

Intro to TensorFlow

SHI WENXUAN

Tel: 13920561100

Email: shiwx@nankai.edu.cn



扫一扫上面的二维码图案，加我微信

Agenda

Part I

- TensorFlow

Part II

- Linear regression

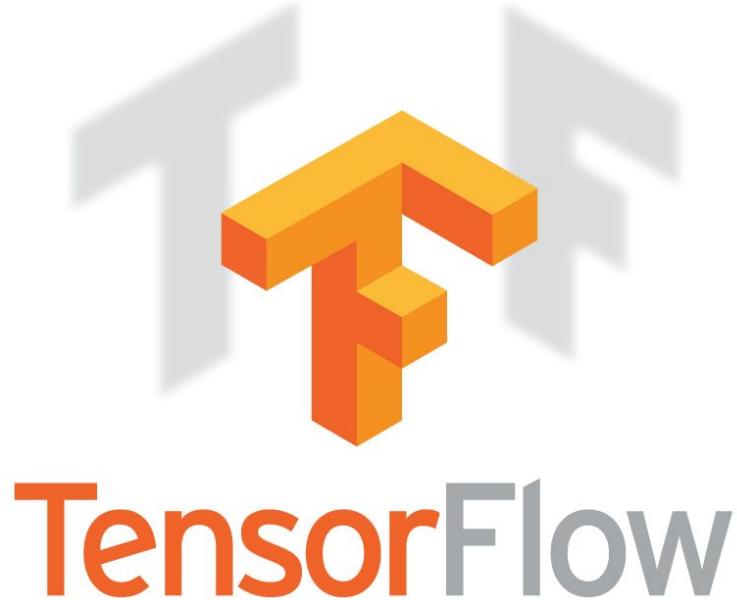
Part III

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

Part IV

- Basic and Deep MNIST

TensorFlow



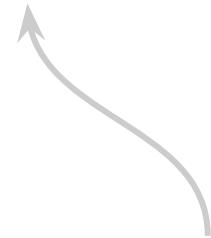
- Fast, flexible, and scalable open-source machine learning library
- For research and production
- Runs on CPU, GPU, Android, iOS, Raspberry PI
- Apache 2.0 license

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

A multidimensional array.



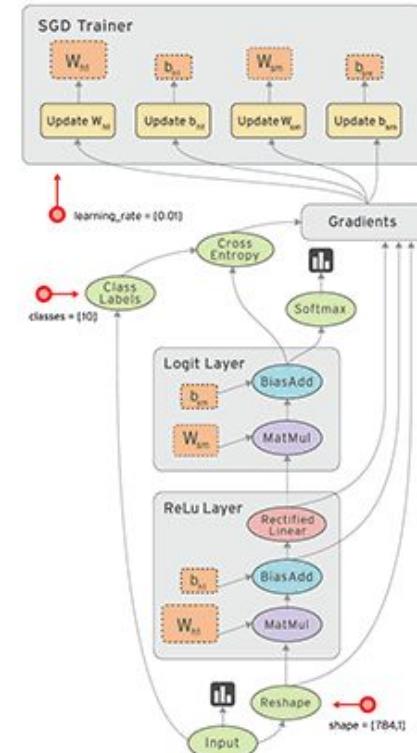
TensorFlow



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

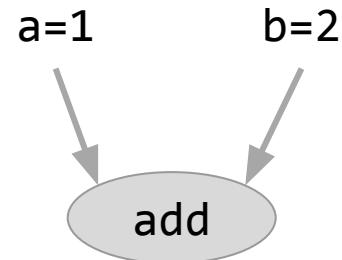


The background features a 3D wireframe landscape graphic rendered in white against a solid orange background. The wireframe consists of a grid of lines forming a series of hills and valleys, creating a sense of depth and perspective. The lighting is soft, emphasizing the ridges and the overall organic shape of the terrain.

But, how?

Build a graph...

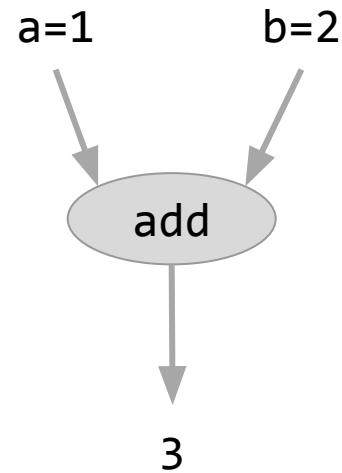
```
import tensorflow as tf  
a = tf.constant(1)  
b = tf.constant(2)  
c = tf.add(a, b)
```



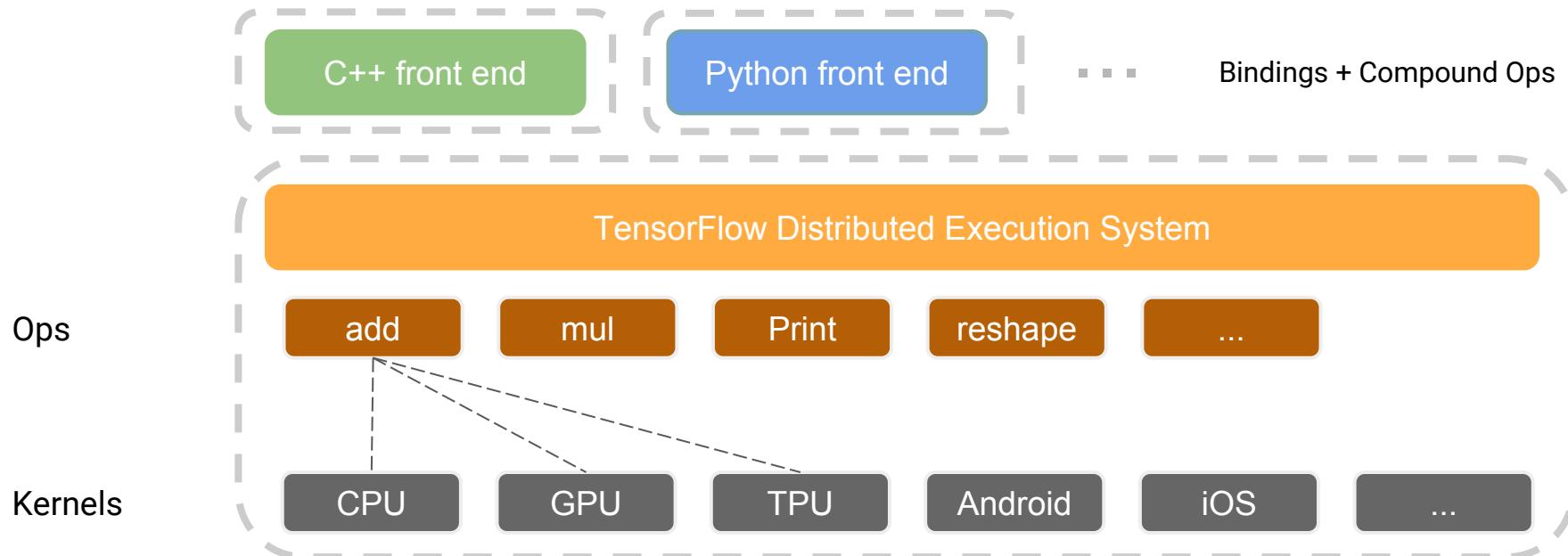
... then run it

```
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)
```



