Intro to TensorFlow

Singapore, December 2016

Josh Gordon
@random_forests

Wolff Dobson
@greenexecutive

Image source: Wikipedia

Agenda

1pm - 1:30pm

- Fibonacci sequence

1:30pm - 2pm

- Linear regression

2:15pm to 3pm

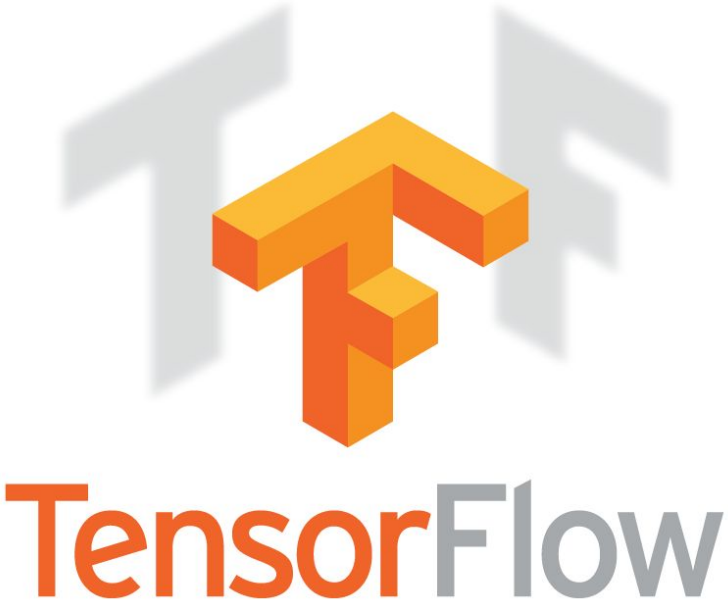
- Create your own artwork, or train your own image classifier

3pm to 4pm

- Basic and Deep MNIST, and tf.contrib.learn

cd tensorflow-workshop; git pull

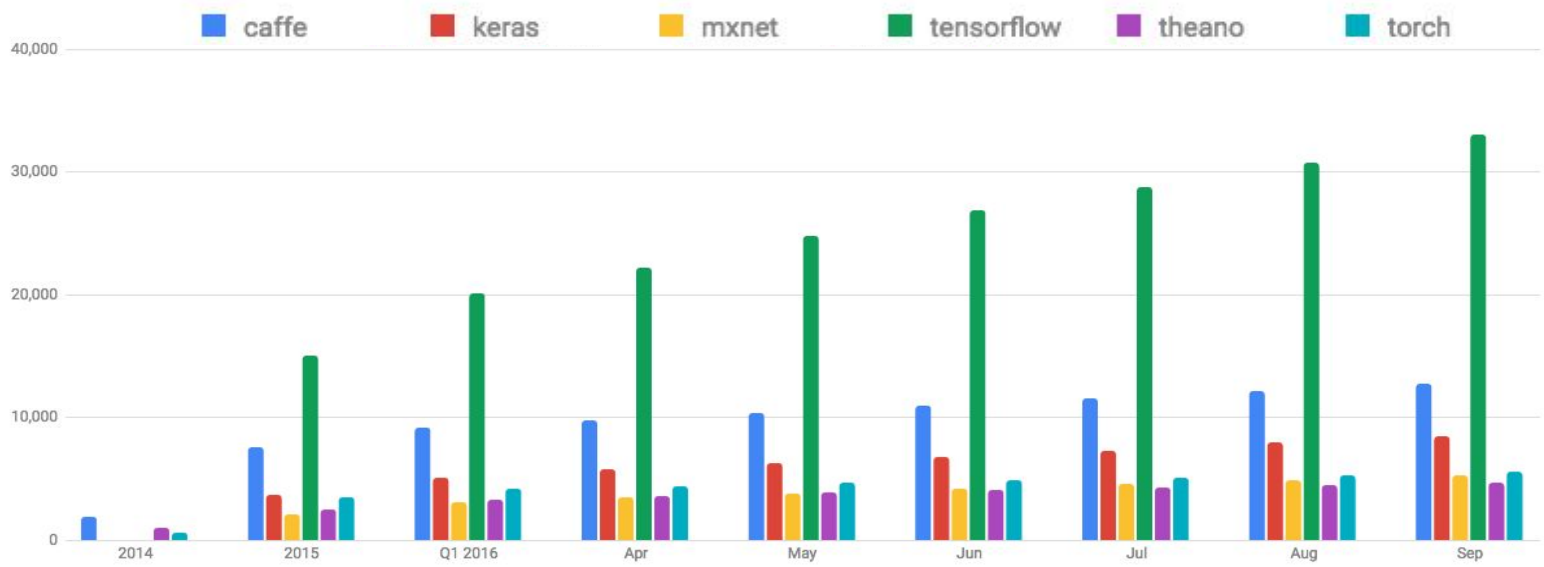
TensorFlow



- Fast, flexible, and scalable open-source machine learning library
- For research and production
- Runs on CPU, GPU, Android, iOS, Raspberry PI
- Apache 2.0 license

https://research.googleblog.com/2016/11/celebrating-tensorflows-first-year.html

Cumulative GitHub Stars



Progress last year

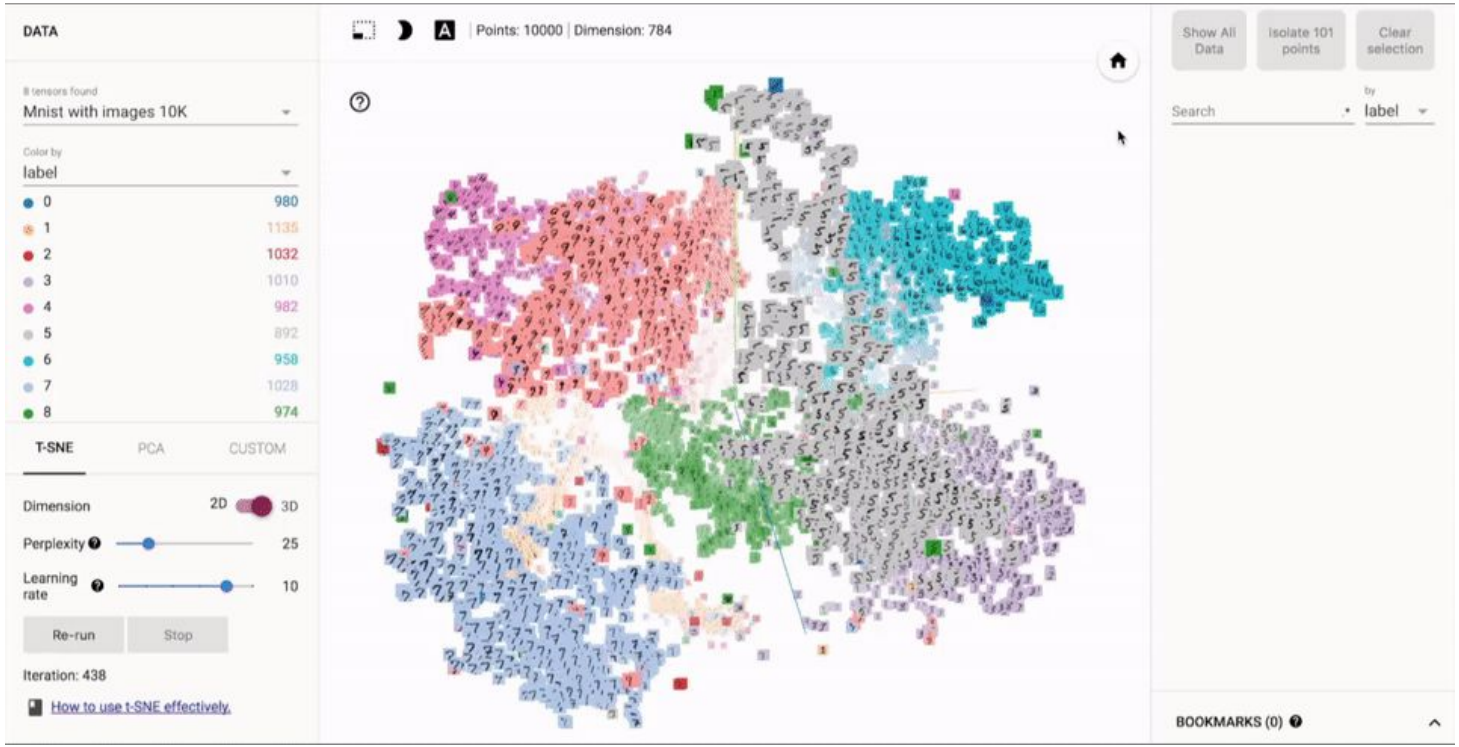
Release	Milestone
November 2015	Initial release
December (0.6)	Faster on GPUs; Python 3.3+
February (0.7)	TensorFlow Serving
April (0.8)	Distributed TensorFlow
June (0.9)	iOS; Mac GPU
August (0.10)	Slim
October (0.11)	HDFS; CUDA 8, CuDNN 5

https://github.com/tensorflow/tensorflow/releases

Progress last year cont’d

Release	Milestone
November (0.12)	Windows 7, 10, and Server 2016; TensorBoard Embedding Visualizer

TensorBoard Embedding Visualization: goo.gl/xfhi1H



Why?

Reproducible Research

Music https://magenta.tensorflow.org/2016/11/09/tuning-recurrent-networks-with-reinforcement-learning/

WaveNet https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Translation https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

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

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

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

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

Open Source Models: from Google and *Many* Others

Fast Style Transfer https://github.com/lengstrom/fast-style-transfer/

Colornet https://github.com/pavelgonchar/colornet

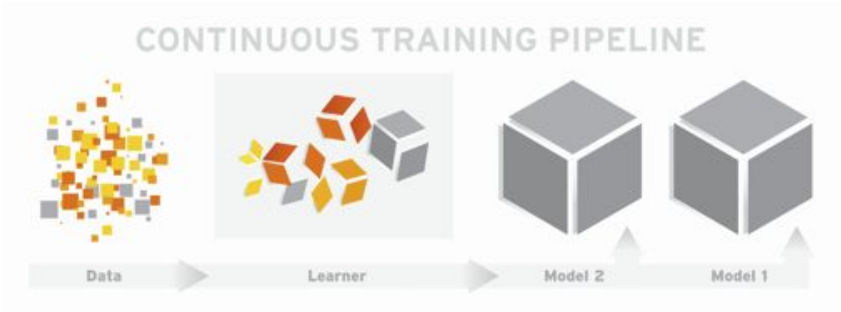
Super Resolution https://github.com/david-gpu/srez

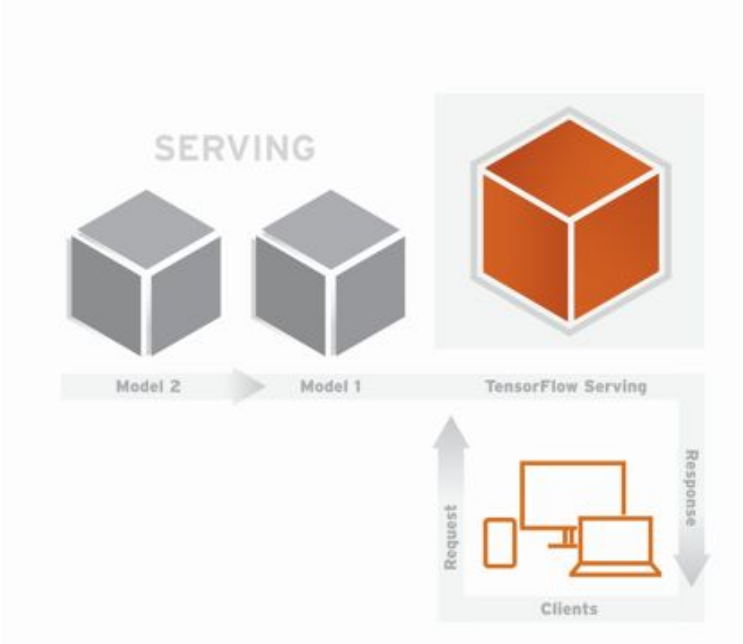
TTS https://github.com/ibab/tensorflow-wavenet

Speech Recongition https://github.com/buriburisuri/speech-to-text-wavenet

Many more https://github.com/jtoy/awesome-tensorflow

Research *and* Production





https://research.googleblog.com/2016/02/running-your-models-in-production-with.html

Warm up

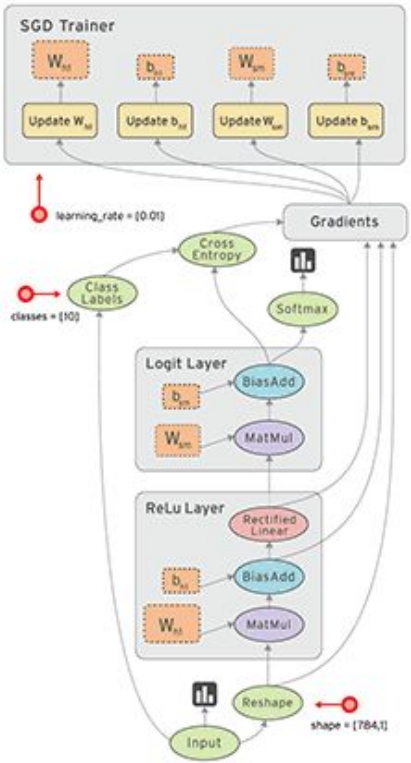
Graphs and Sessions

A multidimensional array.

A graph of operations.

A graph is like a blueprint

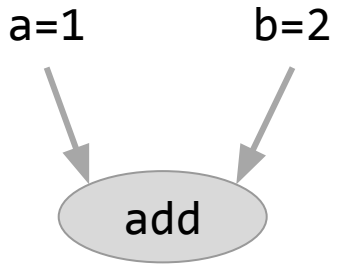
- You define the graph in high-level language (Python)
- Graph is compiled and optimized
- Graph is executed (in part, or fully) on devices
- Nodes represent computations
- Data (tensors) flows between them



But, why?

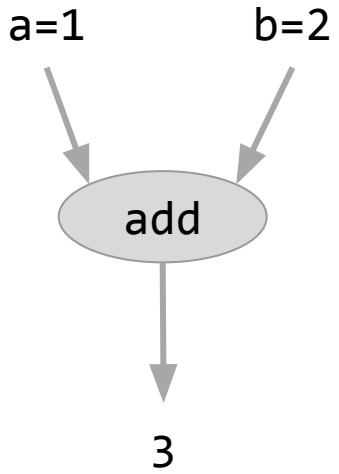
Build a graph...

```
import tensorflow as tf
a = tf.constant(1)
b = tf.constant(2)
c = tf.add(a, b)
```



… then run it

```
import tensorflow as tf
a = tf.constant(1)
b = tf.constant(2)
c = tf.add(a, b)

session = tf.Session()
value_of_c = session.run(c)
```

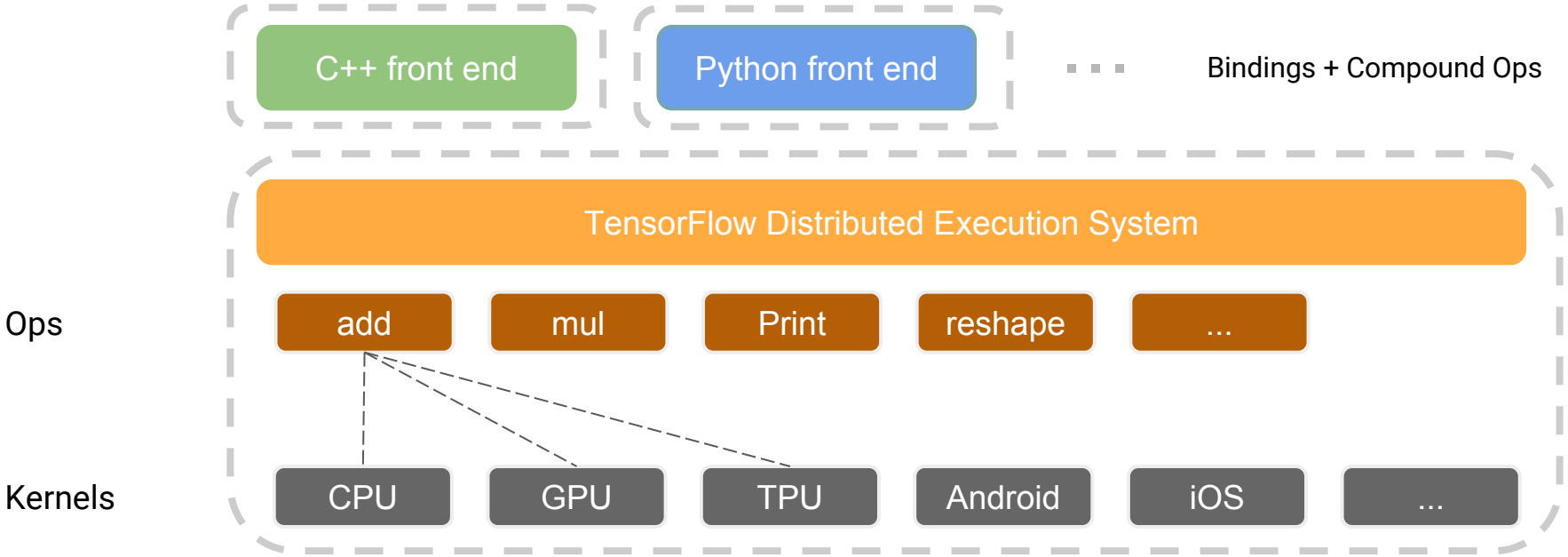


Building graphs looks mostly like numpy

With special operations for deep learning

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

Architecture



What's the output?

```
import tensorflow as tf

my_var = tf.Variable(0)
add_op = tf.add(my_var,1)
assign_op = tf.assign(my_var, add_op)

session = tf.Session()
session.run(tf.global_variables_initializer())

for _ in range(3):
    print (session.run(assign_op))
```

Exercises
goo.gl/nrdsxM

Clone the repo and start a notebook

```
$ git clone https://github.com/random-forests/tensorflow-workshop
$ cd tensorflow-workshop
$ jupyter notebook
```

