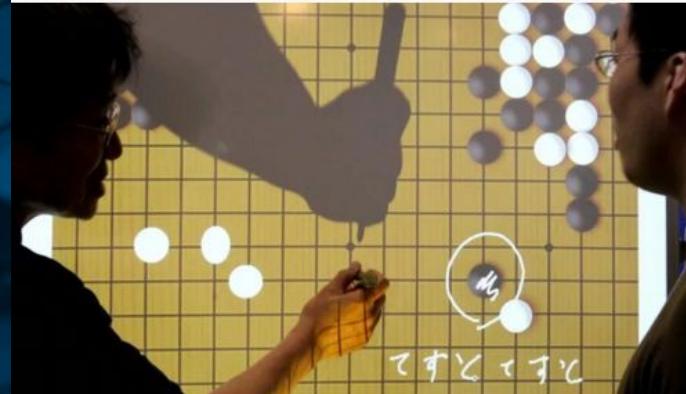
- How did they do it?
- What is the game Go?

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



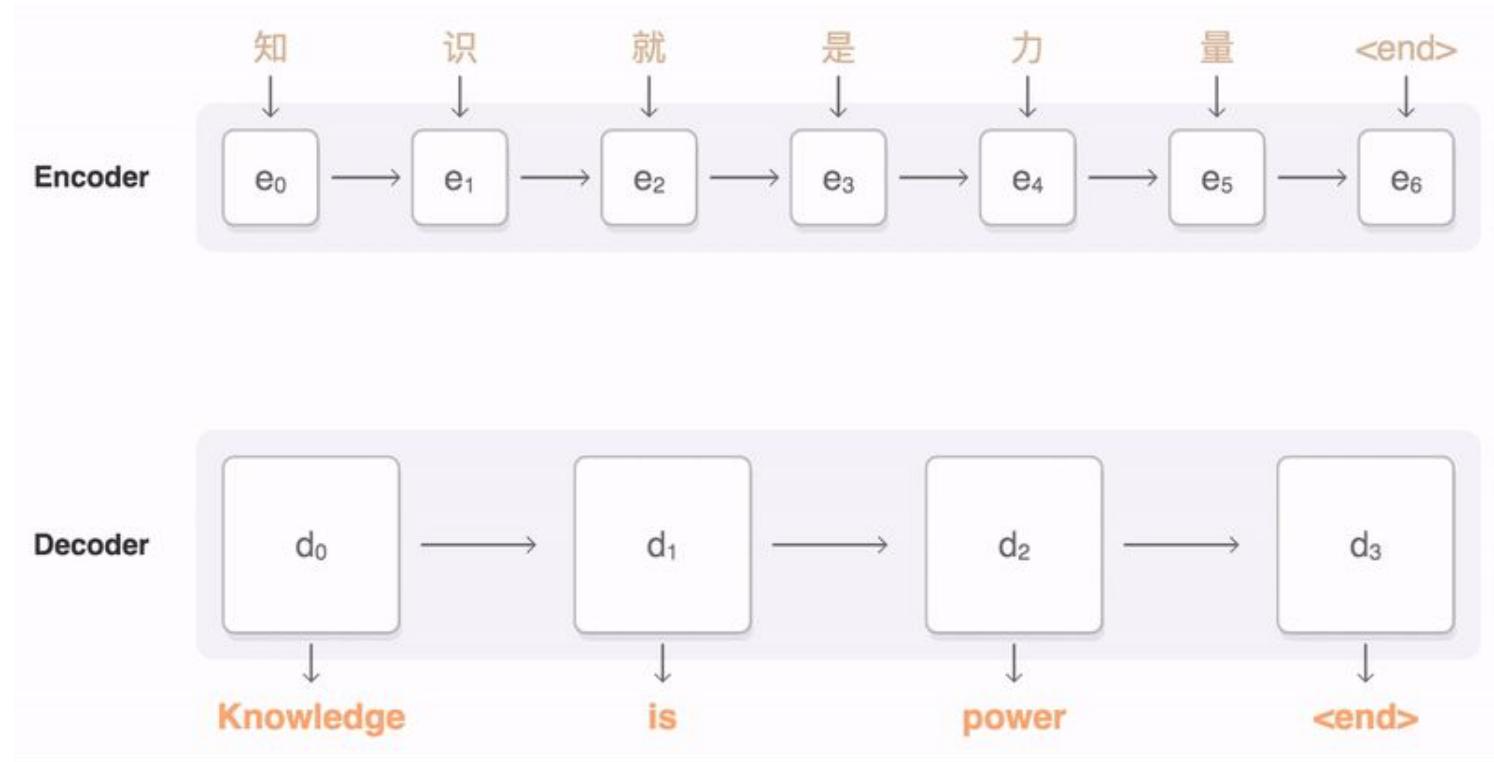
Going up. (Reuters/Kiyoshi Ota)