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))
```

Exercise #1: Fibonacci Sequence

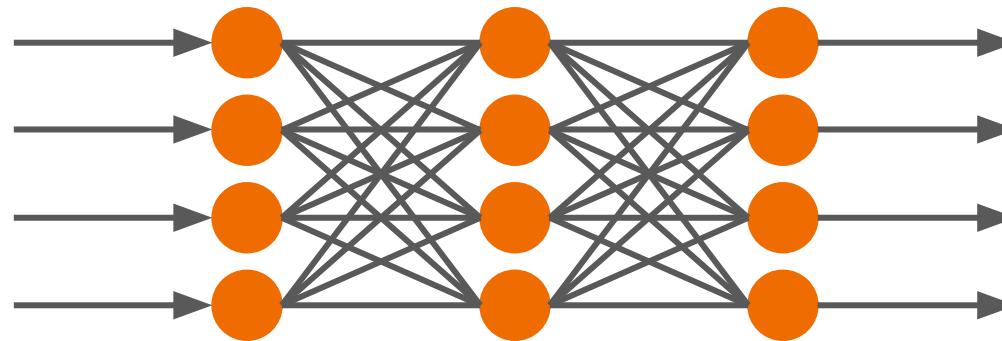
1_warm_up.ipynb

goo.gl/nrdsxM

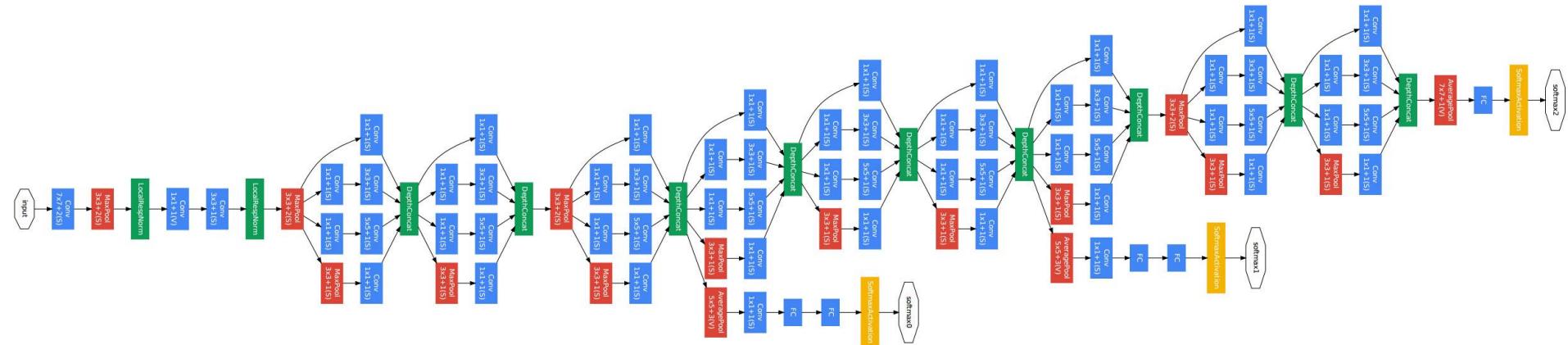


But why?

Neural Networks, yesterday



Remember Inception?



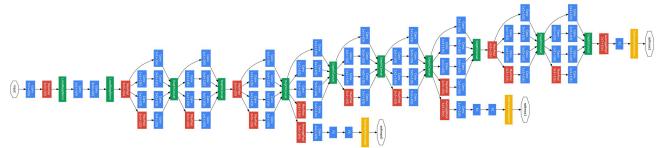
Machine Learning gets complex quickly



Heterogenous
systems



Distributed
systems



Modeling complexity

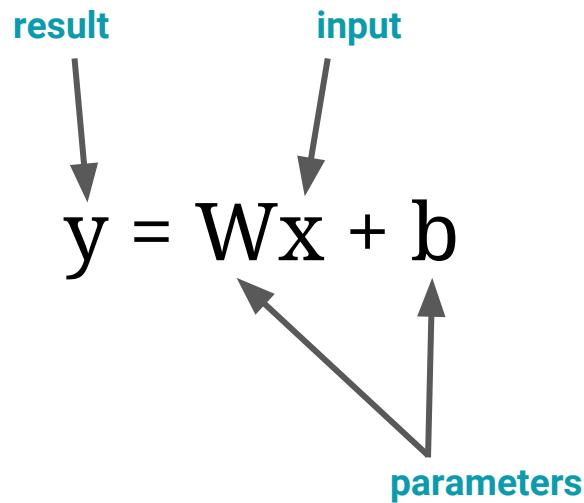
Linear Regression