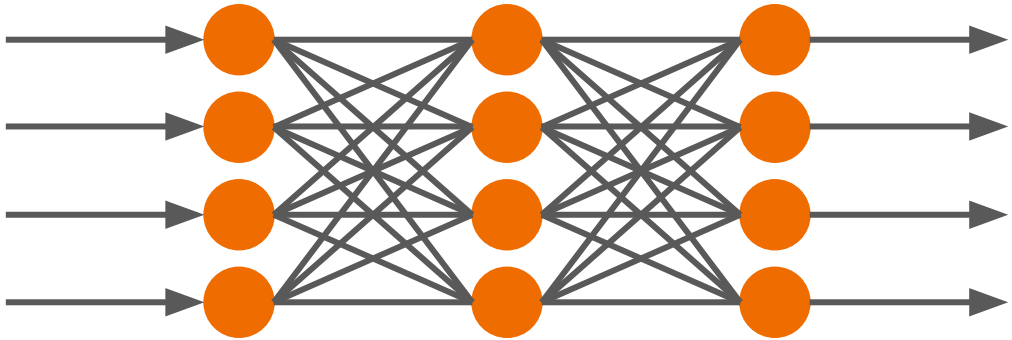
goo.gl/nrdsxM

Exercise #1: Fibonacci Sequence
1_warm_up.ipynb

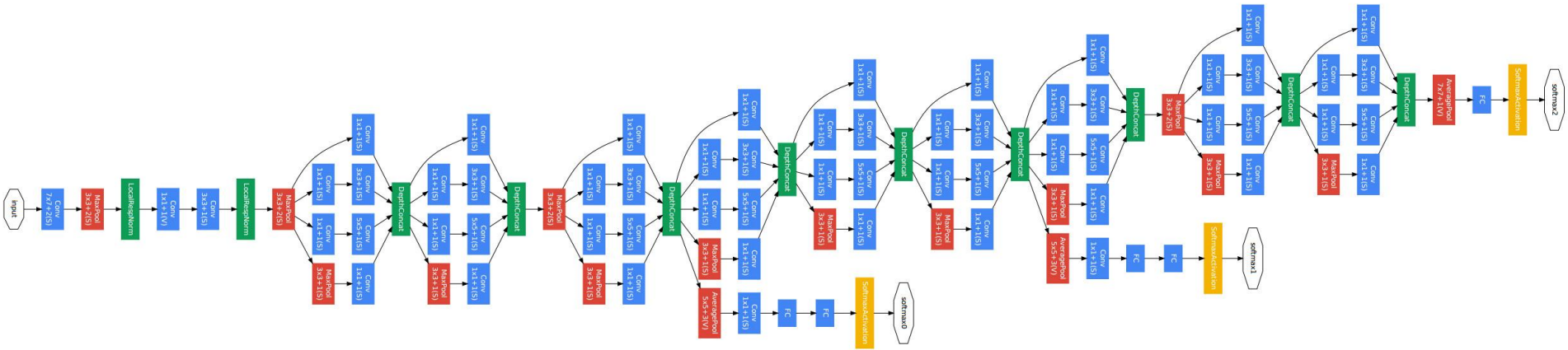
goo.gl/nrdsxM

But why?

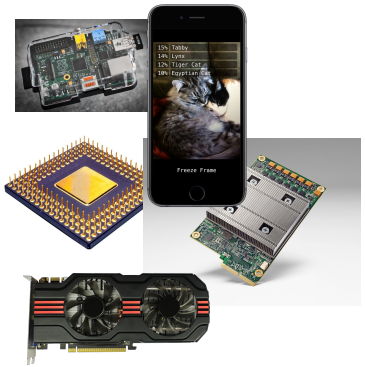
Neural Networks, yesterday



Remember Inception?

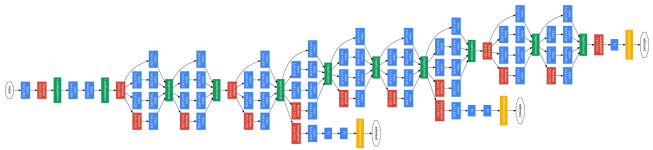


Machine Learning gets complex quickly



Heterogenous systems

Distributed systems

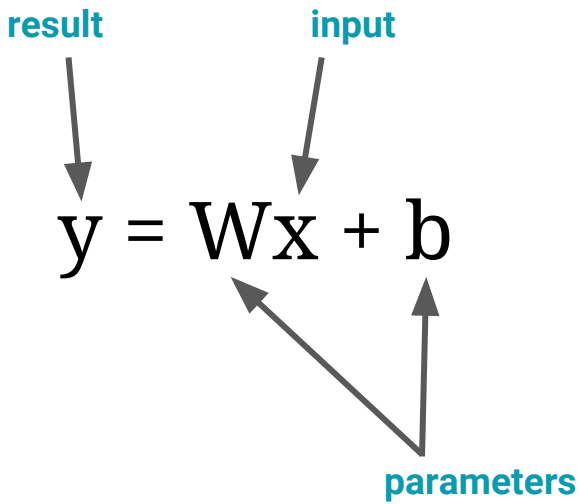


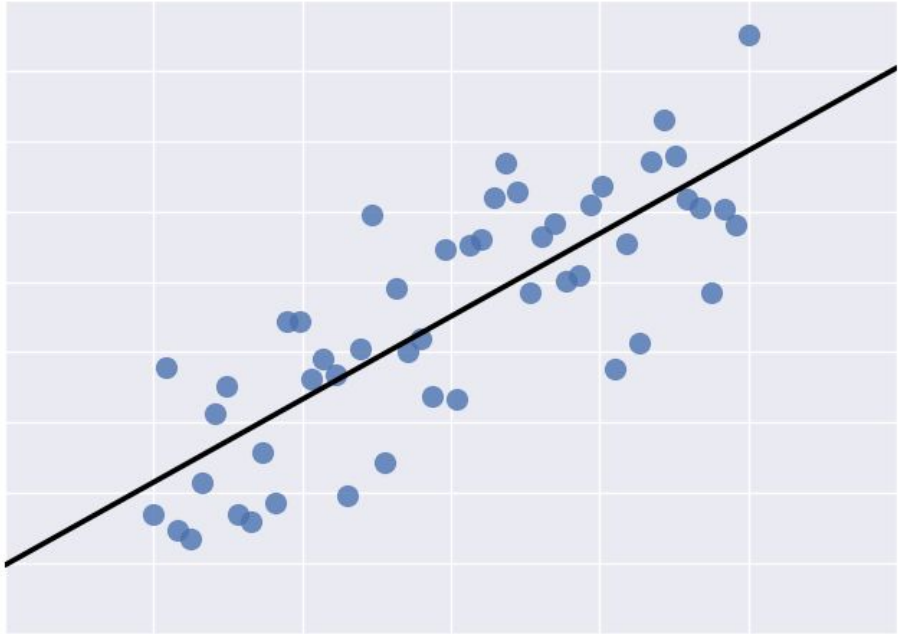
Modeling complexity

Linear Regression

Inference, Loss, Optimize

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*) samples, figure out the value of *W* and *b*.

Let's code this up

1. **Inference** (“given x, this is the code to predicts y”)

2. **Loss** (“quantify how bad our prediction was”)

3. **Optimize** (“update variables to improve the prediction”)

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

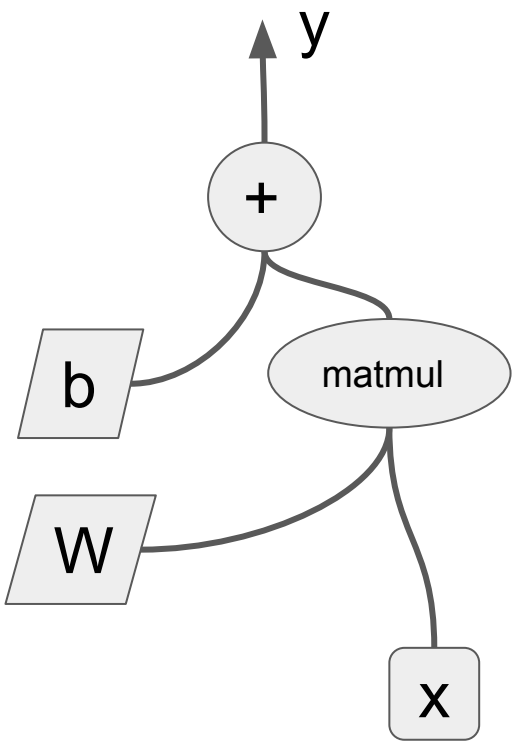
```
import tensorflow as tf

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

W = tf.Variable(tf.random_normal([1], name="W")
```

y = Wx + b in TensorFlow

```
import tensorflow as tf

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

W = tf.Variable(tf.random_normal([1], name="W")

b = tf.Variable(tf.random_normal([1], name="b")
```

y = Wx + b in TensorFlow

```
import tensorflow as tf

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

W = tf.Variable(tf.random_normal([1], name="W")

b = tf.Variable(tf.random_normal([1], name="b")

y = W * x + b
```



Variables Must be Initialized

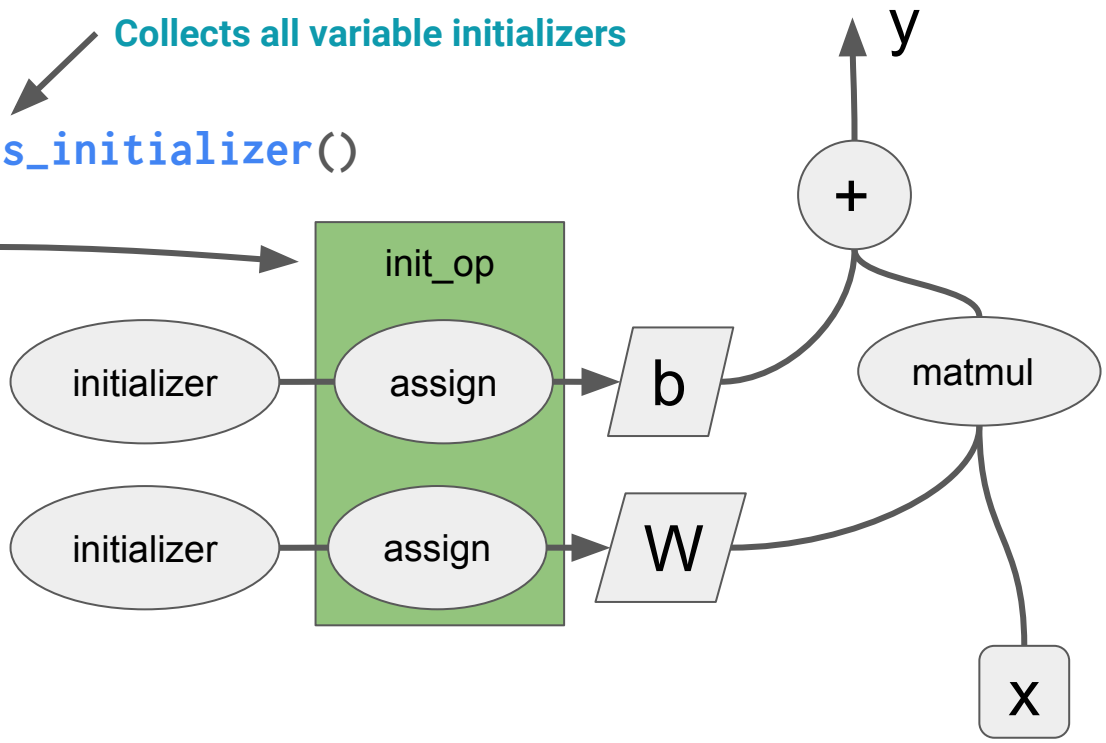
Collects all variable initializers

```
init_op = tf.global_variables_initializer()
```

Makes an execution environment

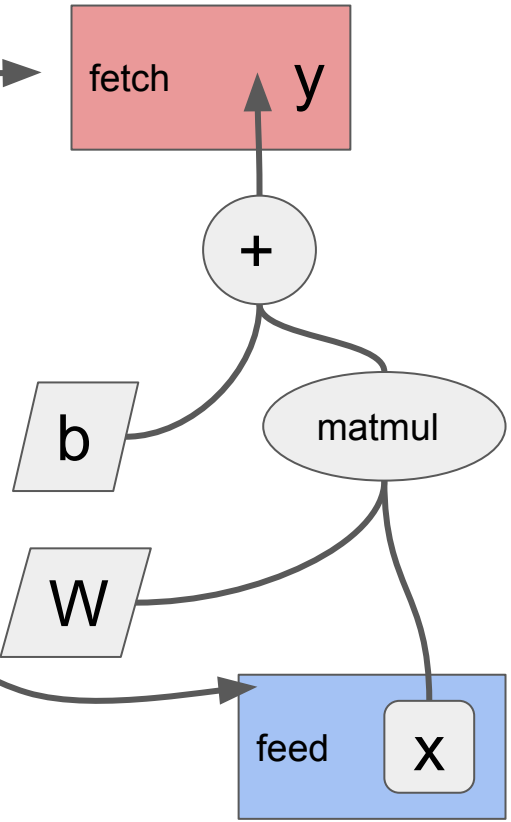
```
sess = tf.Session()

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

Inference: putting it all together

```
import tensorflow as tf
x = tf.placeholder(shape=[None],
                   dtype=tf.float32,
                   name='x')
W = tf.Variable(tf.random_normal([1], name="W"))
b = tf.Variable(tf.random_normal([1], 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}))
```

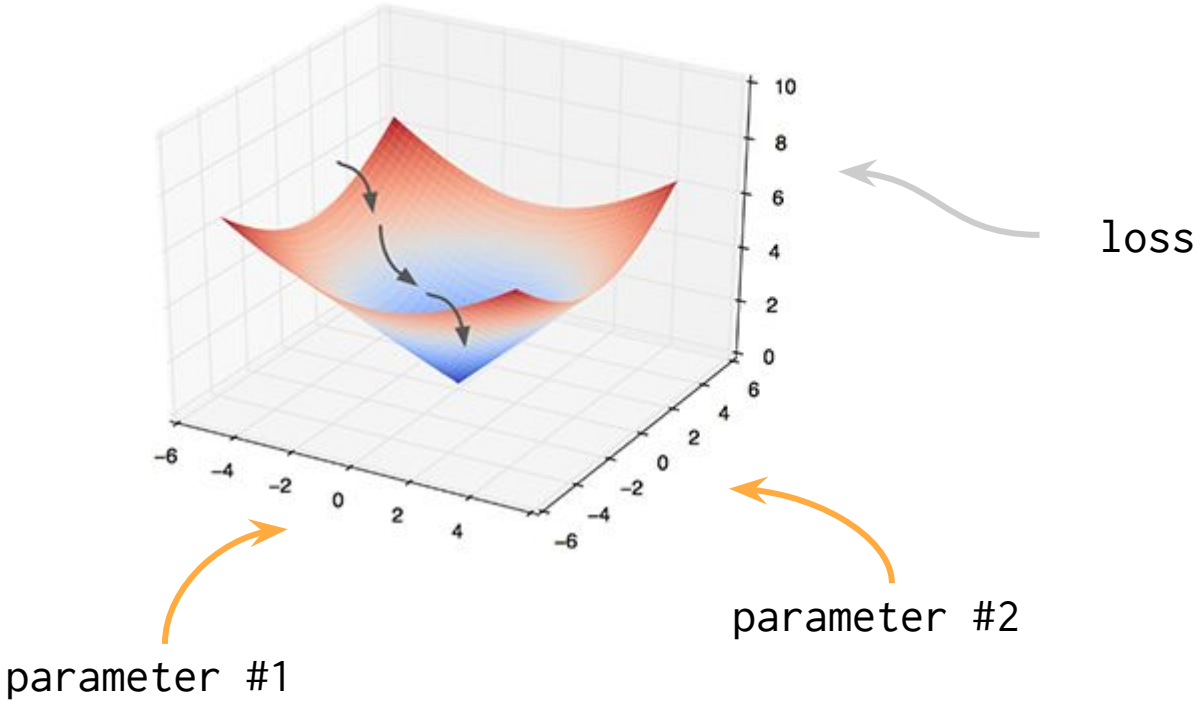
Build the graph

Prepare execution env

Initialize variables

Run the computation

Concepts: Loss and Gradient Descent



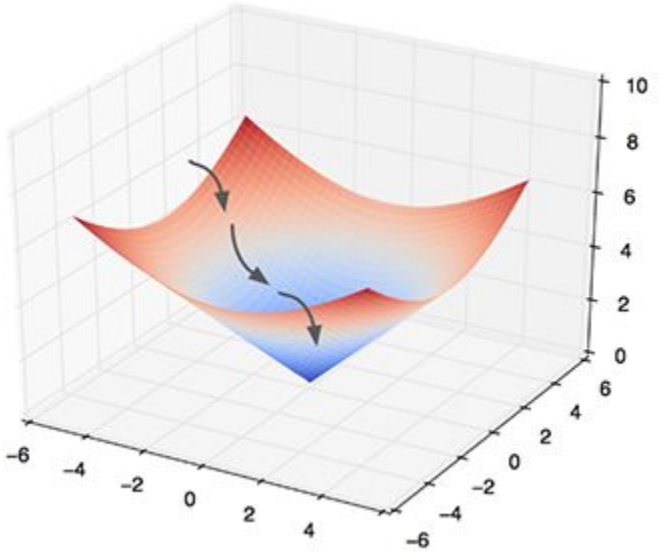
Define a Loss

Given y, y_train compute a loss, for instance:

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

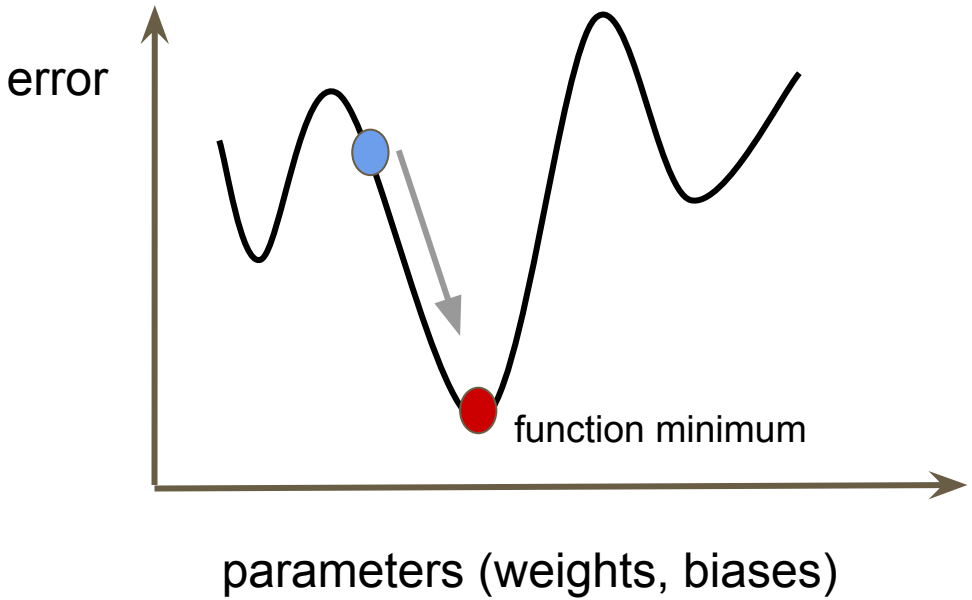
```
# create an operation that calculates loss
loss = tf.reduce_mean(tf.square(y - y_train))
```

Optimization



	-5	-4	-3	-2	-1	0	1	2	3	4	5
5	**50**	41	34	29	26	25	26	29	34	41	50
4	41	**32**	**25**	20	17	16	17	20	25	32	41
3	34	25	18	**13**	10	9	10	13	18	25	34
2	29	20	13	**8**	5	4	5	8	13	20	29
1	26	17	10	**5**	**2**	**1**	2	5	10	17	26
0	25	16	9		1	**0**	1	4	9	16	25
-1	26	17	10	5	2	1	2	5	10	17	26
-2	29	20	13	8	5	4	5	8	13	20	29
-3	34	25	18	13	10	9	10	13	18	25	34
-4	41	32	25	20	17	16	17	20	25	32	41
-5	50	41	34	29	26	25	26	29	34	41	50

Minimize loss using an optimizer



tf.train.GradientDescentOptimizer

Or, use one of:

AdadeltaOptimizer

AdagradOptimizer

AdagradDAOptimizer

AdamOptimizer

...

Automatic Differentiation

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

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

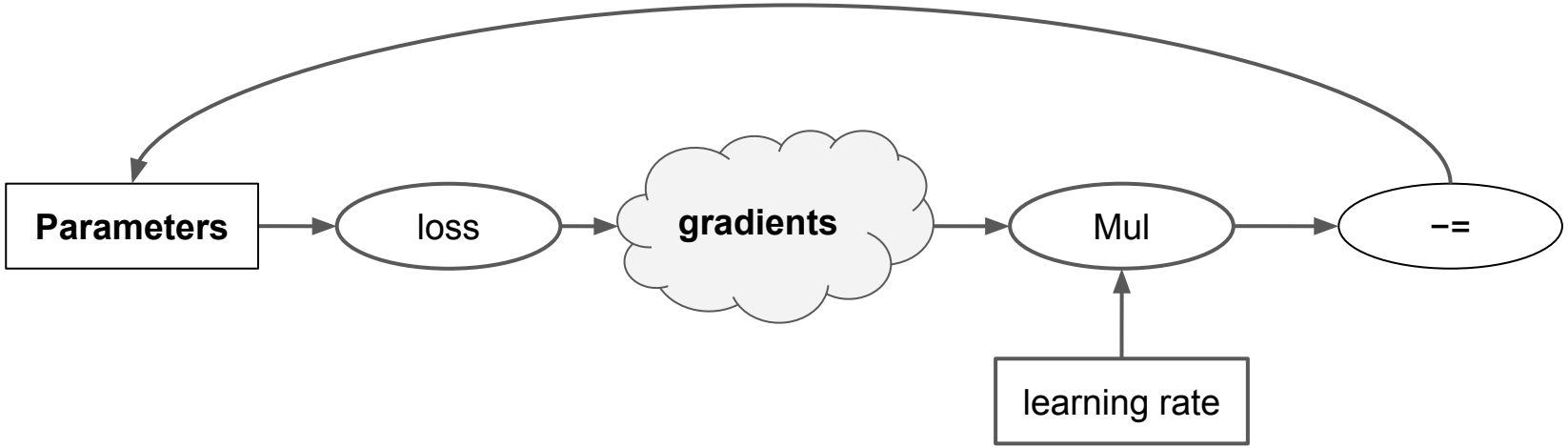
$$b \leftarrow b - \eta \, (dL/db)$$

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

Behind the scenes



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_data[i],
                               y_label: y_data[i]})
```