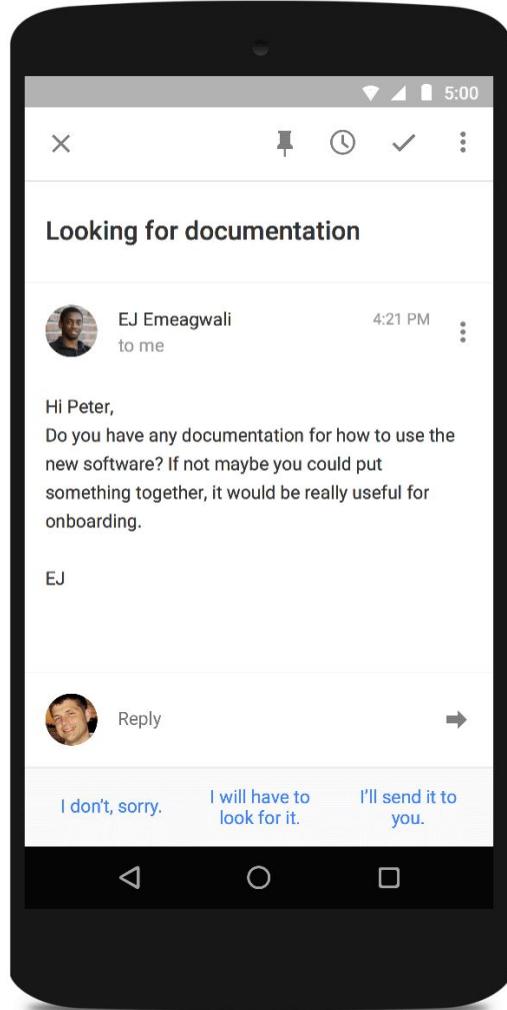
Computers have slowly started to encroach on activities we previously believed only the brilliantly sophisticated human brain could handle. IBM's Deep Blue supercomputer beat Grand Master Garry Kasparov at chess in 1997, and in 2011 IBM's Watson beat former human winners at the quiz game *Jeopardy*. 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.