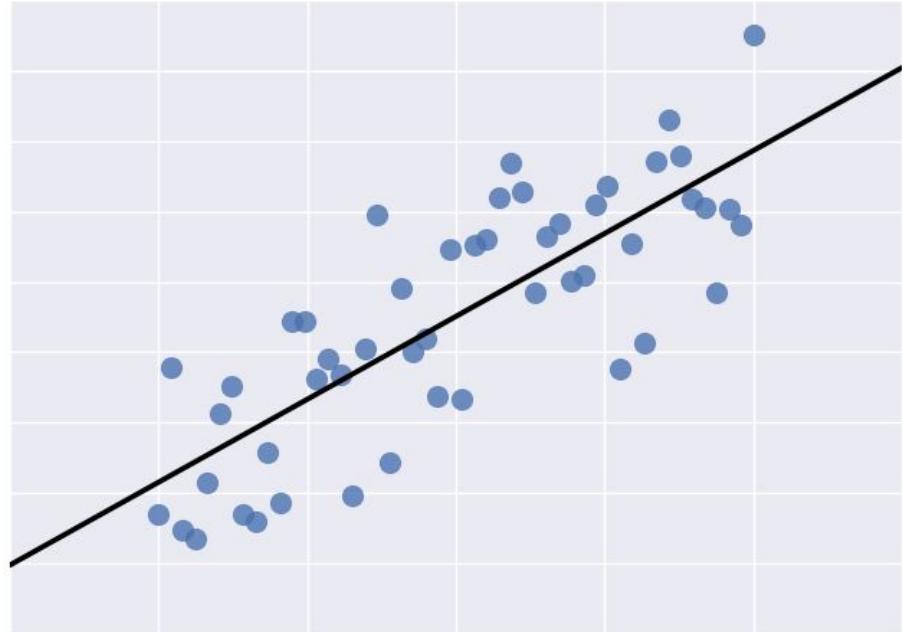
Inference, Loss, Optimize

Linear Regression

$$y = Wx + b$$

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) 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

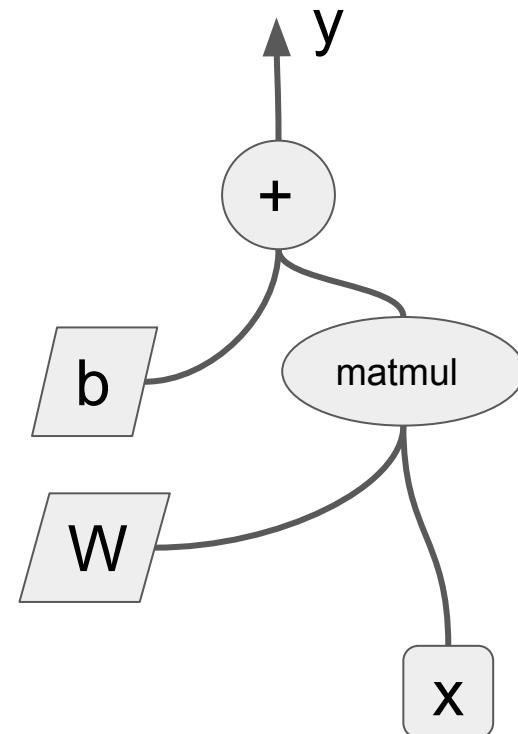
x = tf.placeholder(shape=[None],
                    dtype=tf.float32, name="x")

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

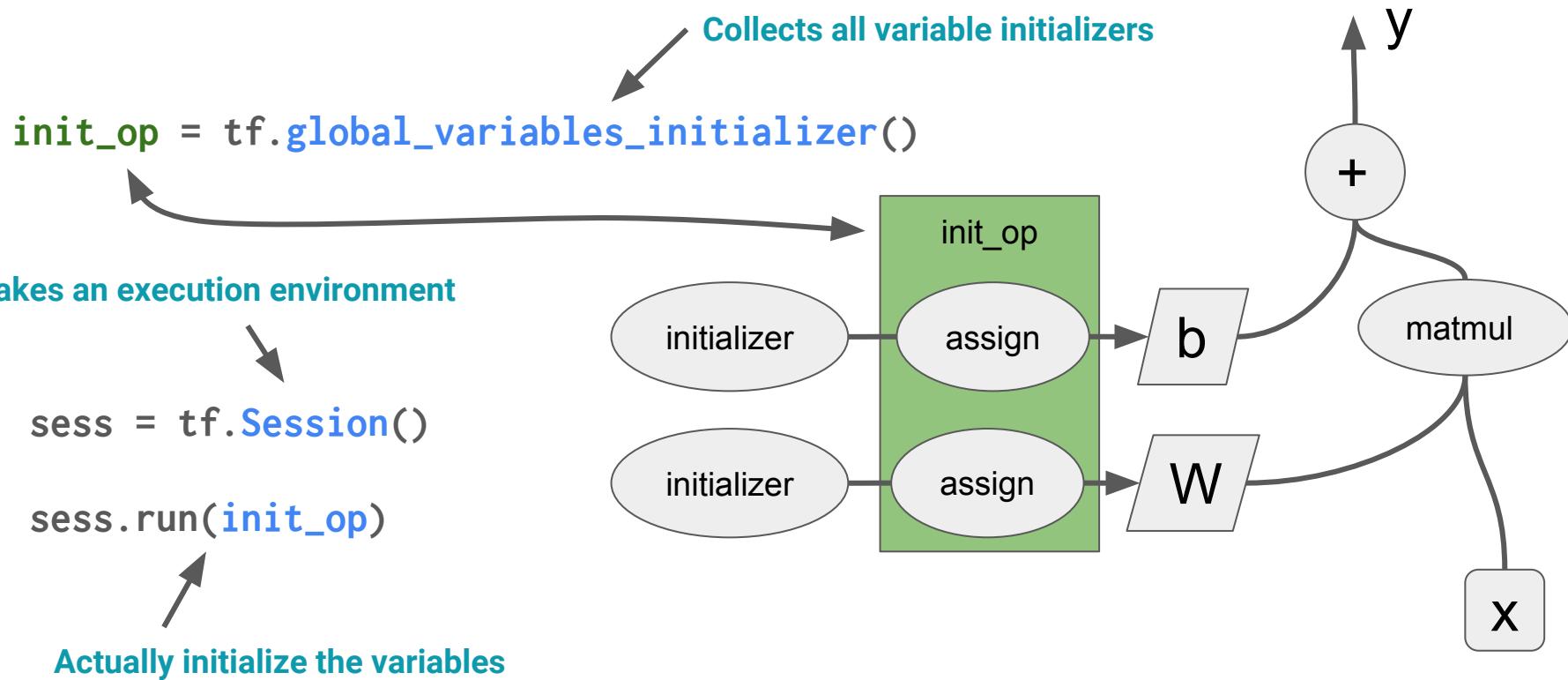
b = tf.Variable(tf.random_normal([1], name="b"))
```