Define a loss

Create an optimizer

Op to minimize the loss

Initialize variables

Iteratively run the training op

Exercise #2: Linear Regression
2_linear_regression.ipynb

goo.gl/nrdsxM

Demo: TensorBoard

goo.gl/nrdsxM

Neural Networks

IS THIS A
CAT or DOG?

CAT DOG

OUTPUT LAYER

ACTIVATED NEURONS

INPUT LAYER

DEEP *neural* NETWORK

Deep Learning

Current state of the art in:

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

Neural Networks in TensorFlow

Options:

1. Use a pre-trained model

2. Use a higher level API that build on top of TensorFlow

3. Define and train your network directly in TensorFlow

Open Source Models

Understanding Images

Inception

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

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

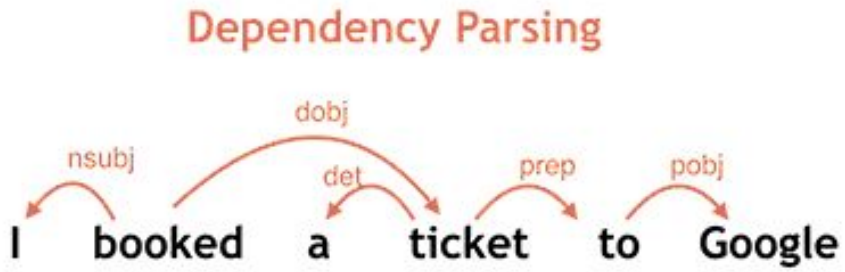
Demo

Show and Tell

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

Understanding Text

Parsey McParseface



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

Text Summarization

Original text

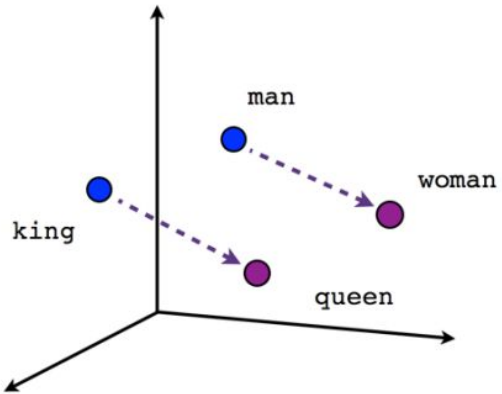
- *Alice and Bob took the train to visit the zoo. They saw a **baby giraffe, a lion, and a flock of colorful tropical birds**.*

Abstractive summary

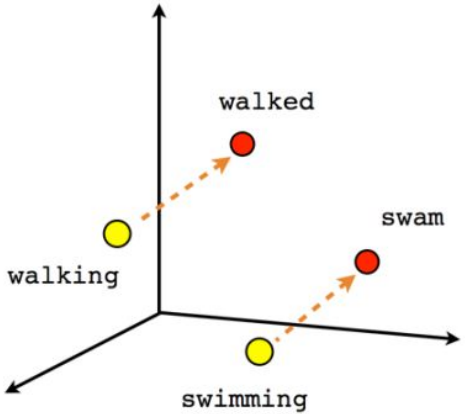
- *Alice and Bob visited the zoo and saw **animals and birds**.*

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

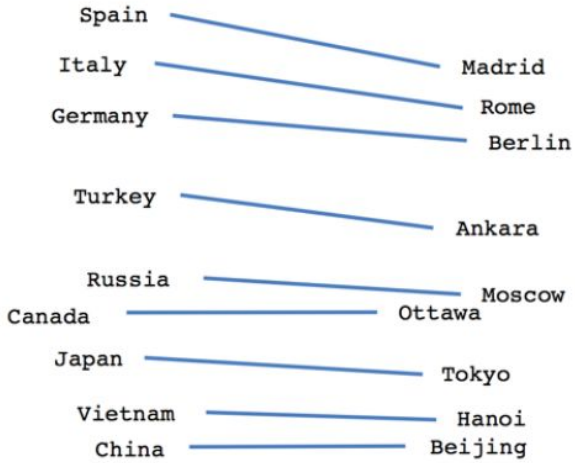
Word Embeddings



Male-Female



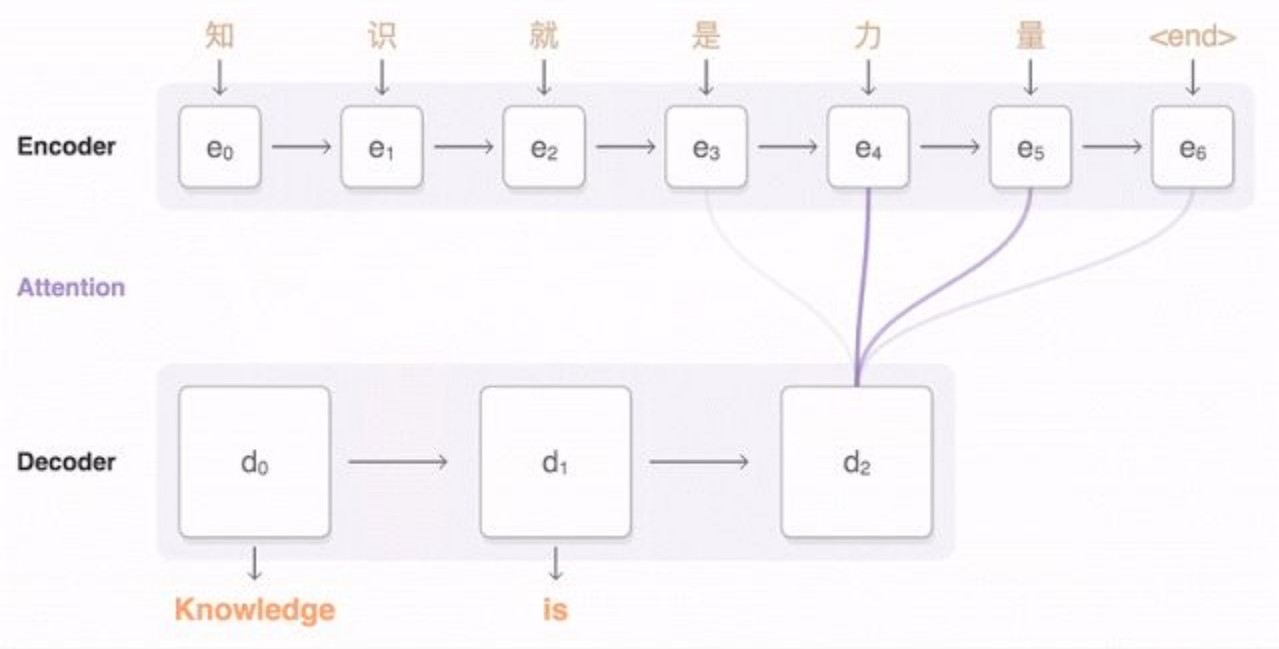
Verb tense



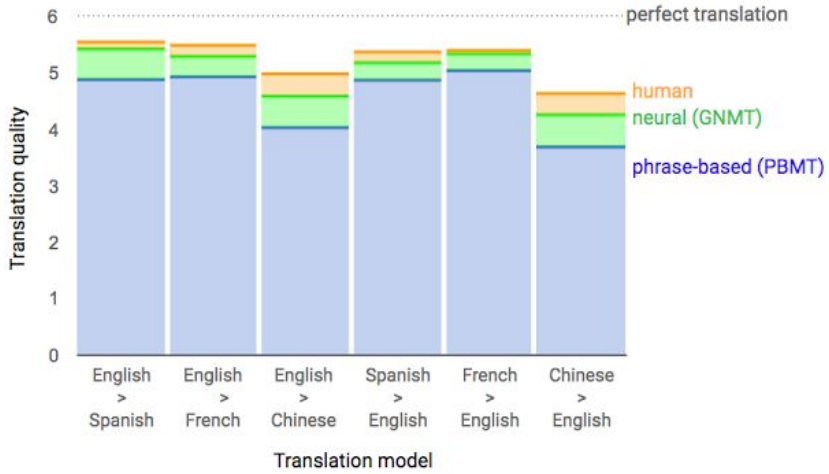
Country-Capital

https://www.tensorflow.org/versions/r0.12/tutorials/word2vec/index.html#vector-representations-of-words

Translation



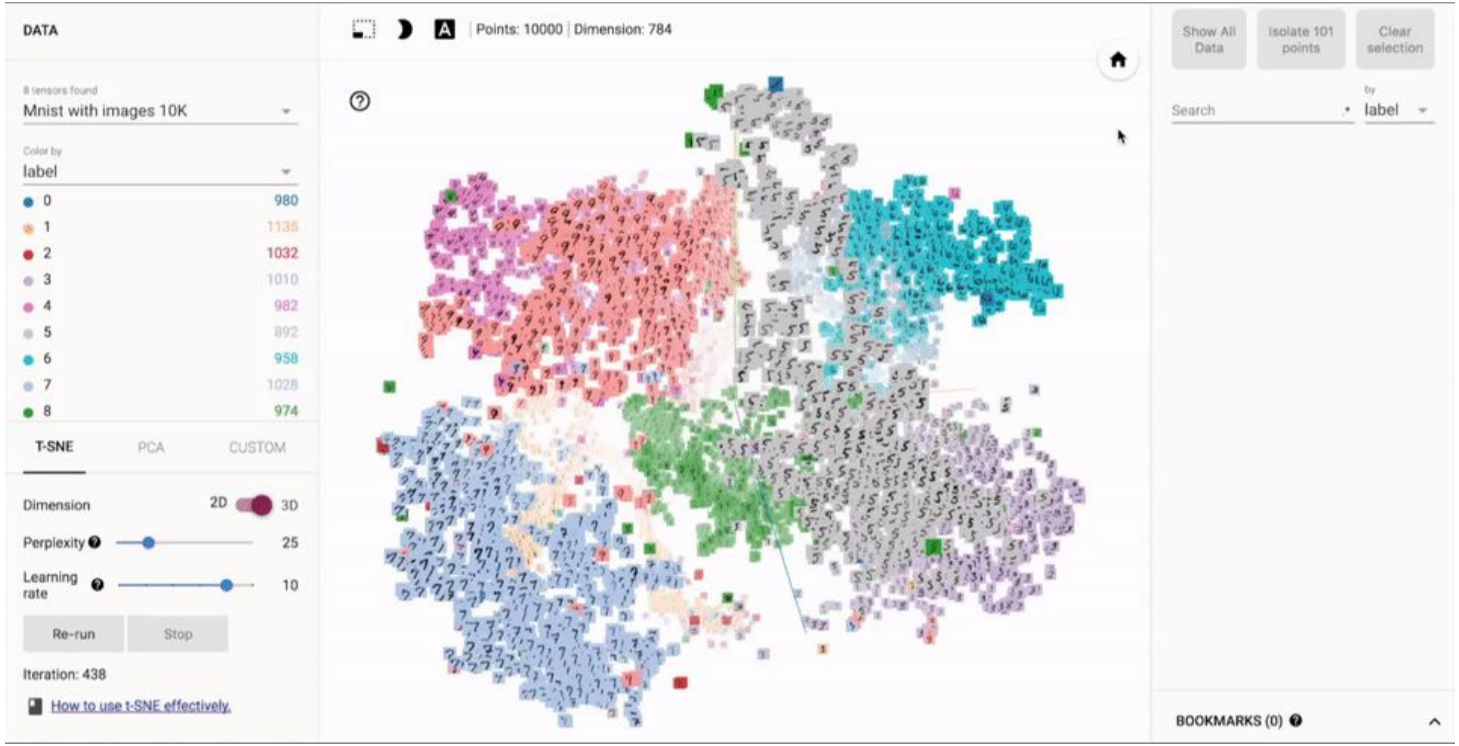
https://research.googleblog.com/2016/09/a-neural-network-for-machine.html



Input sentence:	*Translation (PBMT):*	*Translation (GNMT):*	*Translation (human):*
李克強此行將啟動中加總理年度對話機制，與加拿大總理杜魯多舉行兩國總理首次年度對話。	Li Keqiang premier added this line to start the annual dialogue mechanism with the Canadian Prime Minister Trudeau two prime ministers held its first annual session.	Li Keqiang will start the annual dialogue mechanism with Prime Minister Trudeau of Canada and hold the first annual dialogue between the two premiers.	Li Keqiang will initiate the annual dialogue mechanism between premiers of China and Canada during this visit, and hold the first annual dialogue with Premier Trudeau of Canada.

https://research.googleblog.com/2016/09/a-neural-network-for-machine.html

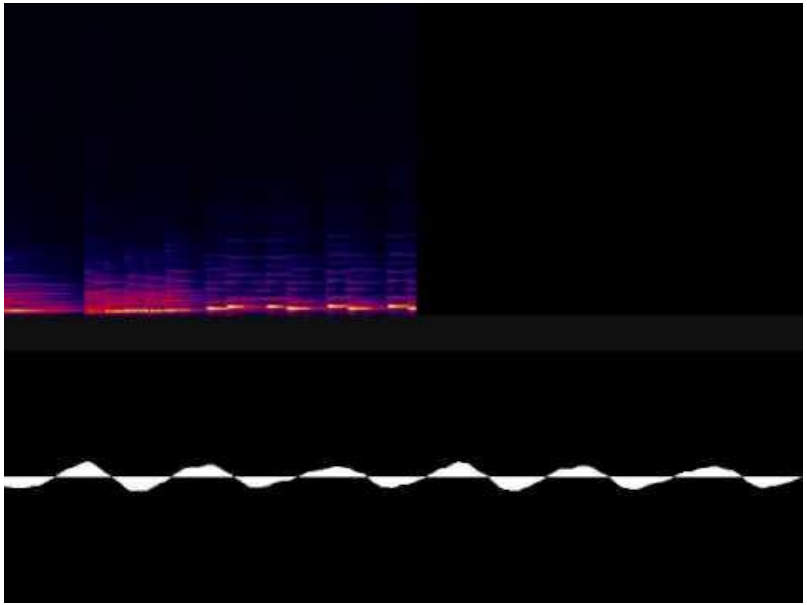
TensorBoard



https://www.tensorflow.org/versions/r0.12/how_tos/embedding_viz/index.html

Speech and Music

MAGENTA



https://magenta.tensorflow.org/2016/11/09/tuning-recurrent-networks-with-reinforcement-learning/