EXIT

Google Neural Machine Translation



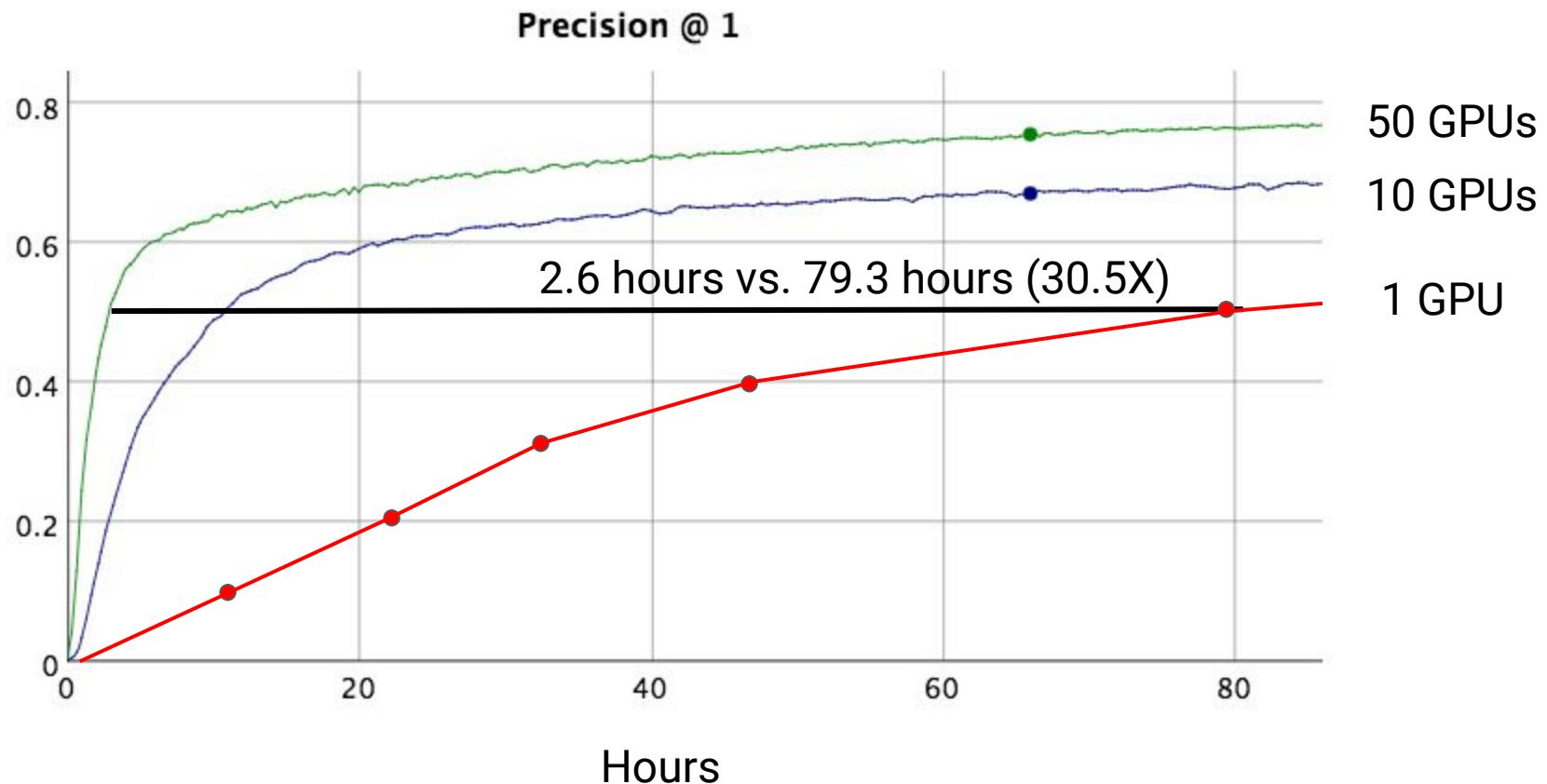


Smart reply in Inbox by Gmail

10%

of all responses
sent on mobile

Image Model (Inception) Synchronous Training



Bonus material



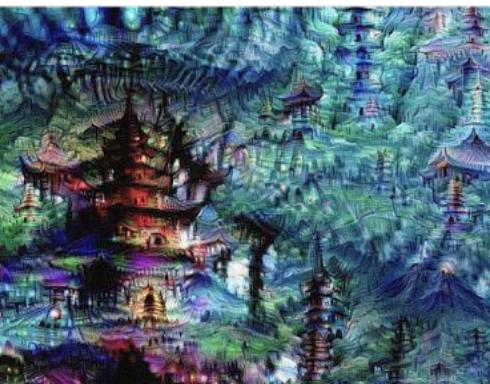
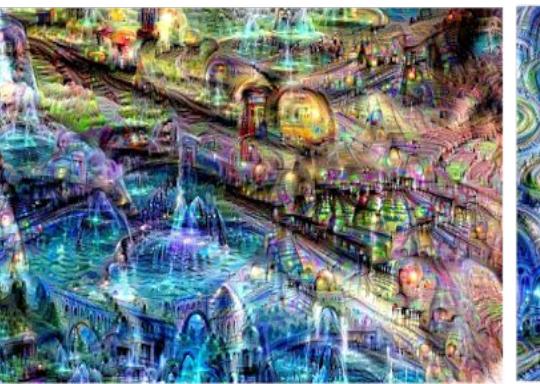
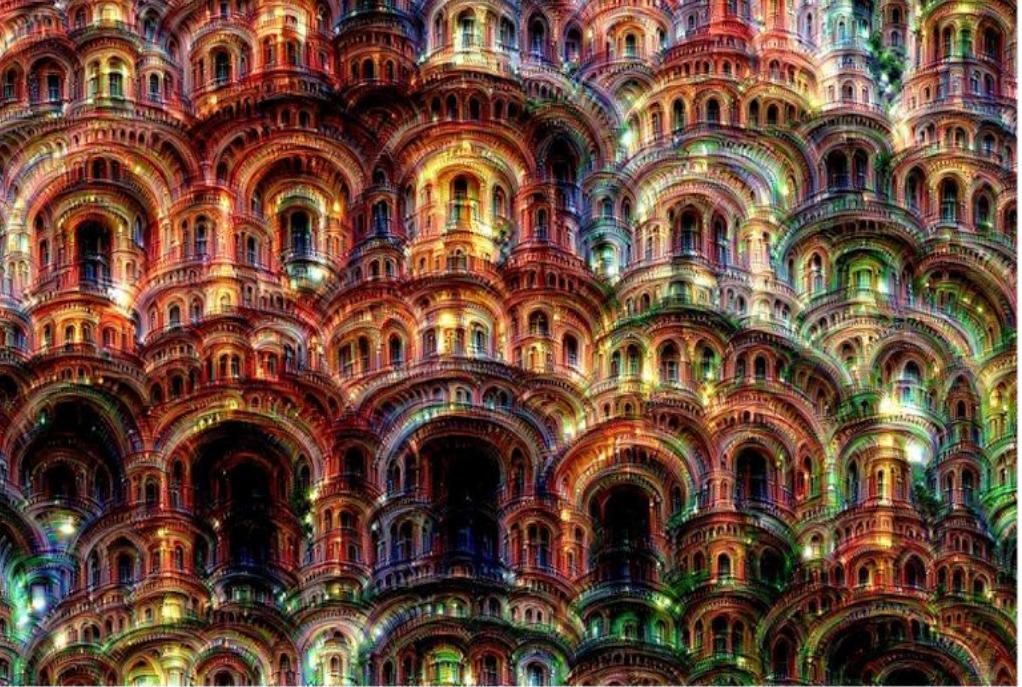
Image: Leonid Afremov – Rain Princess



Image: Stephen Wilkes, National Geographic

A vibrant, abstract painting of a city skyline at night, featuring tall buildings with colorful lights and a busy street below filled with yellow taxis.