$y = Wx + b$ 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

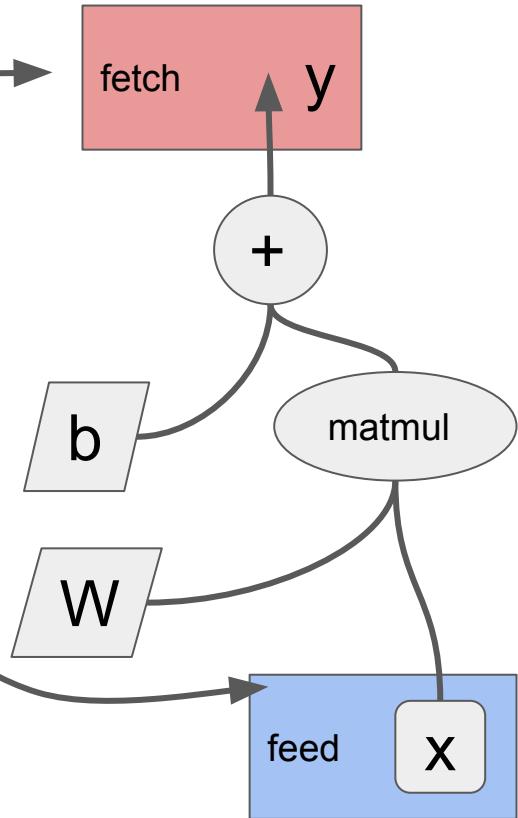


Running the Computation

`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}))
```

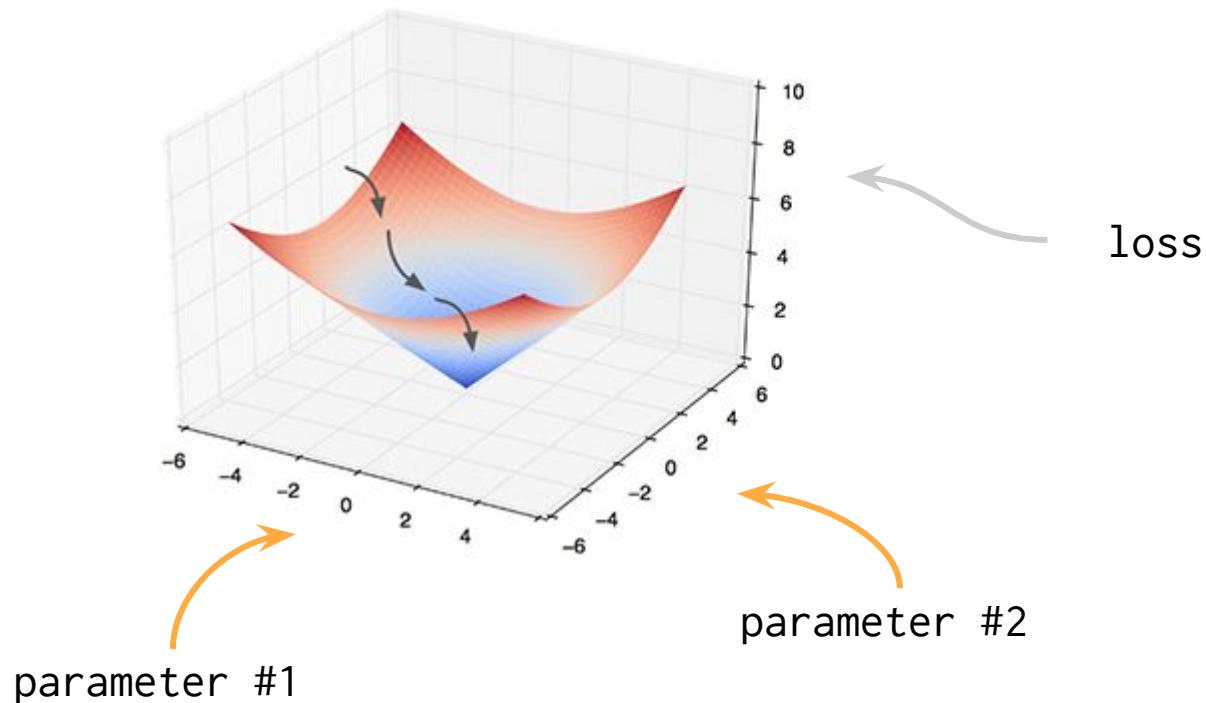
Build the graph

Prepare execution env

Initialize variables

Run the computation

Concepts: Loss and Gradient Descent



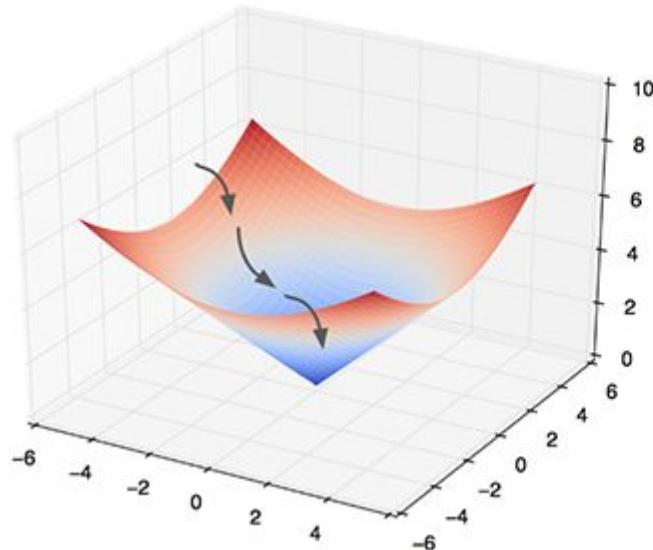
Define a Loss

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

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

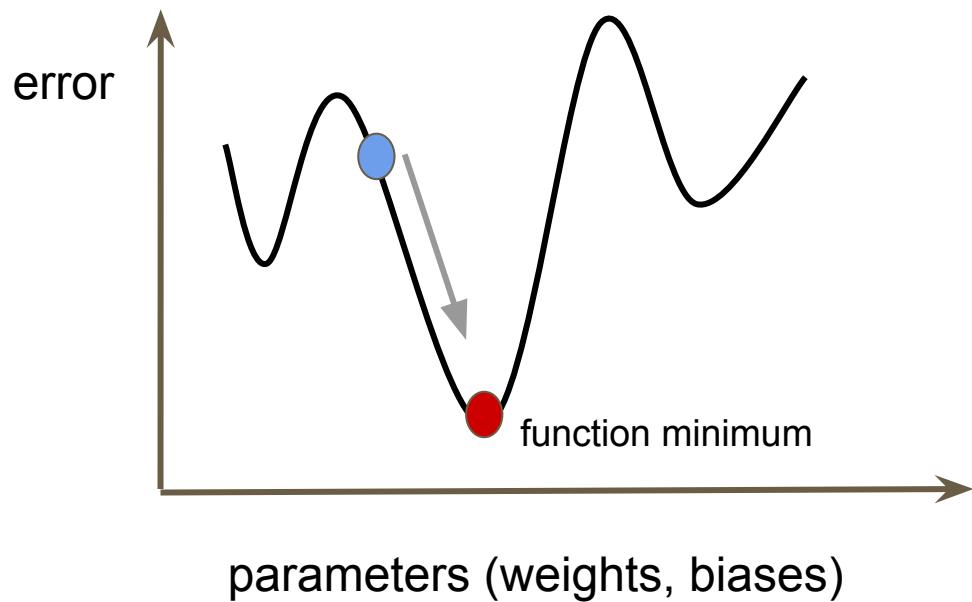
```
# create an operation that calculates loss  
loss = tf.reduce_mean(tf.square(y - y_train))
```

Optimization



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

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
```

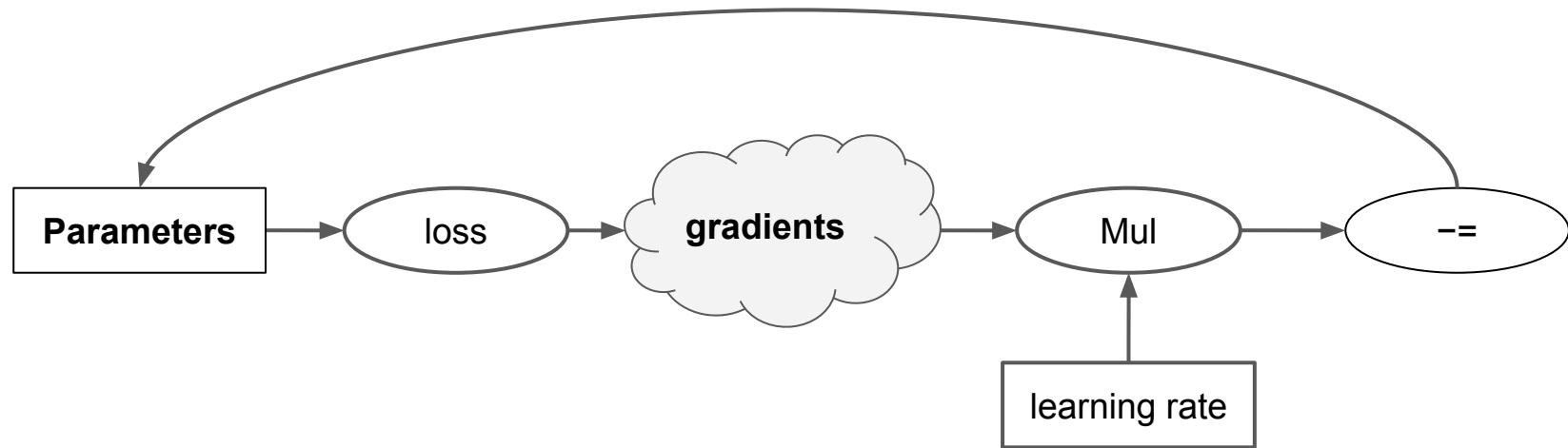
```
optimizer = tf.train.GradientDescentOptimizer(0.5)
```

```
# 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]})
```

The diagram illustrates the components of the provided TensorFlow code. It uses curly braces to group related code snippets and associate them with their respective functions:

- The first three lines of code (defining loss, creating an optimizer, and defining the minimize op) are grouped by a brace on the right, labeled "Define a loss", "Create an optimizer", and "Op to minimize the loss" respectively.
- The line "sess.run(tf.initialize_all_variables())" is grouped by a brace on the right, labeled "Initialize variables".
- The loop starting with "for i in range(1000)" and its body are grouped by a brace on the right, labeled "Iteratively run the training op".