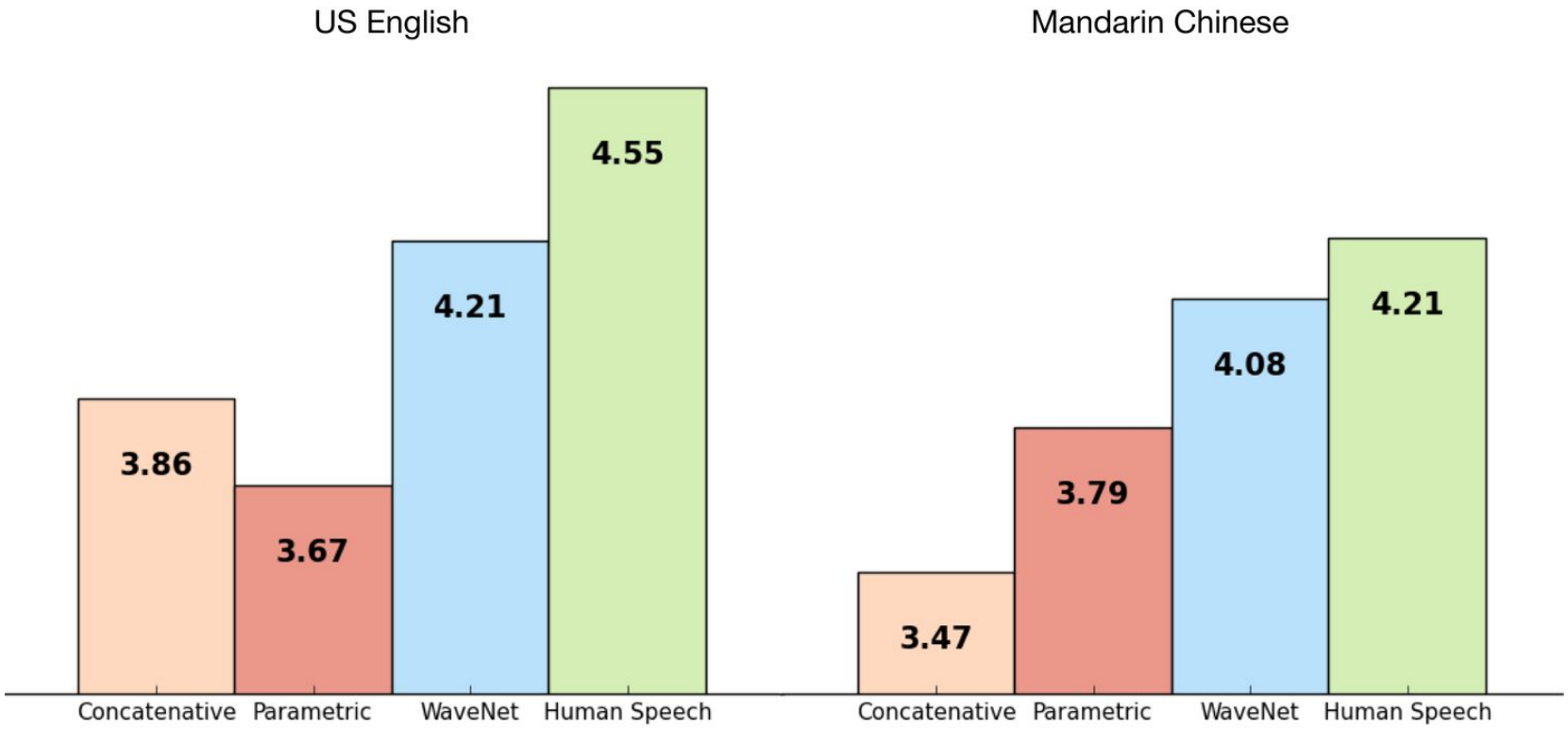
TTS

English - Parametric
English - Concatenative
English - WaveNet
Mandarin - Parametric
Mandarin - Concatenative
Mandarin - WaveNet
Bonus - on hold

https://deepmind.com/blog/wavenet-generative-model-raw-audio/

https://magenta.tensorflow.org/2016/11/09/tuning-recurrent-networks-with-reinforcement-learning/

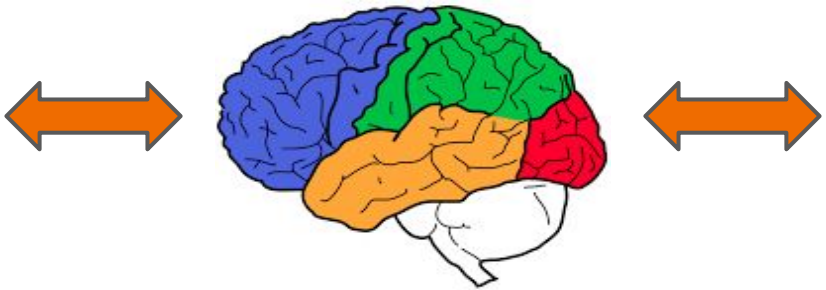
WaveNet



https://deepmind.com/blog/wavenet-generative-model-raw-audio/

“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

Deep Learning and Art

Image source - Wikipedia

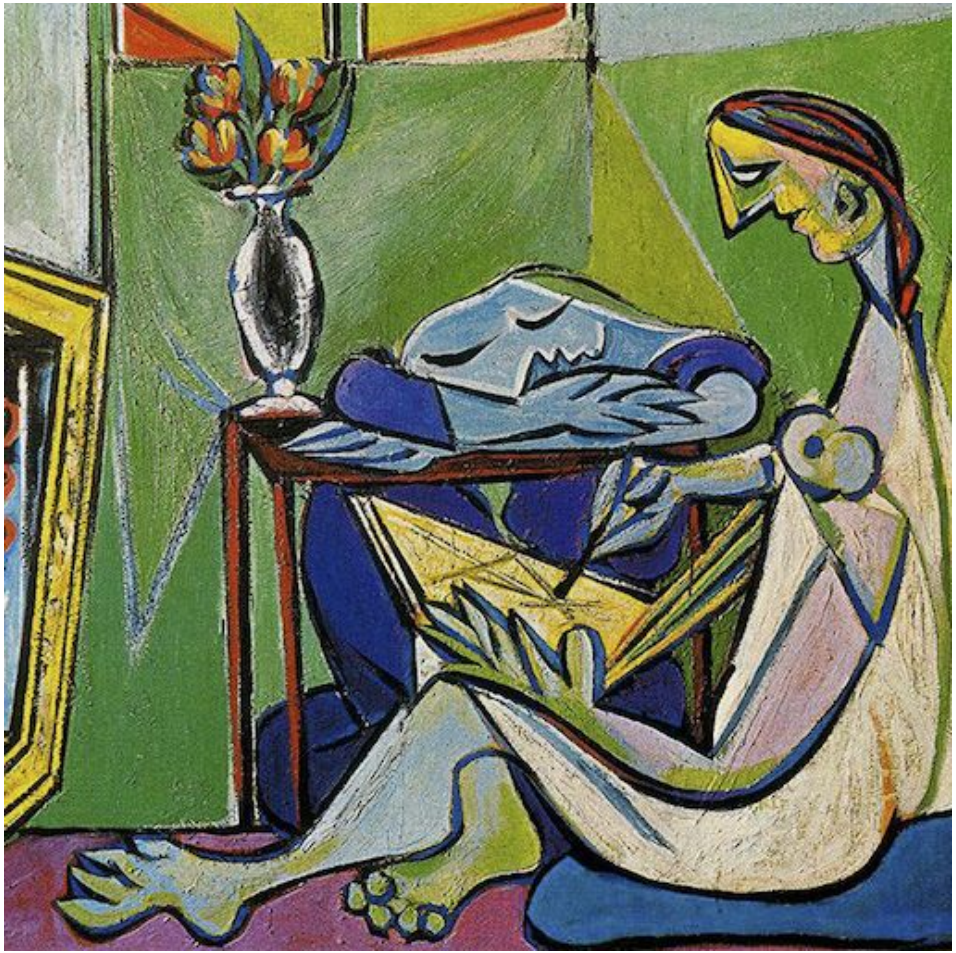
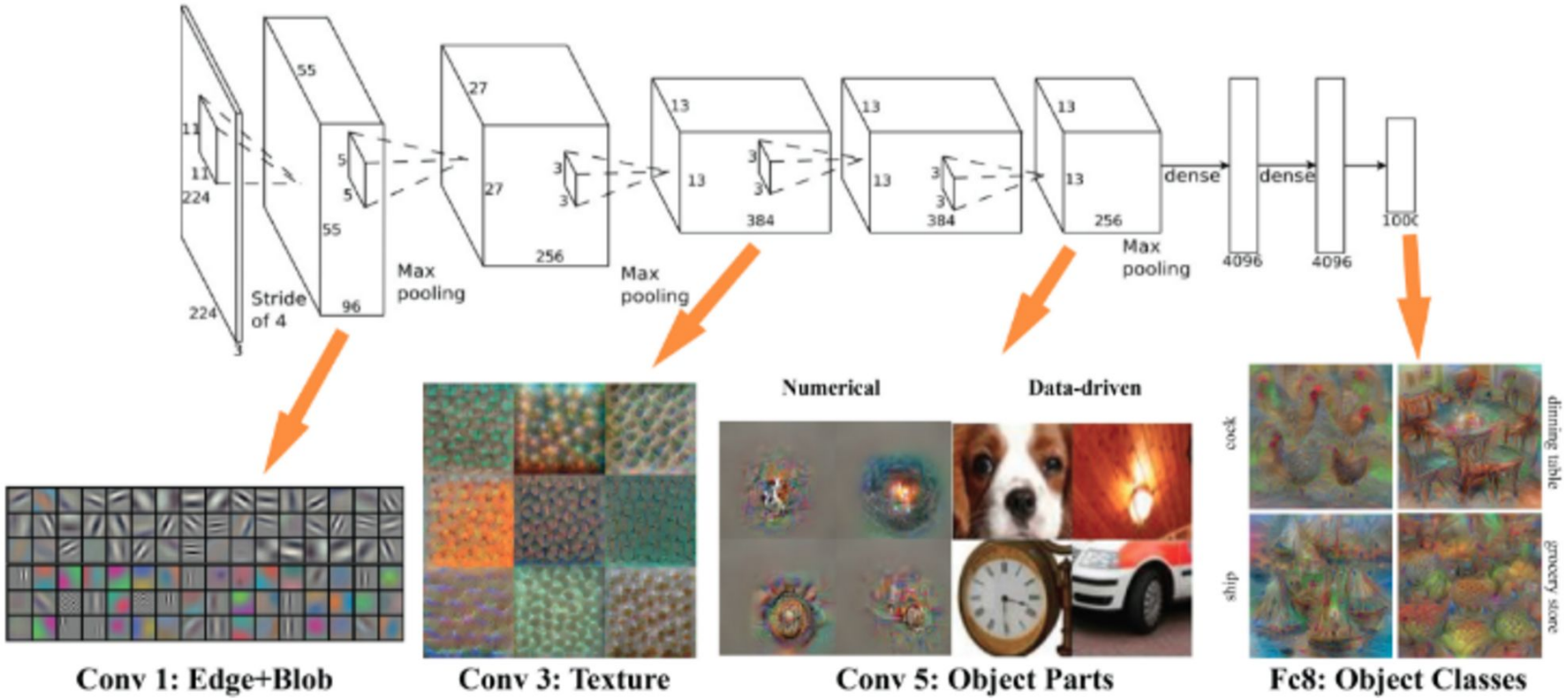
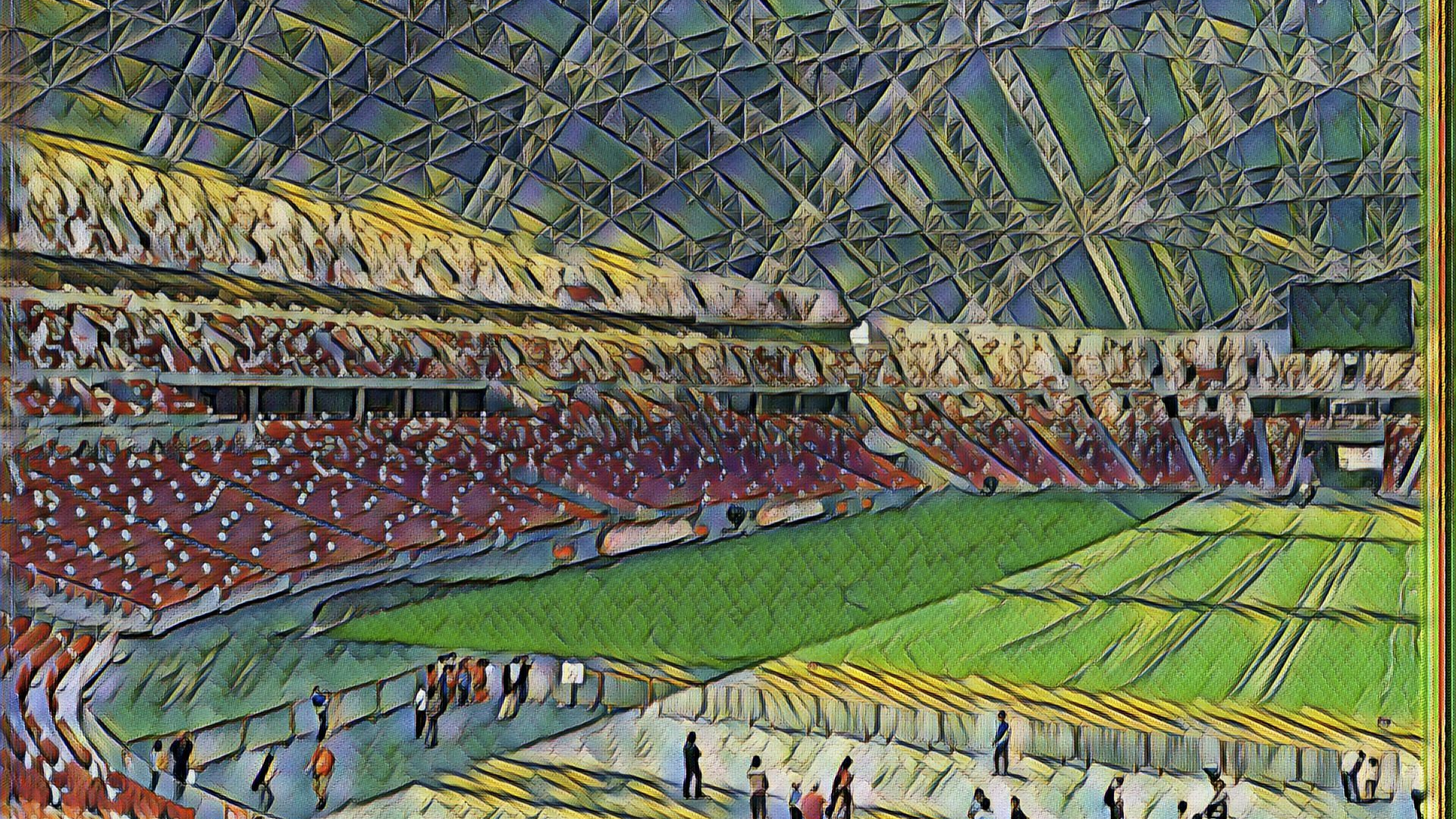


Image source - Wiki Art

Image source - Wikipedia



From: A Matlab Plugin to Visualize Neurons from Deep Models, Donglai Wei et. al.



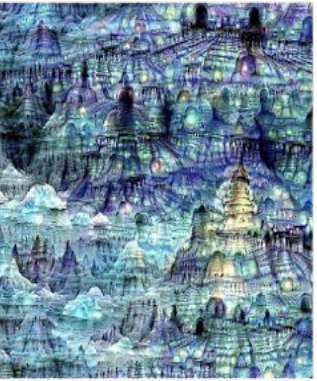
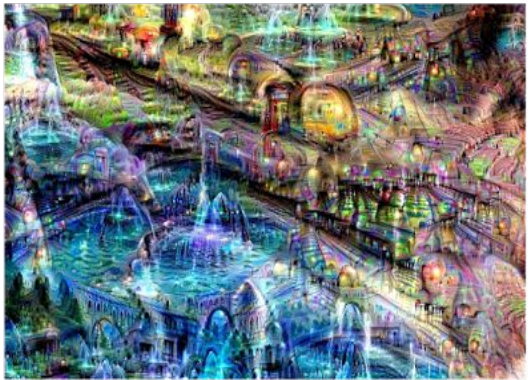
https://github.com/lengstrom/fast-style-transfer/

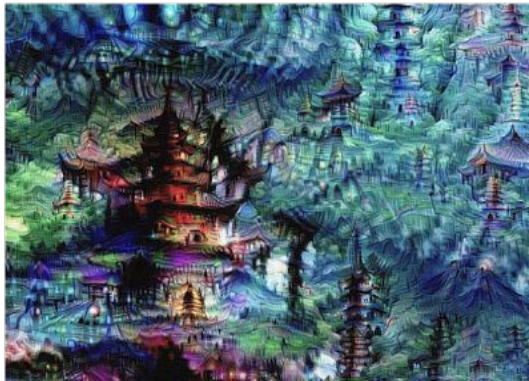
https://github.com/lengstrom/fast-style-transfer/

From https://github.com/cysmith/neural-style-tf


Fast Style Transfer
goo.gl/owF2z9
Style Transfer
goo.gl/M5hiYY





https://www.youtube.com/watch?v=DgPaCWJL7XI


goo.gl/1kBXyO

References

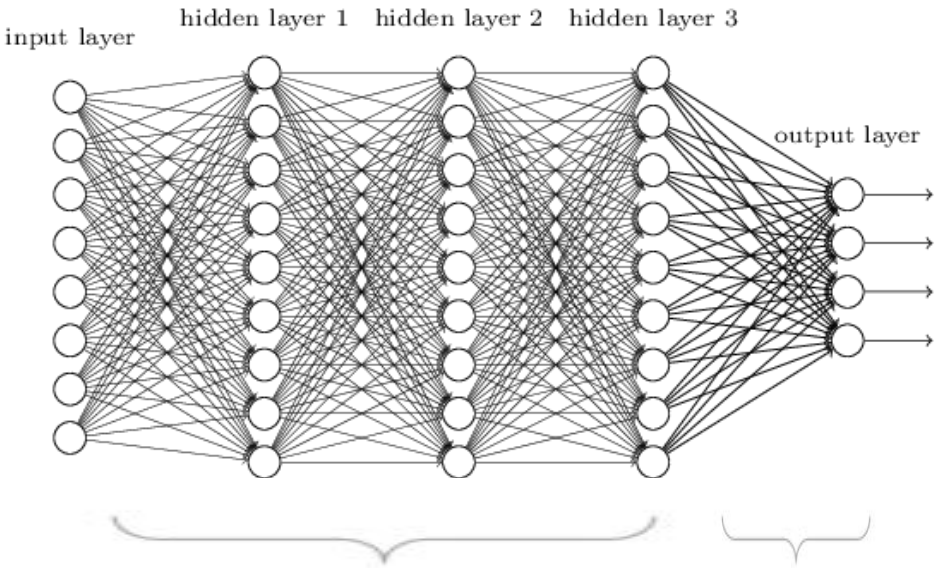
Supercharging Style Transfer
https://research.googleblog.com/2016/10/supercharging-style-transfer.html