goo.gl/fyDxhC





goo.gl/OCYseb

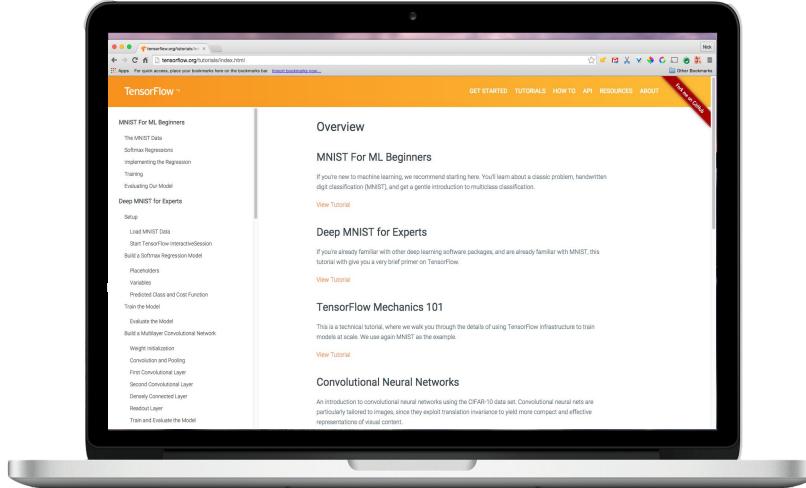
Next steps

Tutorials and code
tensorflow.org

Deep Learning by Google
Udacity class goo.gl/iHssII

Totally new to ML?
Recipes goo.gl/KewA03

Learn more about Neural Networks?
cs231n.github.io



Thank you and have fun!



Josh Gordon
[@random_forests](https://twitter.com/random_forests)



Wolff Dobson
[@greenexecutive](https://twitter.com/greenexecutive)