Exercise #2: Linear Regression

2_linear_regression.ipynb

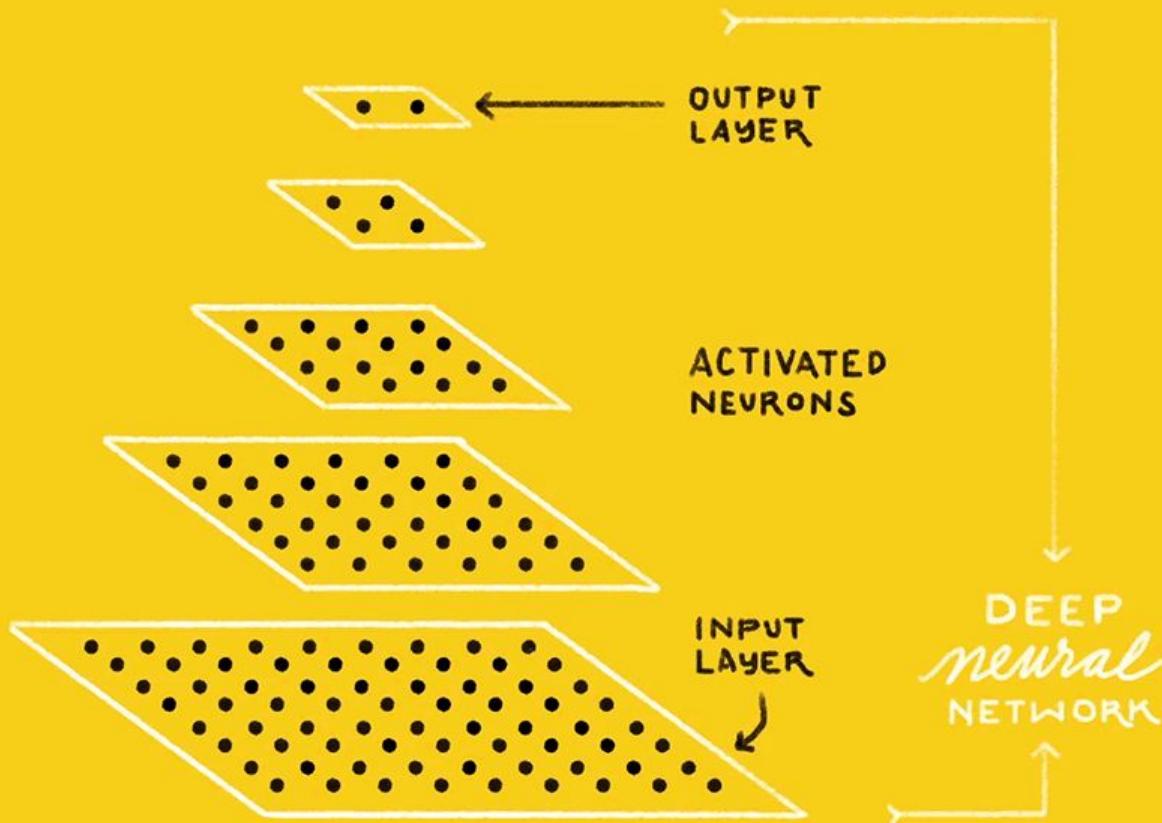
goo.gl/nrdsxM

Neural Networks

IS THIS A
CAT or DOG?



CAT DOG



Deep Learning

Current state of the art in:

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

Cloud Machine Learning

See, Hear and Understand the world



Google Cloud Platform



Cloud
Vision



Cloud
Natural Language



Cloud
Speech



Cloud
Translate

Cloud Vision API

Faces

Faces, facial landmarks, emotions



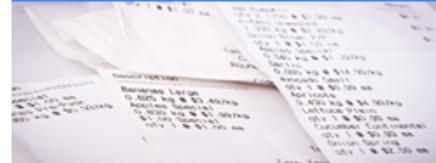
Label

Detect entities from furniture to transportation



OCR

Read and extract text, with support for > 10 languages



Logos

Identify product logos



Safe Search

Detect explicit content - adult, violent, medical and spoof



Landmarks & Image Properties

Detect landmarks & dominant color of image



Mobile Vision API

Face API

faces, facial landmarks,
eyes open, smiling



Barcode API

1D and 2D barcodes



Text API

Latin-based text / structure



NEW!

Common Mobile Vision API

Support for fast image and video on-device detection and tracking.