Inceptionism: Going Deeper into Neural Networks
https://research.googleblog.com/2015/06/inceptionism-going-deeper-into-neural.html

Create your own image classifier

goo.gl/xGsB9d

Transfer Learning



Most of the network is unchanged

Only the last layer of weights and the outputs are updated

Image: neuralnetworksanddeeplearning.com

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

For more: @random_forests

Exercise #3

Fast Style Transfer
goo.gl/owF2z9

TensorFlow for Poets
goo.gl/xGsB9d

Deep Dream
goo.gl/1kBXyO

Also: Super-resolution

https://github.com/david-gpu/srez

Basic and Deep MNIST

Hello “computer vision” world

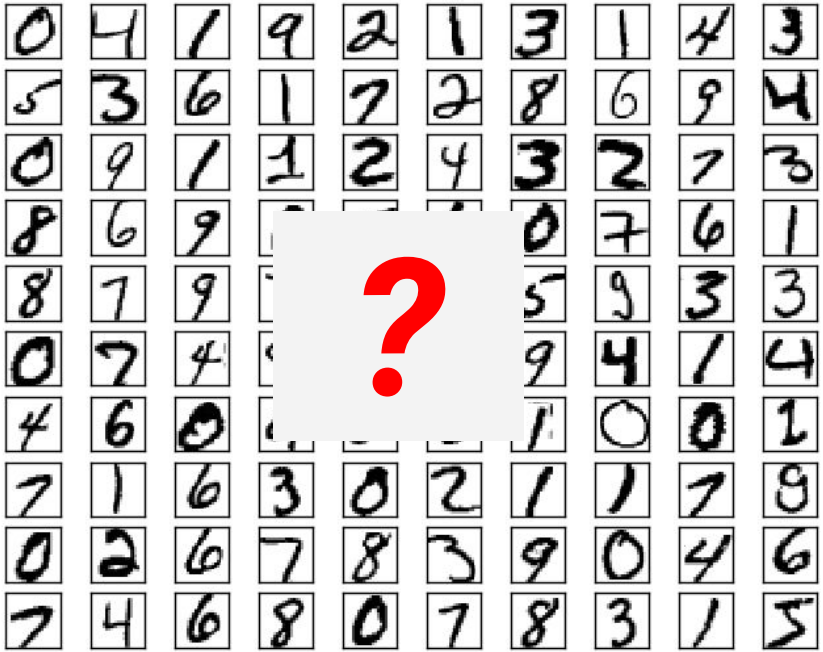
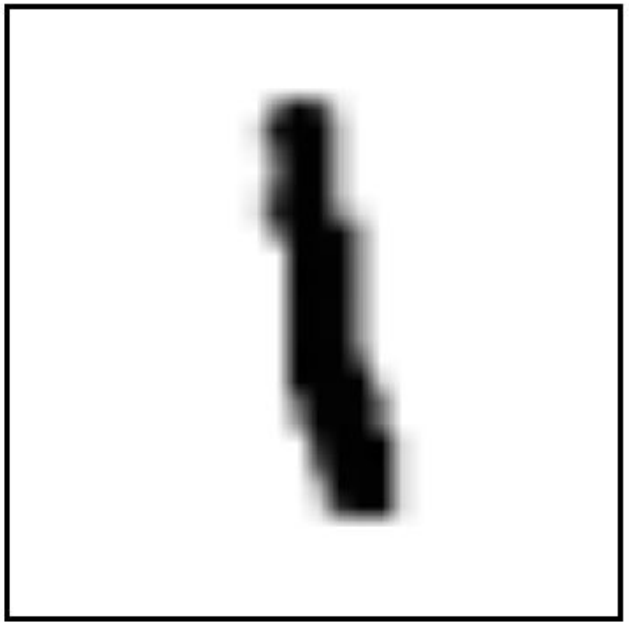


Image from https://github.com/mnielsen/neural-networks-and-deep-learning

We see



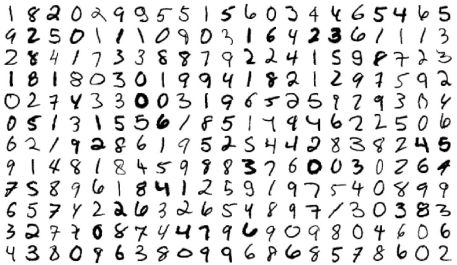
Computer “sees”

$$\approx$$

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

Dataset

Original: 65,000



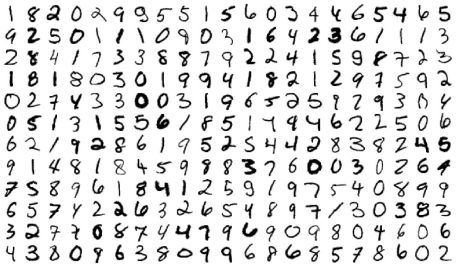
Images

Labels

5 0 4 1 9 2

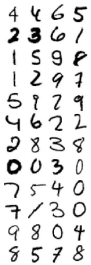
Train / test split

Original: 65,000



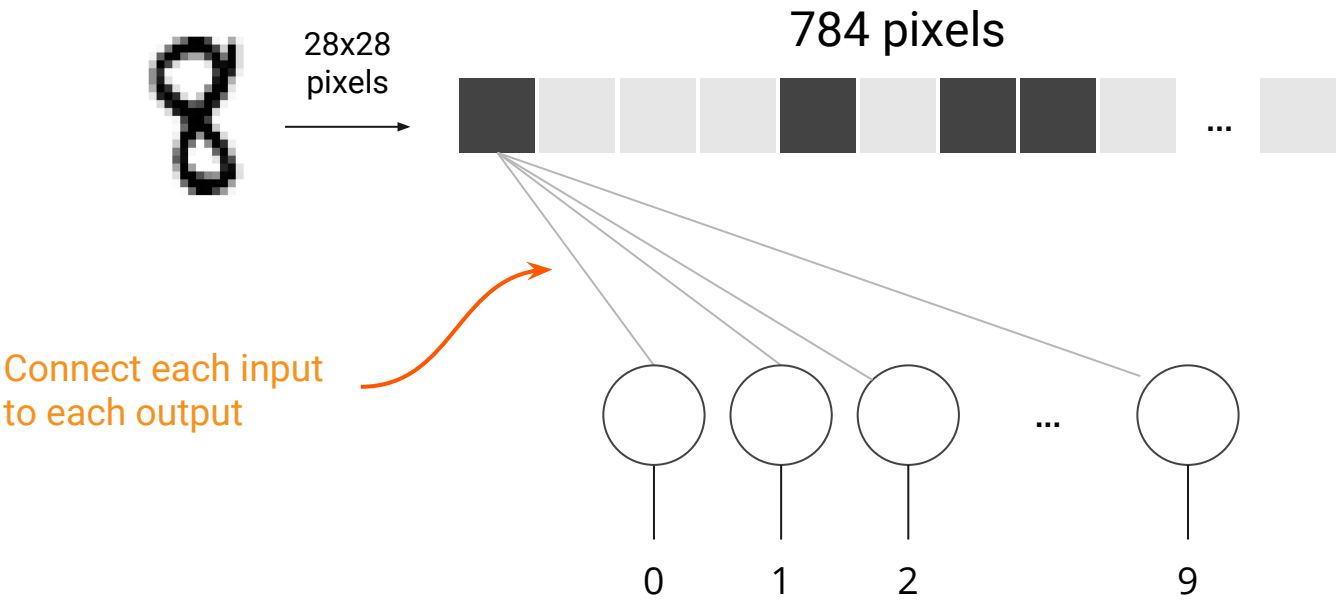
Train: 55,000

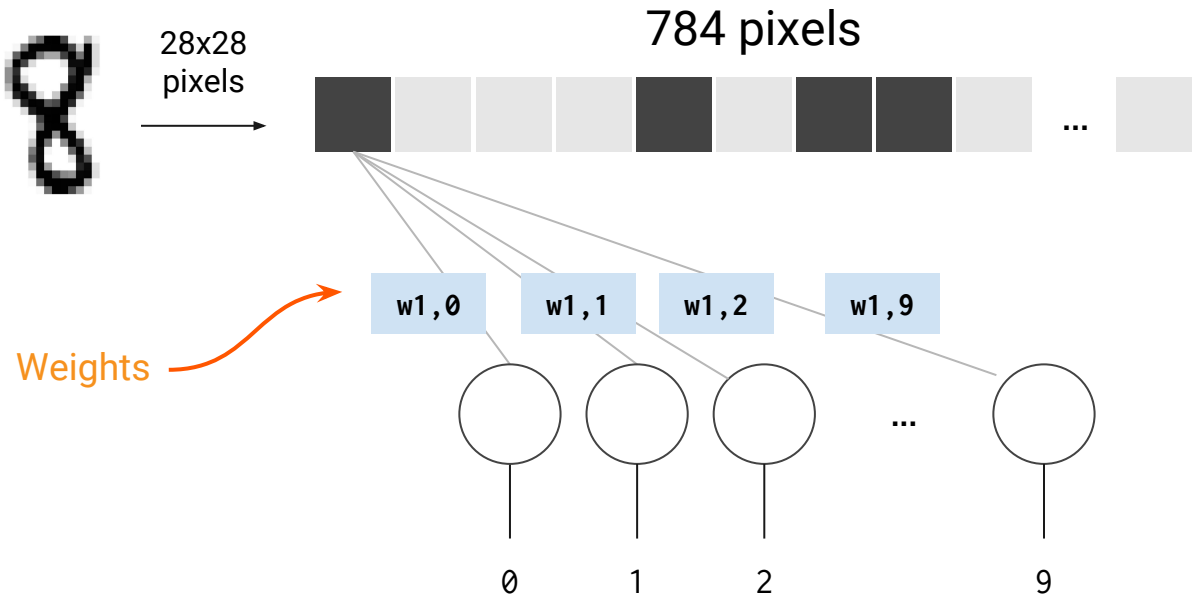
Test: 10,000

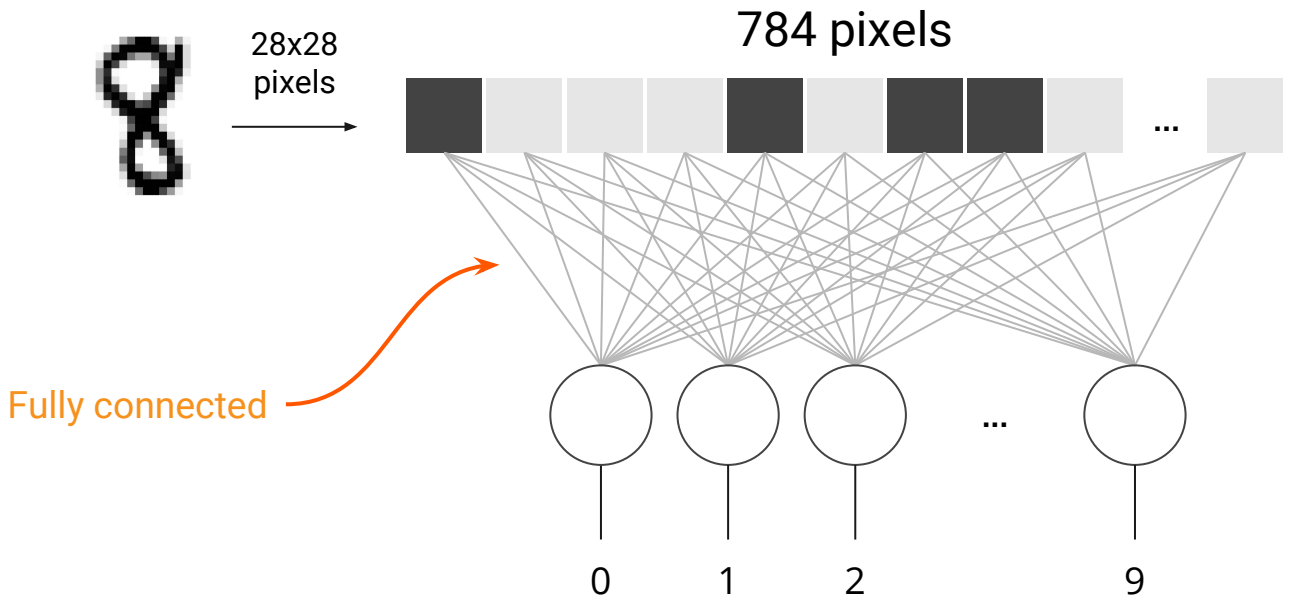


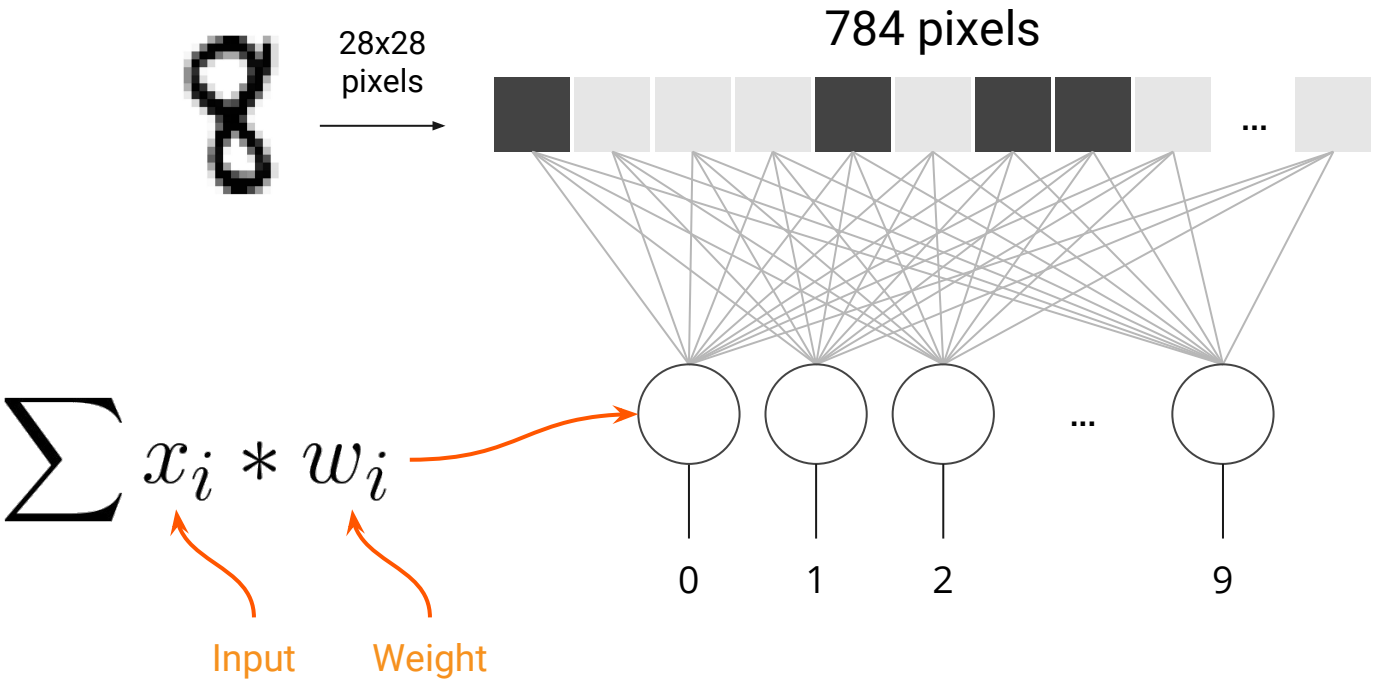
Placeholders

```
x = tf.placeholder(tf.float32, [None, NUM_PIXELS], name="pixels")

y_ = tf.placeholder(tf.float32, [None, NUM_CLASSES], name="labels")
```


28x28
pixels
784 pixels
...
Connect each input
to each output
...
0
1
2
9


28x28
pixels
784 pixels
...
w1,0
w1,1
w1,2
w1,9
Weights
...
0
1
2
9


28x28
pixels
784 pixels
...
Fully connected
...
0
1
2
9


28x28
pixels
784 pixels
...
$\sum x_i * w_i$
Input
Weight
0
1
2
...
9

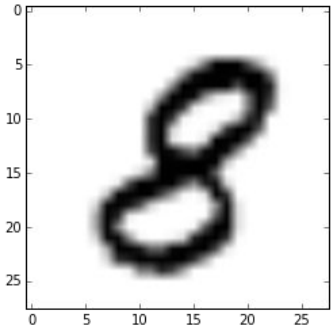
Inference...