Understanding Images

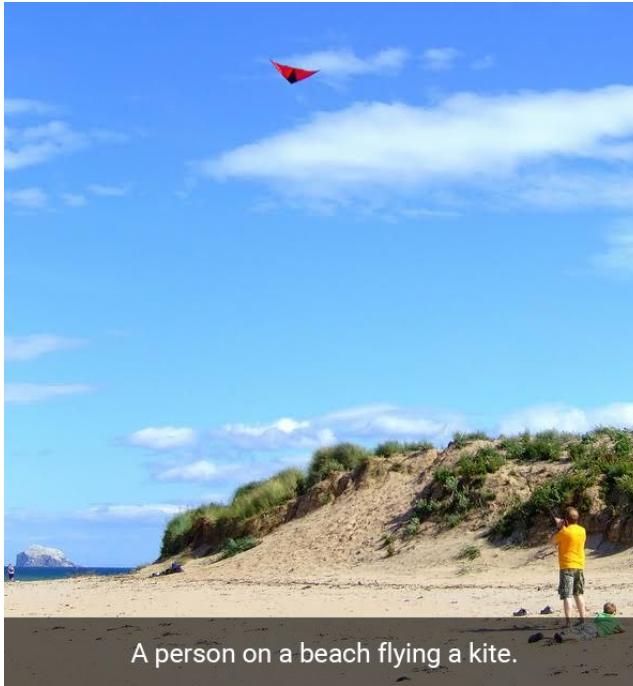
Inception



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

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

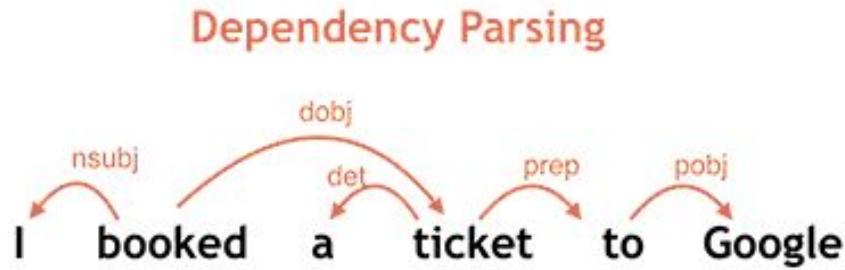
Show and Tell



<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

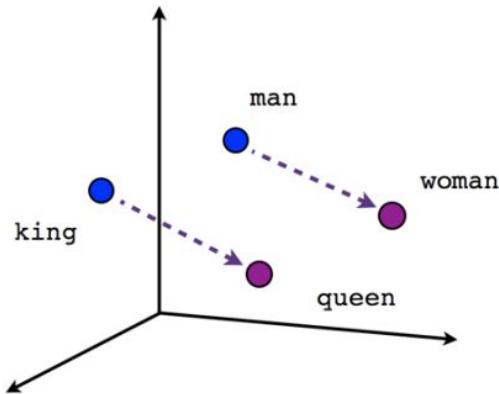
Original text

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

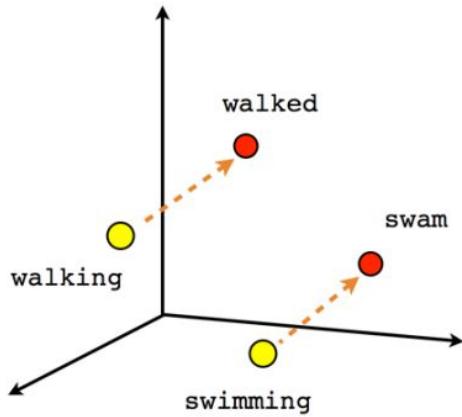
Abstractive summary

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

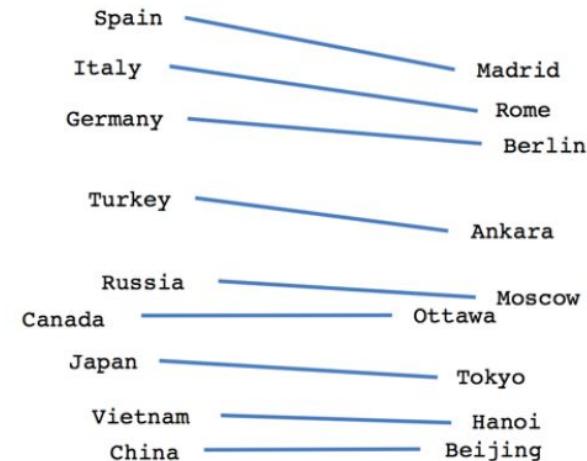
Word Embeddings



Male-Female

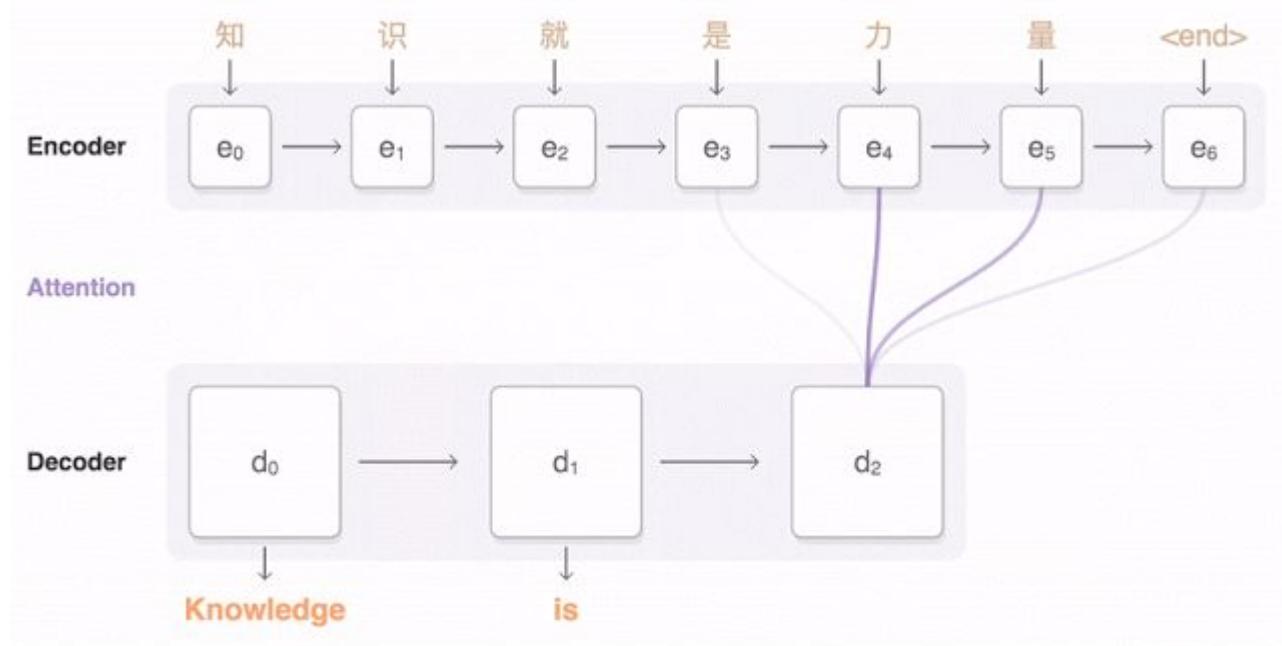


Verb tense



Country-Capital

Translation



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

Deep Learning and Art

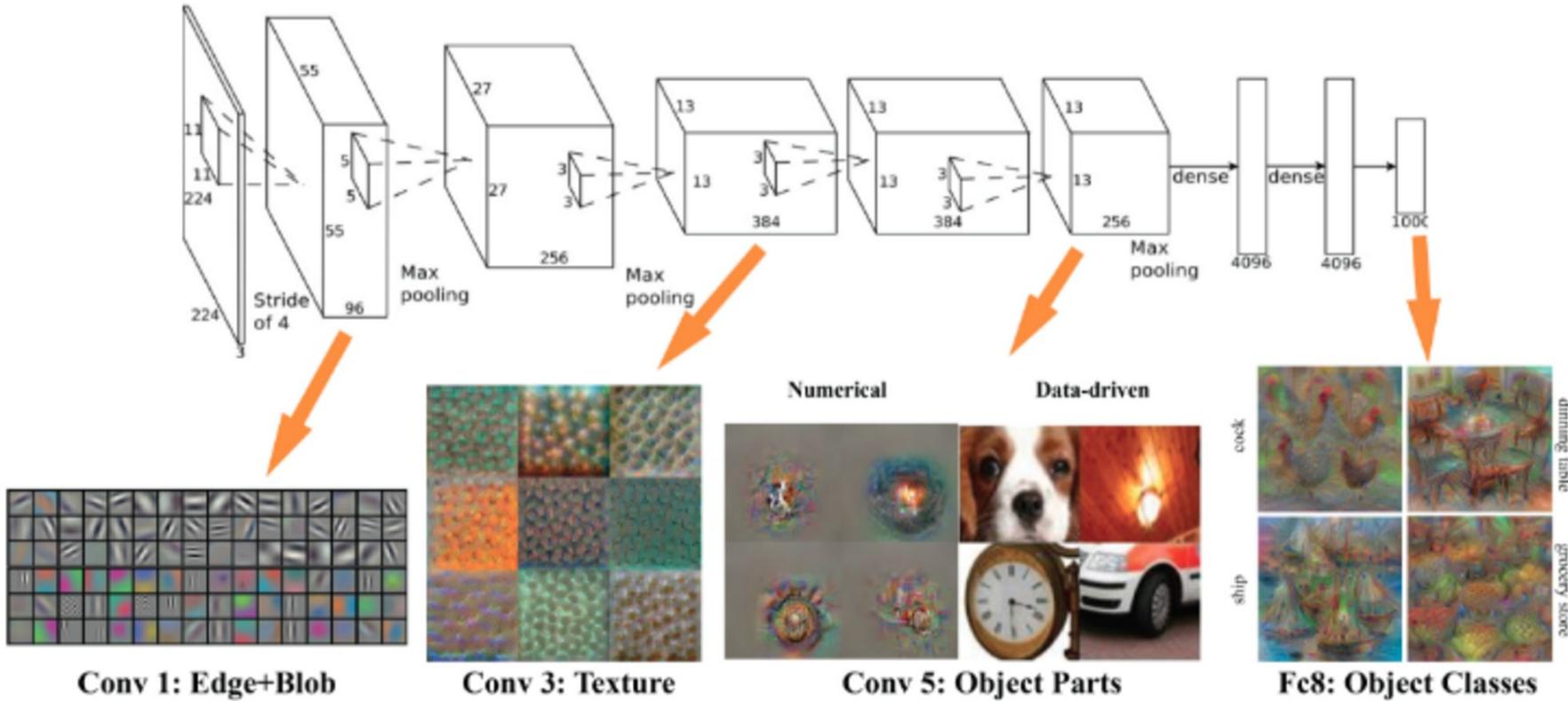


Image source - [Wikipedia](#)

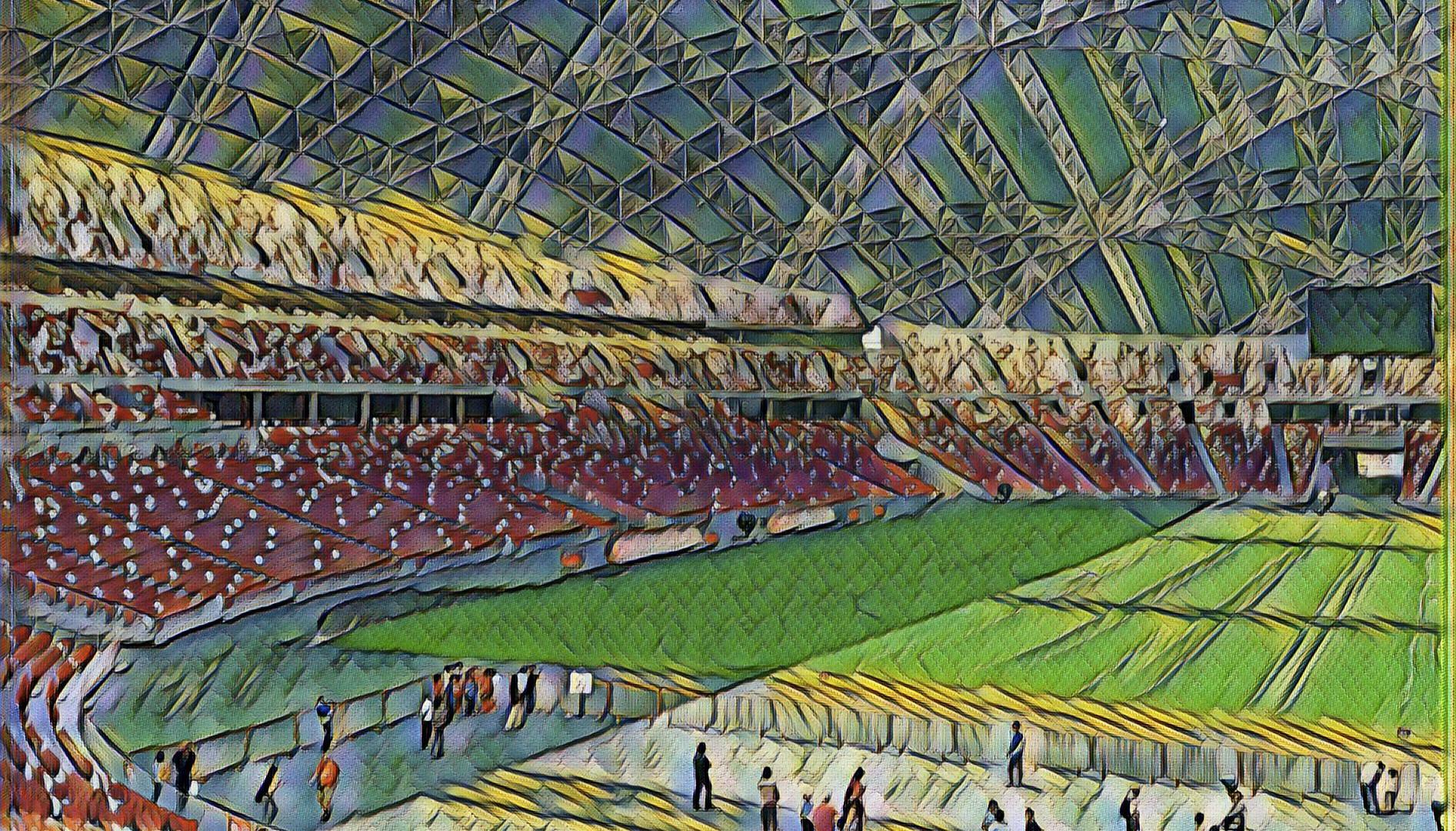




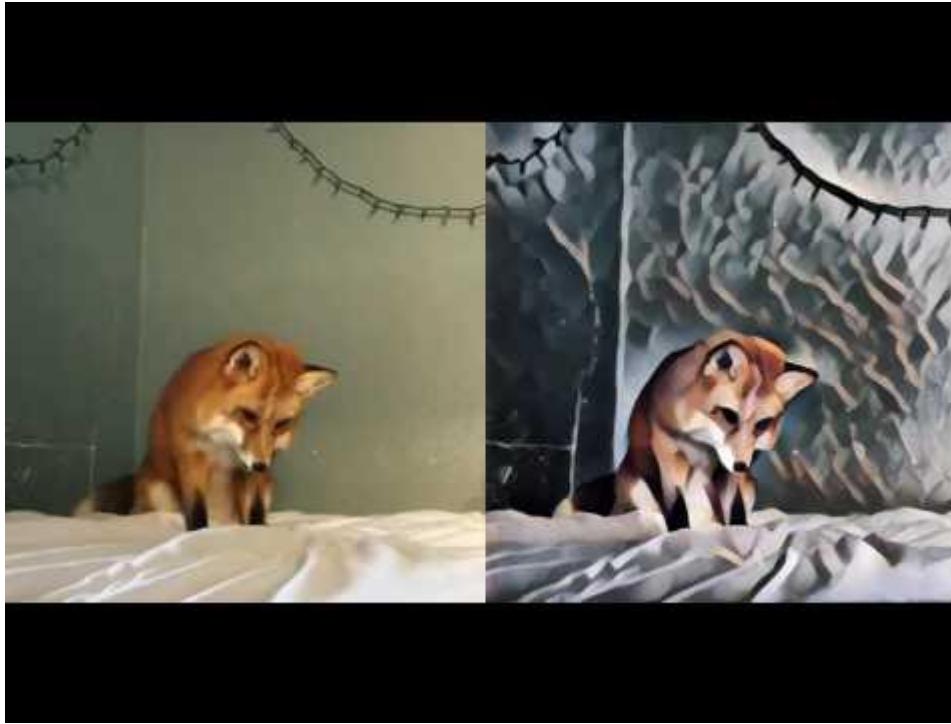
Image source - [Wikipedia](#)



From: [A Matlab Plugin to Visualize Neurons from Deep Models](#), Donglai Wei et. al.