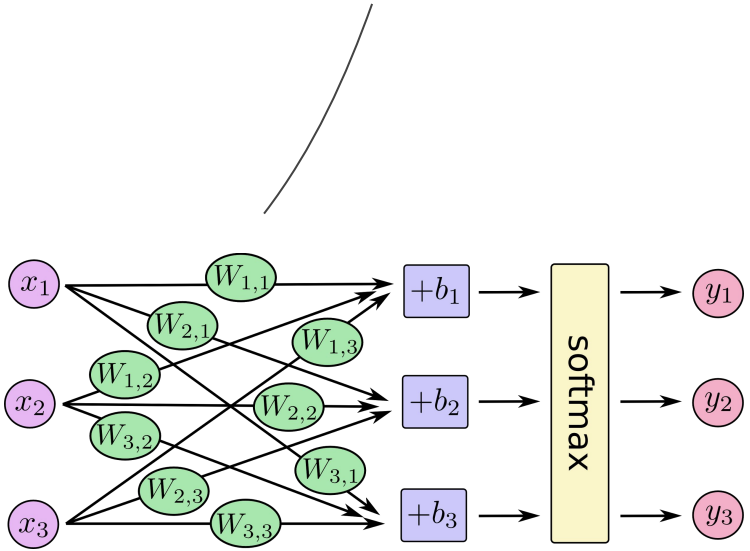
```
W = tf.Variable(tf.zeros([NUM_PIXELS, NUM_CLASSES]), name="weights")

b = tf.Variable(tf.zeros([NUM_CLASSES]), name="biases")

y = tf.matmul(x, W) + b
```

Softmax

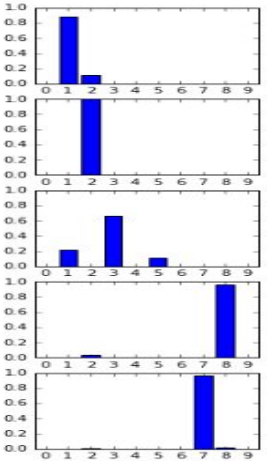
784 weights for each digit, so 784 x 10 total connections

input vector
(pixel data)

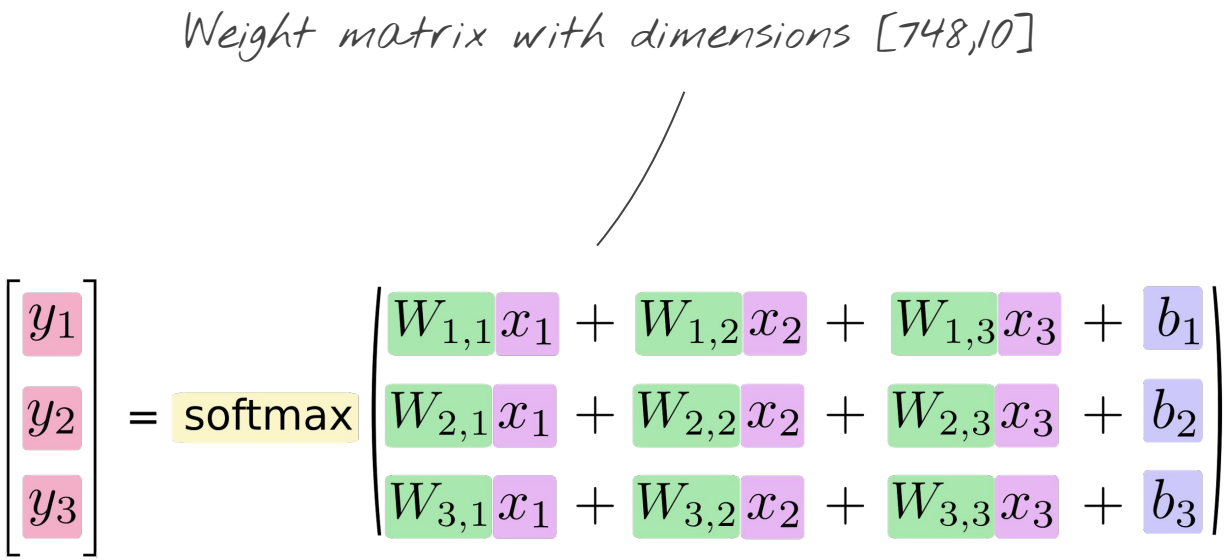




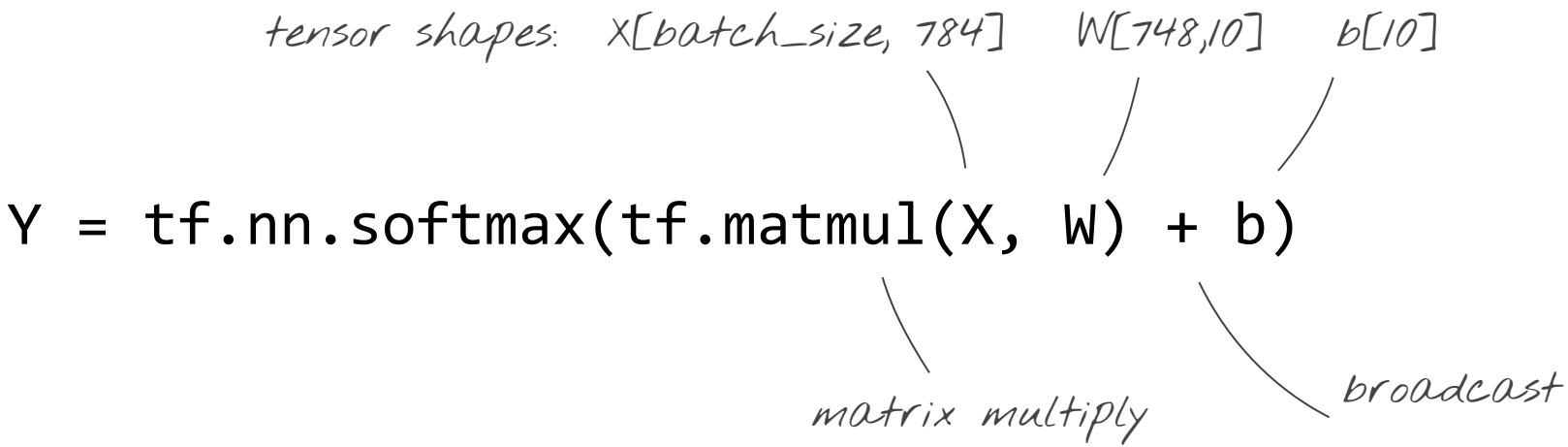
output vector
(probability of
each digit)



Written as a matrix multiply



In TensorFlow



Loss

```
loss =
   tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(y, y_))
```

Optimization

```
train_step =
   tf.train.GradientDescentOptimizer(LEARNING_RATE).minimize(loss)
```

Training with batches

```
for i in range(TRAIN_STEPS):
    batch_xs, batch_ys = mnist.train.next_batch(BATCH_SIZE)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

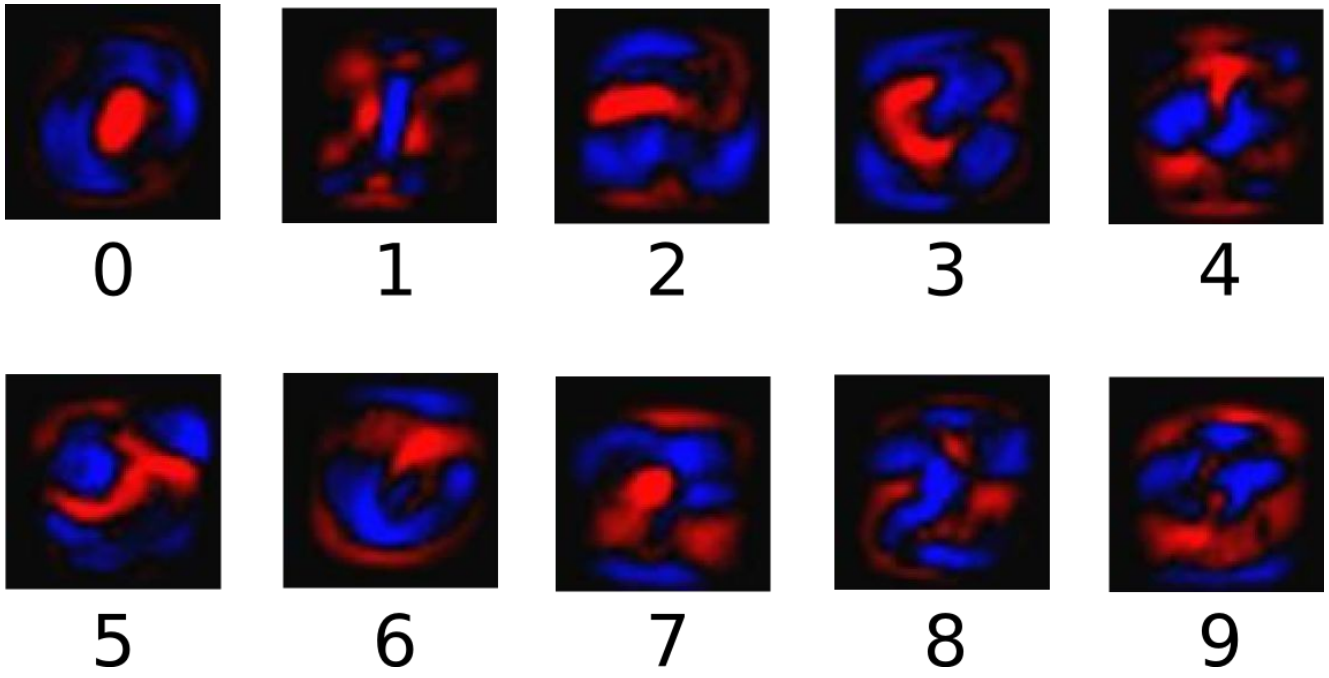
Evaluation

```
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

print("Accuracy %f" % sess.run(accuracy,
    feed_dict={x:  mnist.test.images, y_: mnist.test.labels}))
```

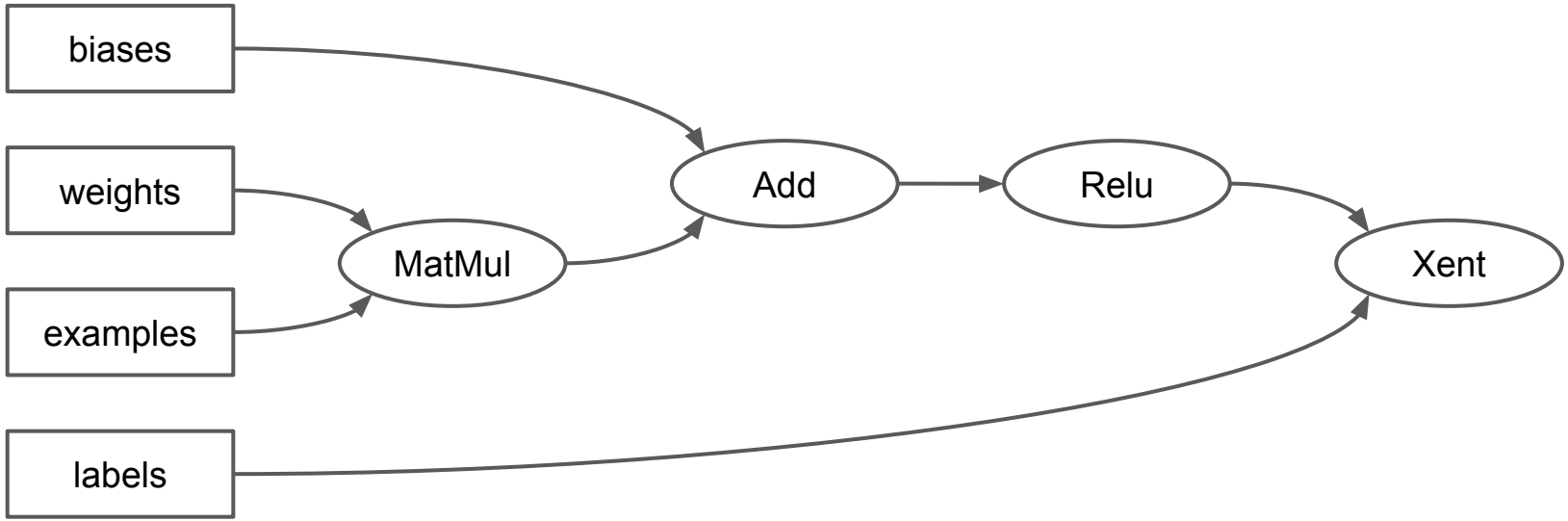
3_basic_mnist.ipynb

goo.gl/nrdsxM

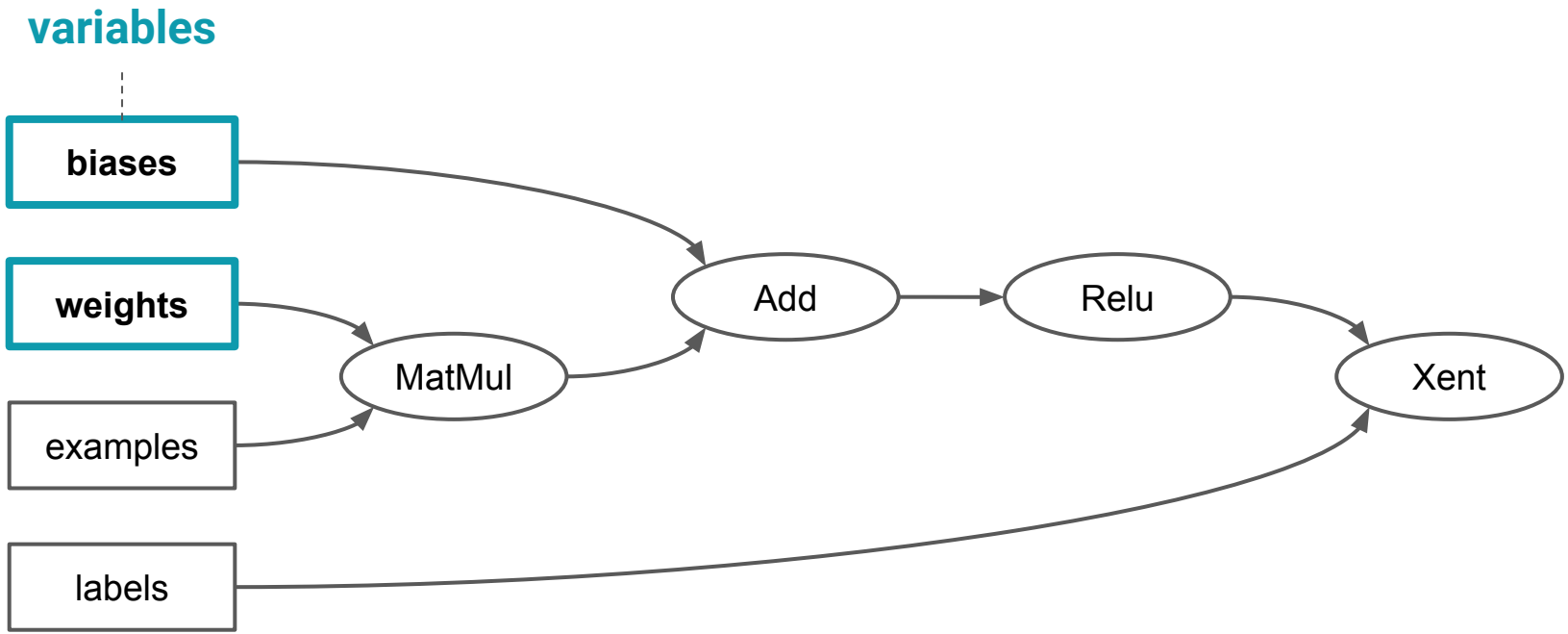
Visualizing the learned weights



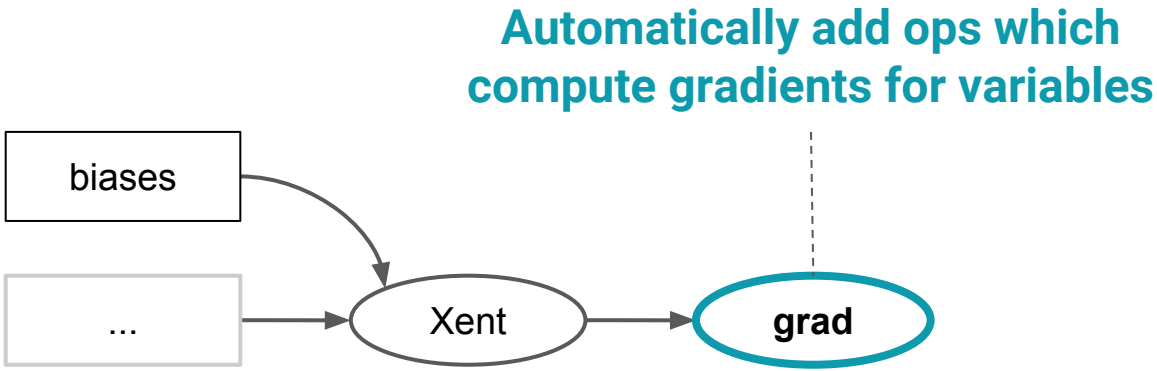
Graphs can be processed, compiled, remotely executed, assigned to devices.



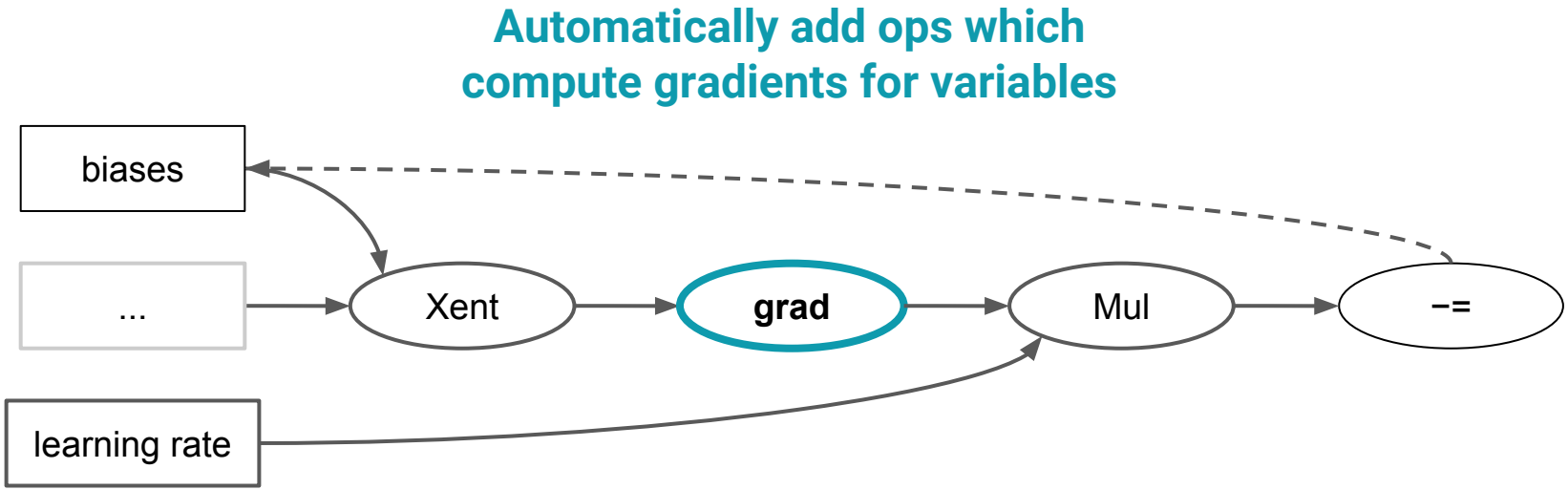
Data flow, with state



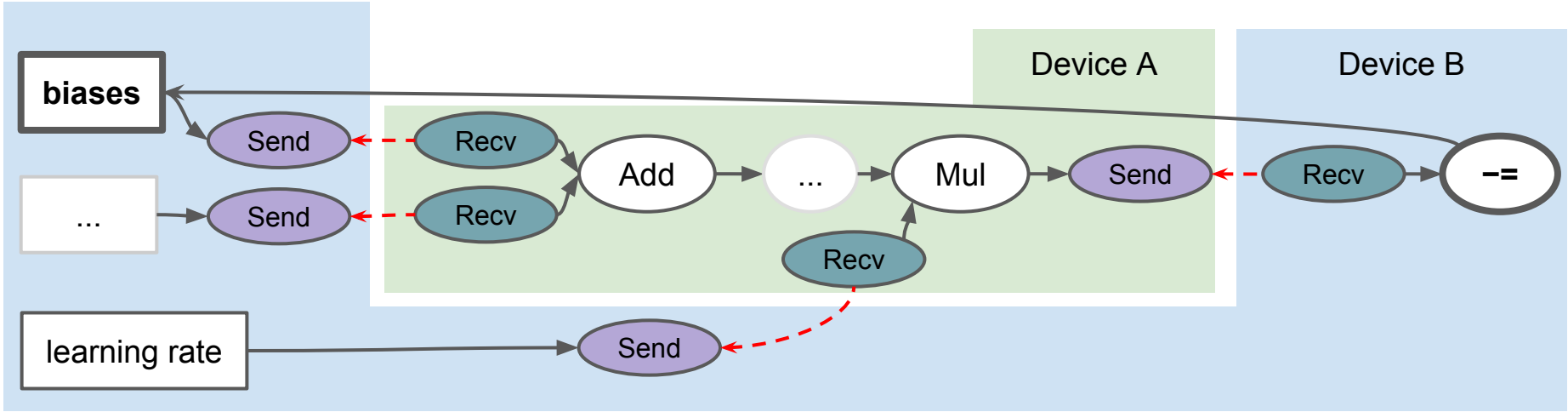
Automatic Differentiation



Automatic Differentiation

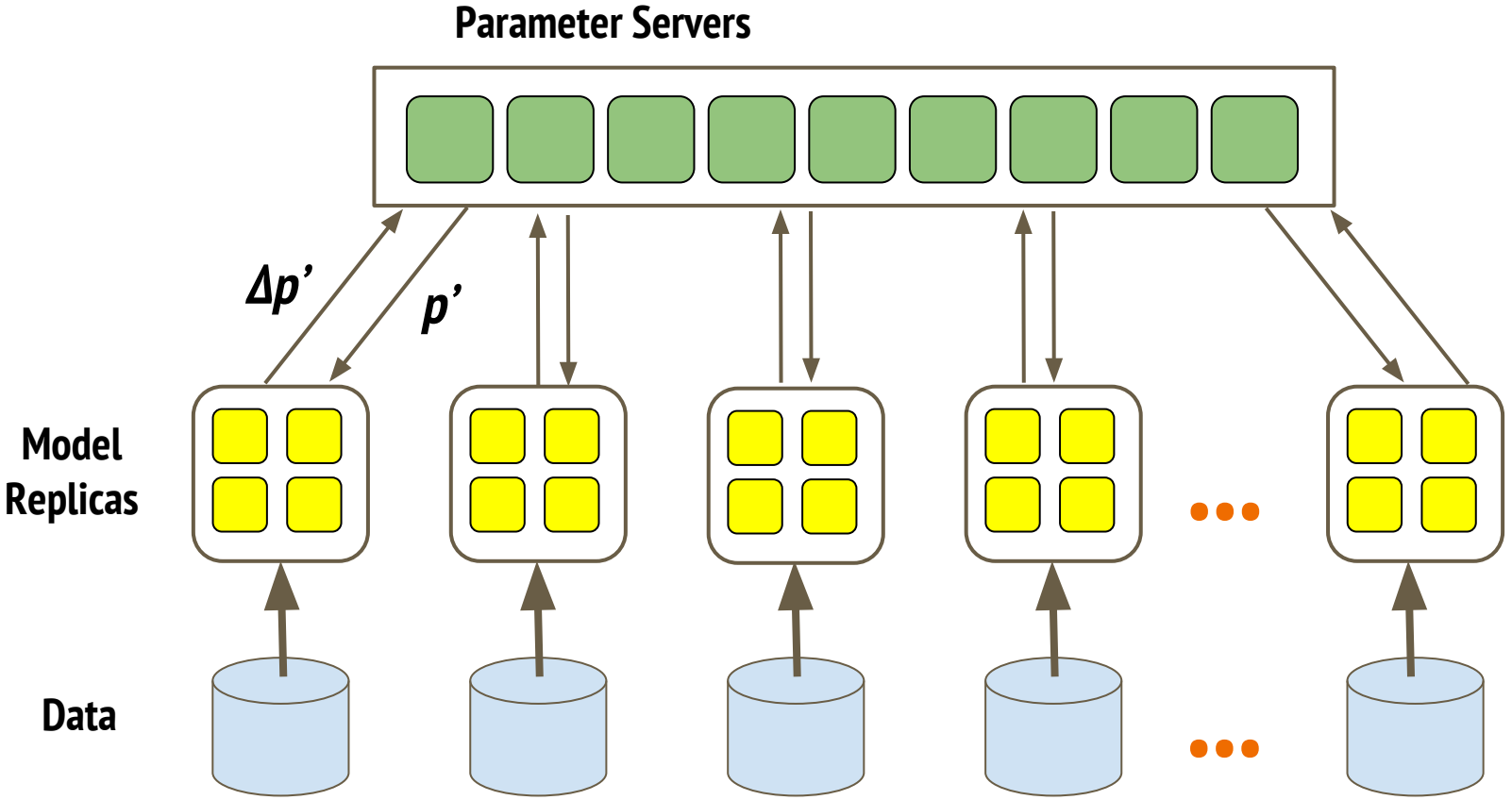


Distributed

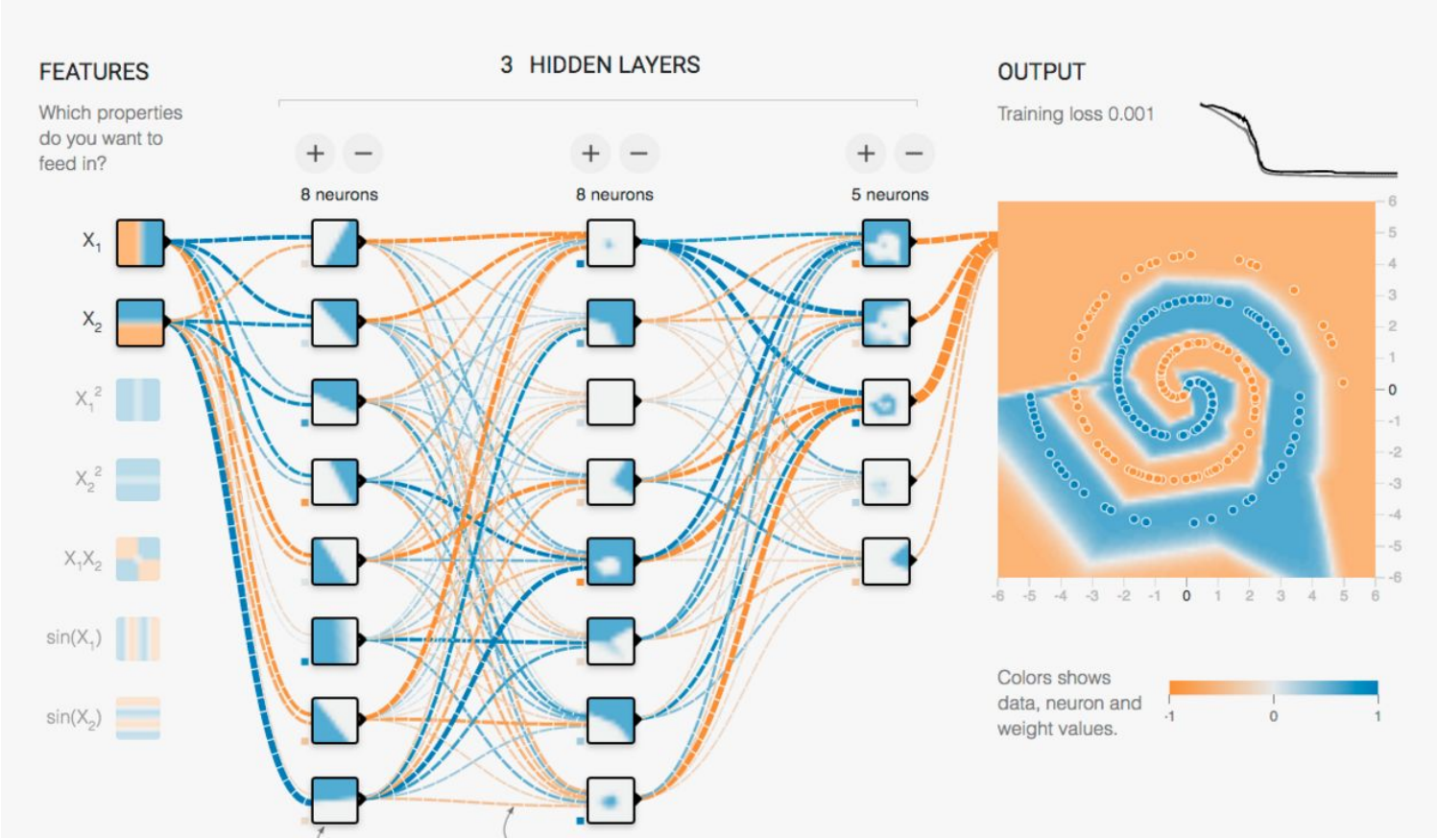


Send / receive nodes added automatically
Devices: Processes, Machines, CPUs, GPUs, TPUs, etc

Data Parallelism



Going Deeper

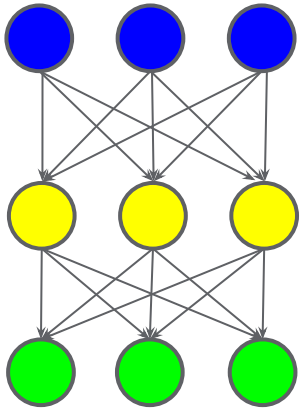


Hidden layers = Hierarchies of features

Demo: TensorFlow Playground

goo.gl/mXhncM

Hidden layers



```
weights1 = weight_variable(NUM_PIXELS, HIDDEN1_UNITS)

biases1 = bias_variable(HIDDEN1_UNITS)

hidden1 = tf.nn.relu(tf.matmul(x, weights1) + biases1)

weights2 = weight_variable(HIDDEN1_UNITS, NUM_CLASSES)

biases2 = bias_variable(NUM_CLASSES, "biases2")

y = tf.matmul(hidden1, weights2) + biases2
```

4_deep_mnist.ipynb

goo.gl/nrdsxM

Higher level APIs

Higher level APIs

tf.contrib.learn

https://www.tensorflow.org/versions/r0.12/tutorials/tflearn/index.html

tf.slim

https://research.googleblog.com/2016/08/tf-slim-high-level-library-to-define.html

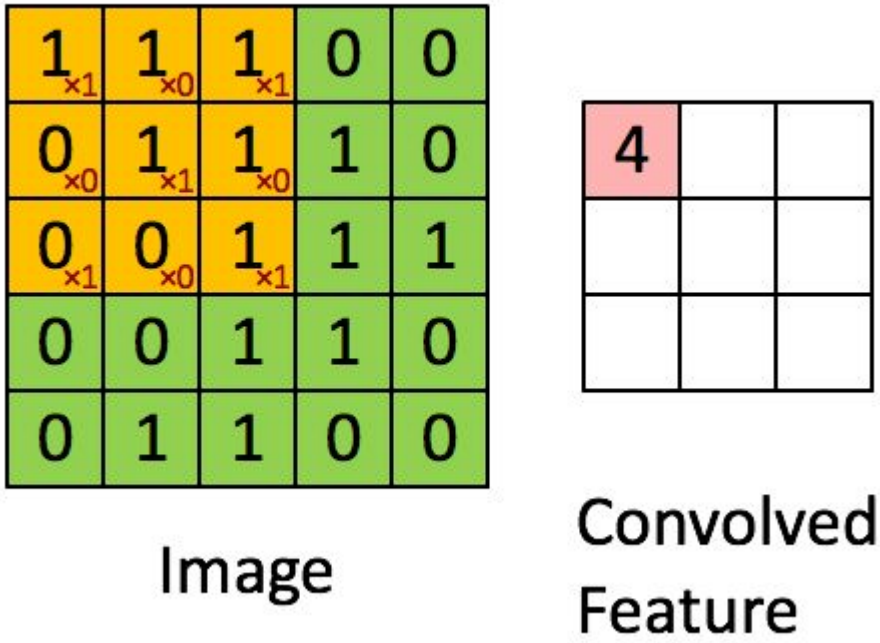
Keras

https://keras.io/

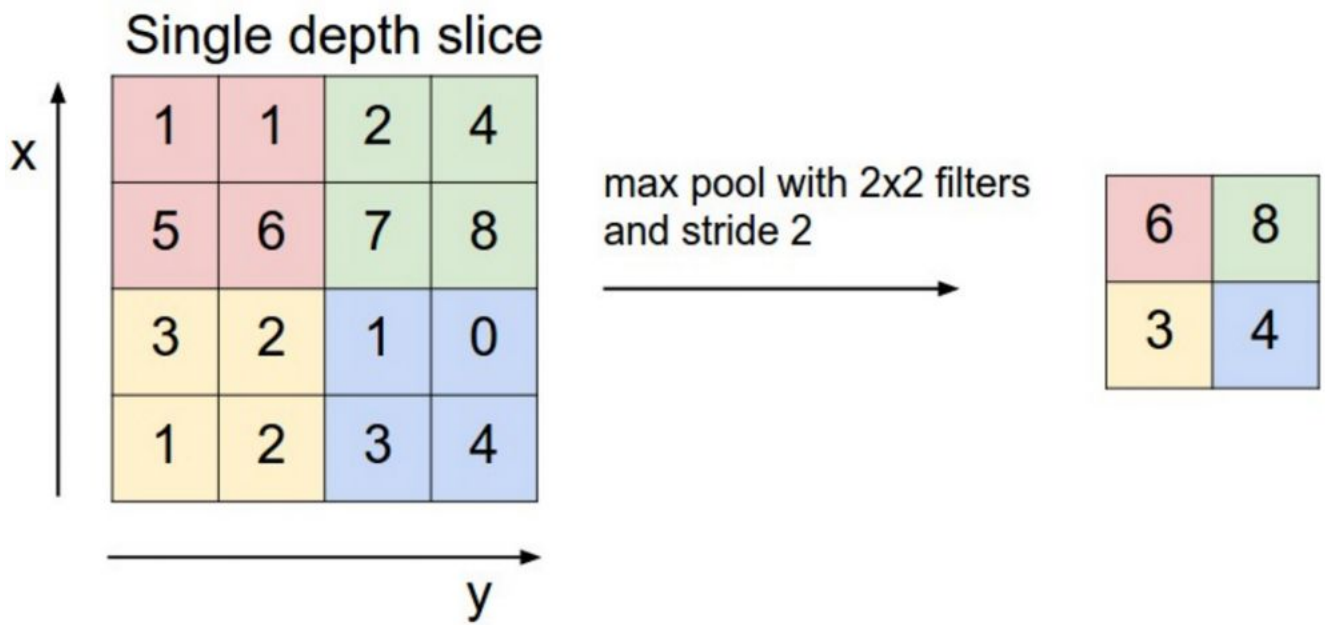
5_mnist_tf_contrib_learn.ipynb

goo.gl/nrdsxM

Convolution

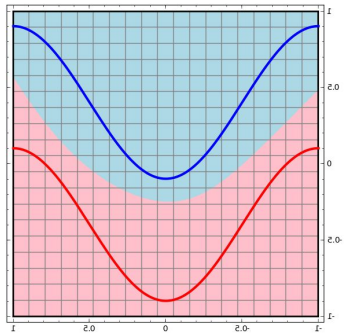


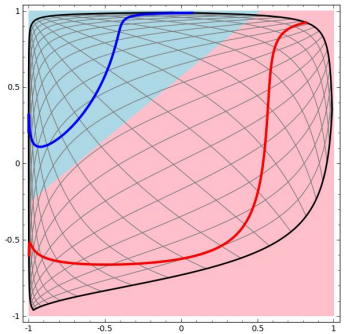
Convolution with 3×3 Filter. Source:
http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

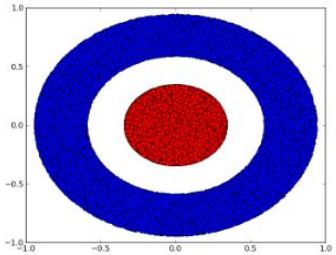


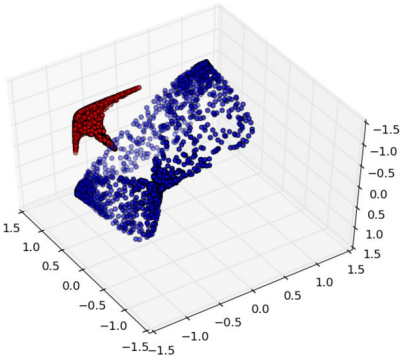
Max pooling in CNN. Source: http://cs231n.github.io/convolutional-networks/#pool

Learn more: colah.github.io









From: Neural Networks, Manifolds, and Topology, colah's blog

Learn more: 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

Progression and caveats

- Linear ~90%
- Two-layer DNN ~96%
- ConvNet ~99% (close to the state of the art on this toy problem)

Should you focus on accuracy when learning ML?

- Nope. What matters is designing a proper experiment.

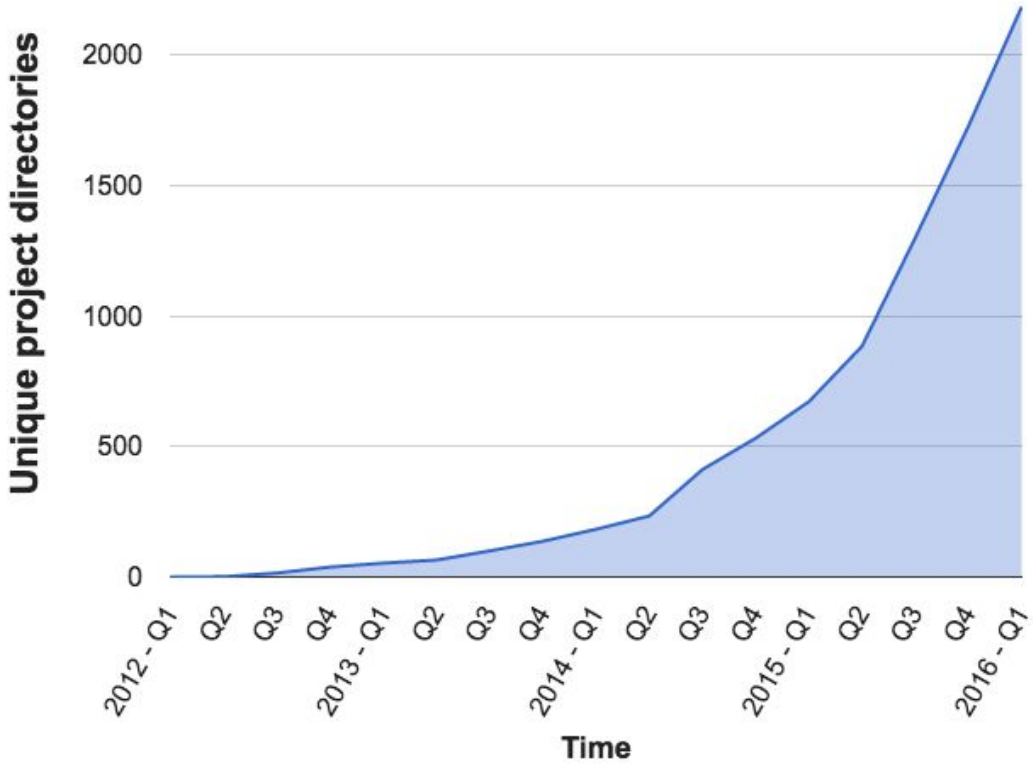
6_conv_mnist.ipynb

goo.gl/nrdsxM

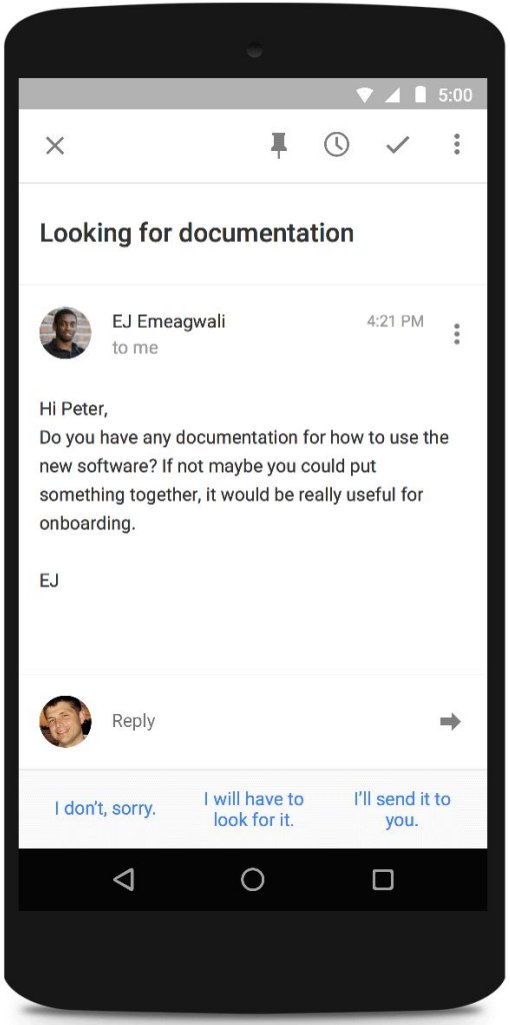
Deep Learning at Google

Growing Use of Deep Learning at Google

of directories containing model description files




EXIT

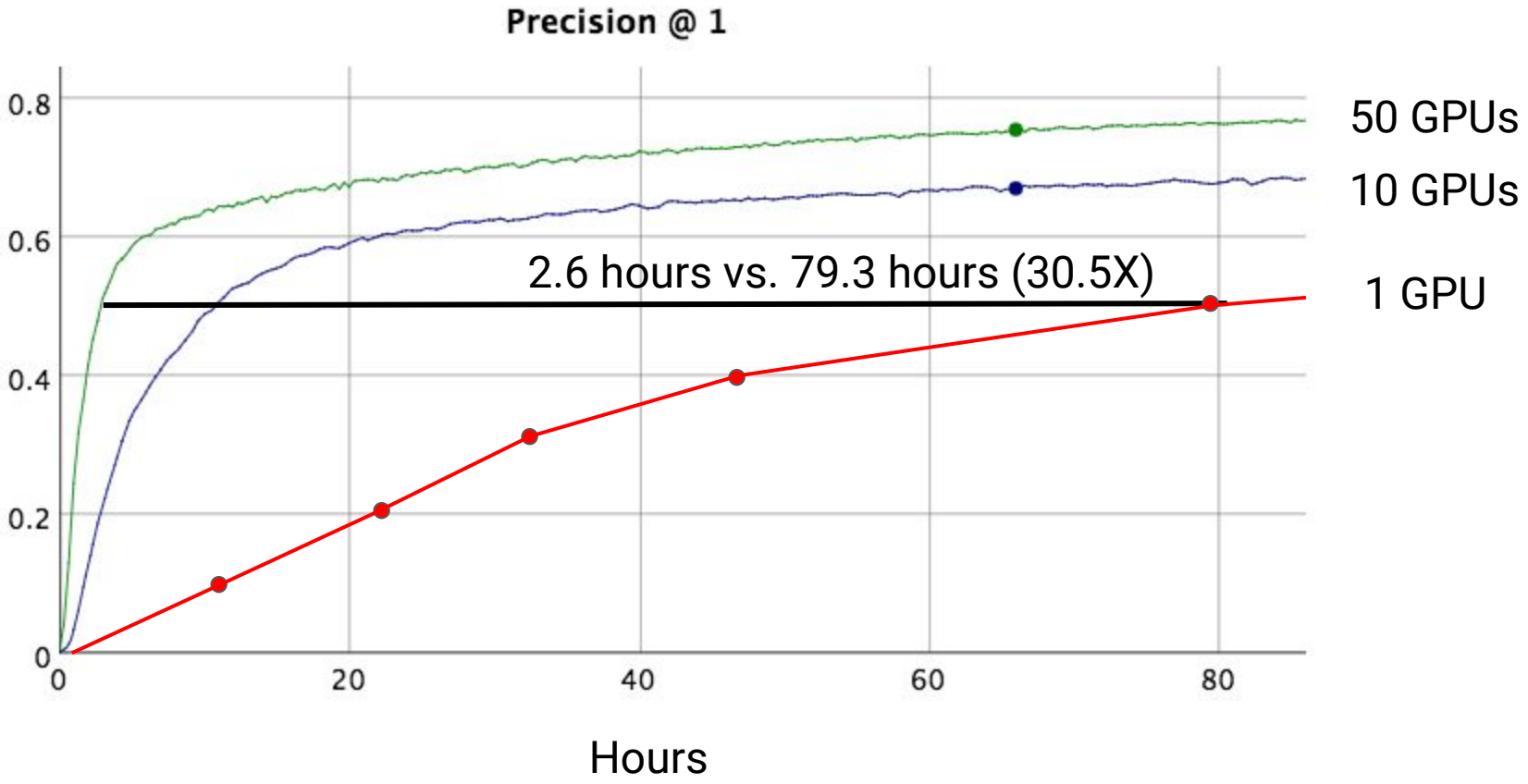


Smart reply in Inbox by Gmail

10%

of all responses sent on mobile

Image Model (Inception) Synchronous Training



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

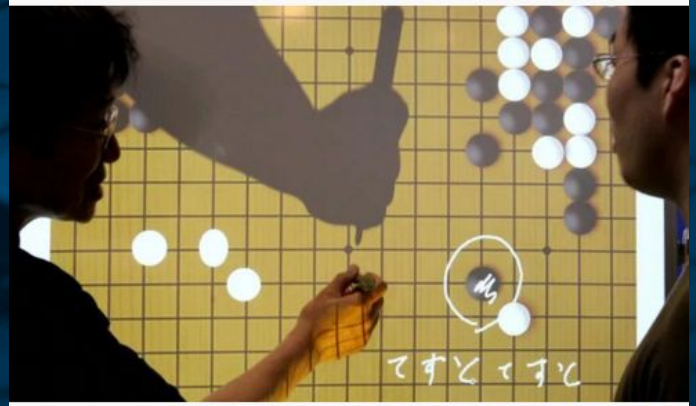
How did they do it?

What is the game Go?

Facebook trains AI to beat humans at Go

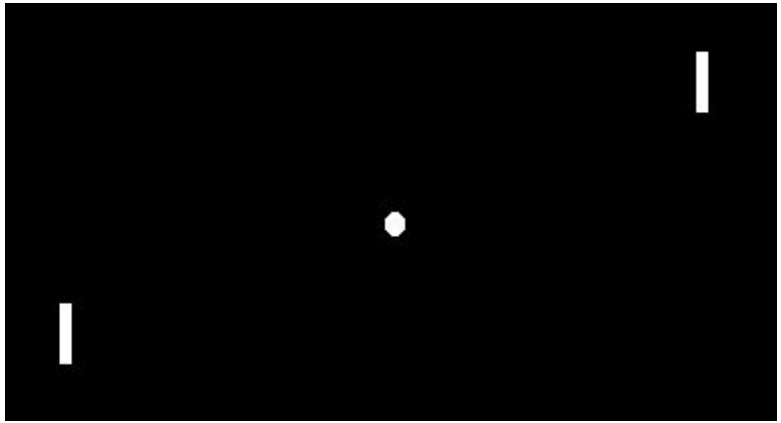
Google's AI just cracked the game that supposedly no computer could beat

By Mike Murphy | January 27, 2016



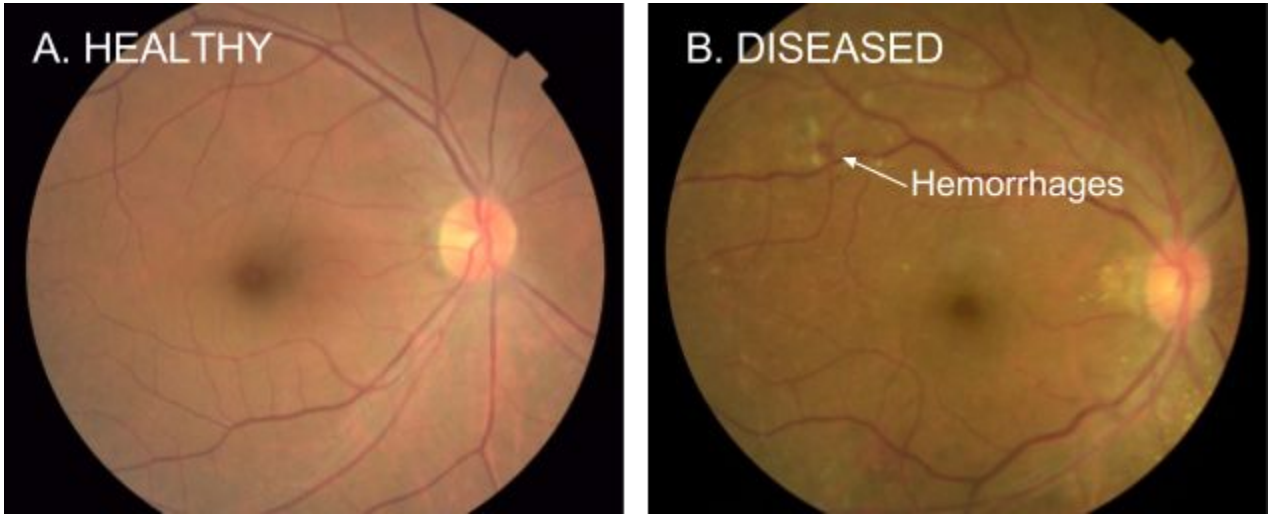
Going up. (Reuters/Kiyoshi Ota)

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.



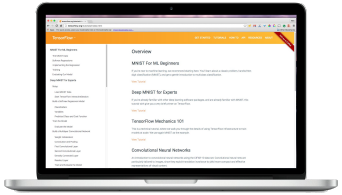
https://deepmind.com/research/dqn/

Detection of Diabetic Eye Disease



https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html

Next steps



Tutorials and code
tensorflow.org

Totally new to ML?
Recipes goo.gl/KewA03

Intro to Deep Learning with TensorFlow
Udacity class goo.gl/iHssII

Stanford's CS231n
cs231n.github.io

Udacity's Machine Learning Nanodegree
goo.gl/ODpXj4

These slides + exercises
goo.gl/nrdsxM

TensorFlow Playground
goo.gl/mXhncM

Chris Olah's blog
Colah.github.io

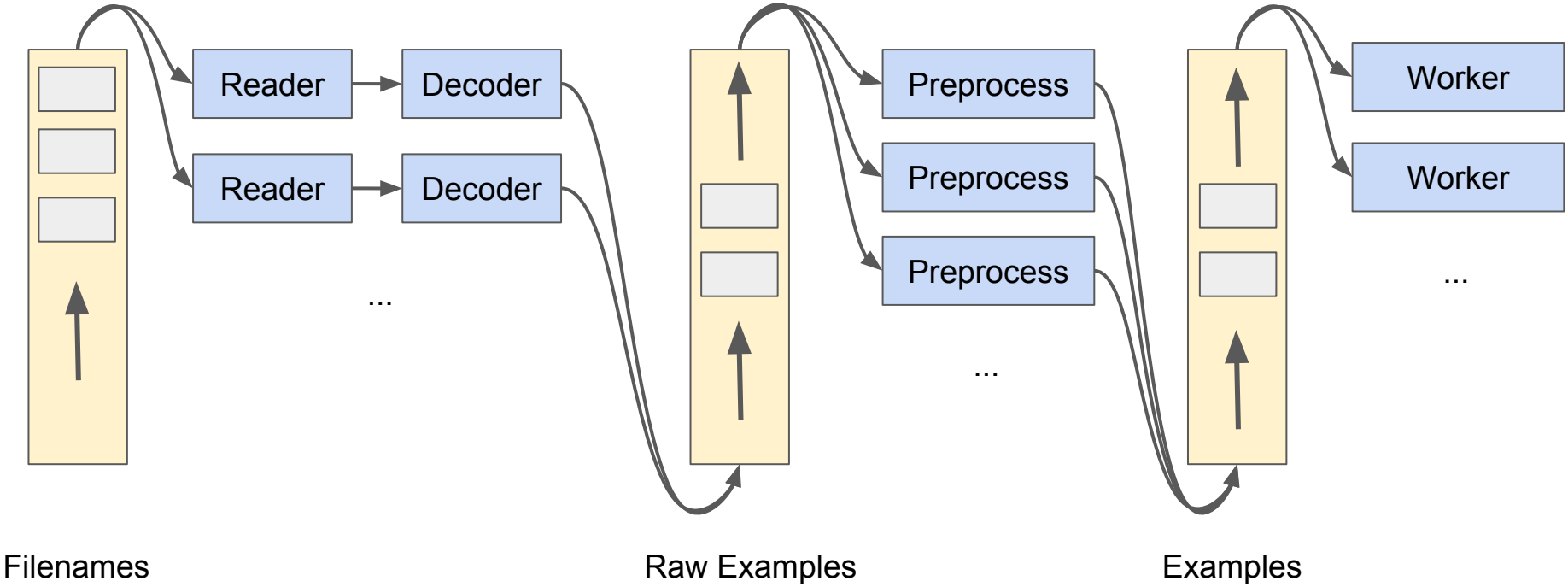
Michael Nielsen's book
neuralnetworksanddeeplearning.com

Thank you and have fun!

Josh Gordon
@random_forests

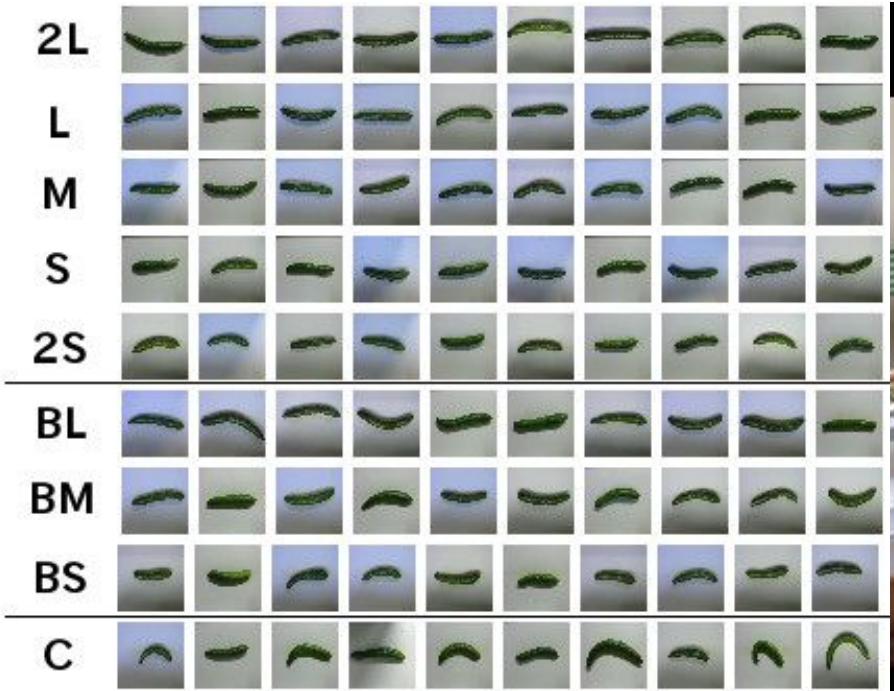
Wolff Dobson
@greenexec

Input Pipelines with Queues



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

TensorFlow powered Cucumber Sorter



From: http://workpiles.com/2016/02/tensorflow-cnn-cucumber/

Questions?