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



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



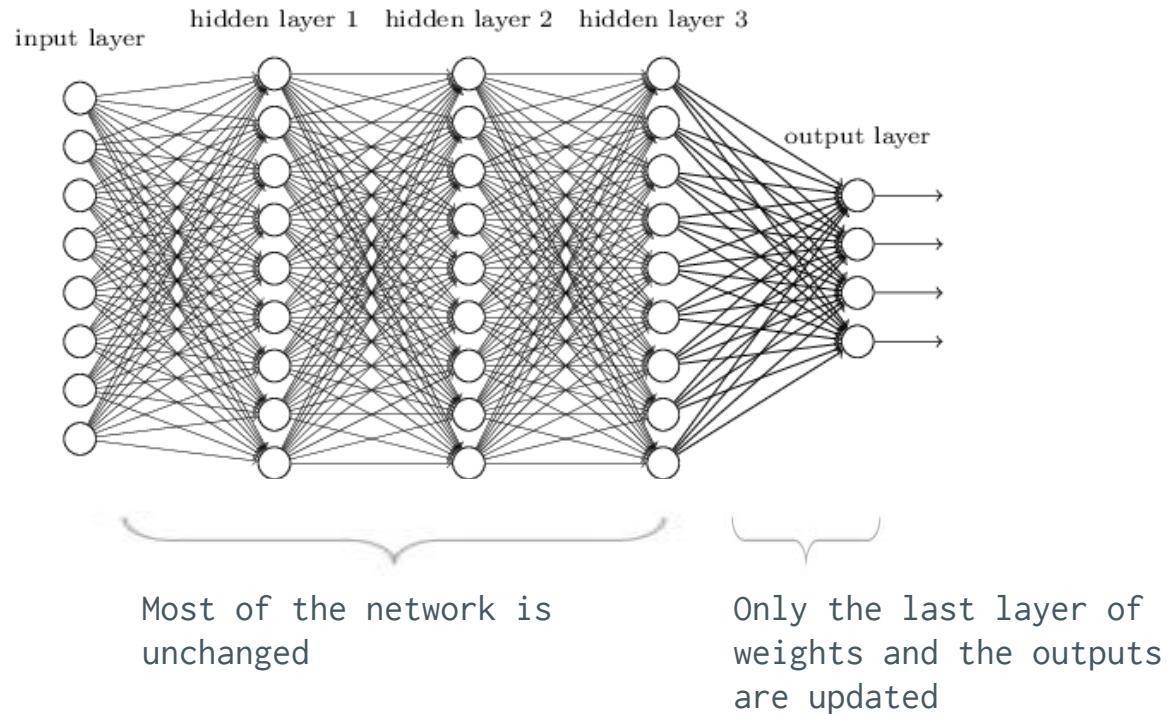
goo.gl/1kBXyO

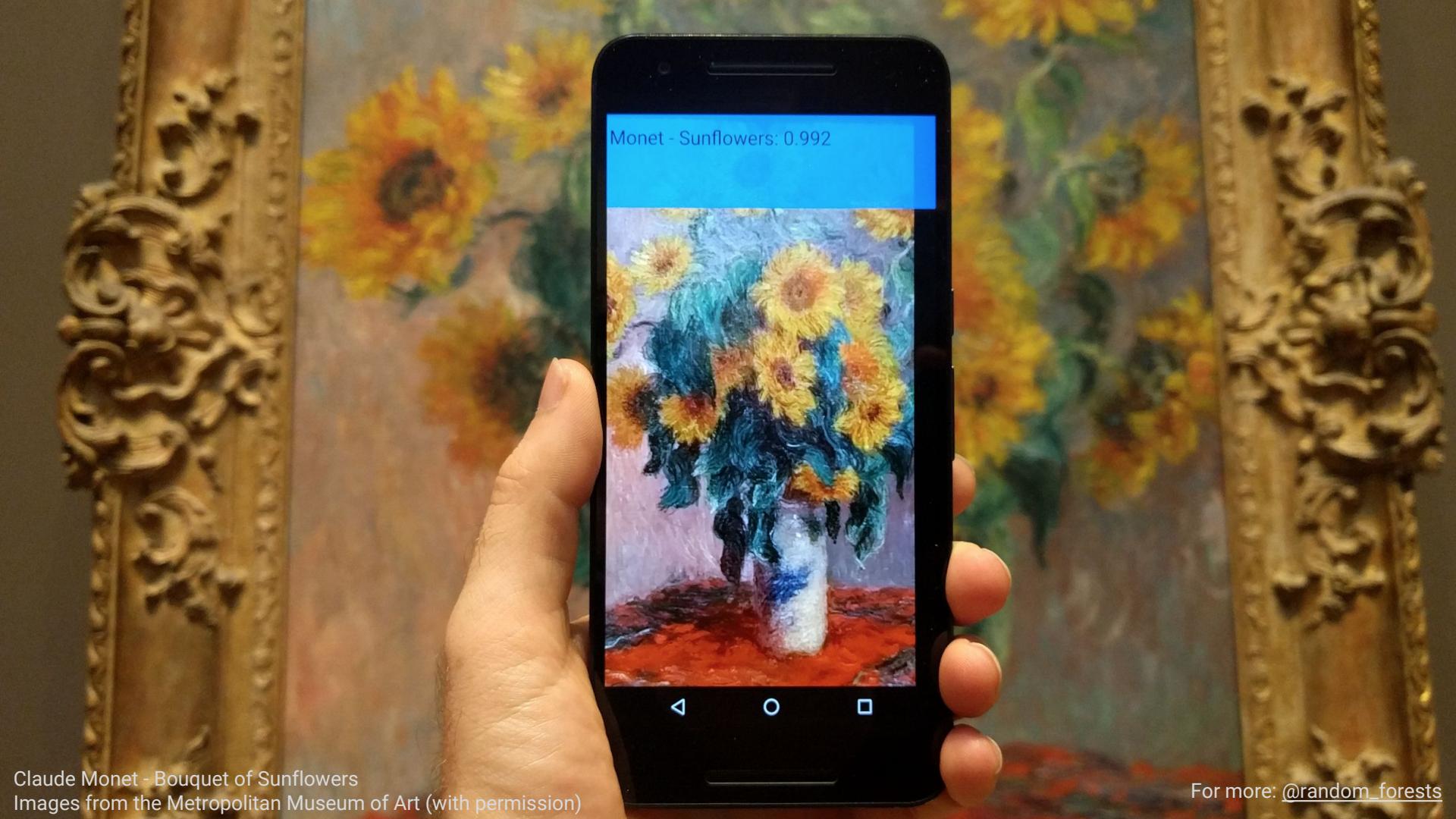


Create your own image classifier

goo.gl/xGsB9d

Transfer Learning





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

For more: [@random_forests](https://github.com/Random-Forests)

Exercise #3

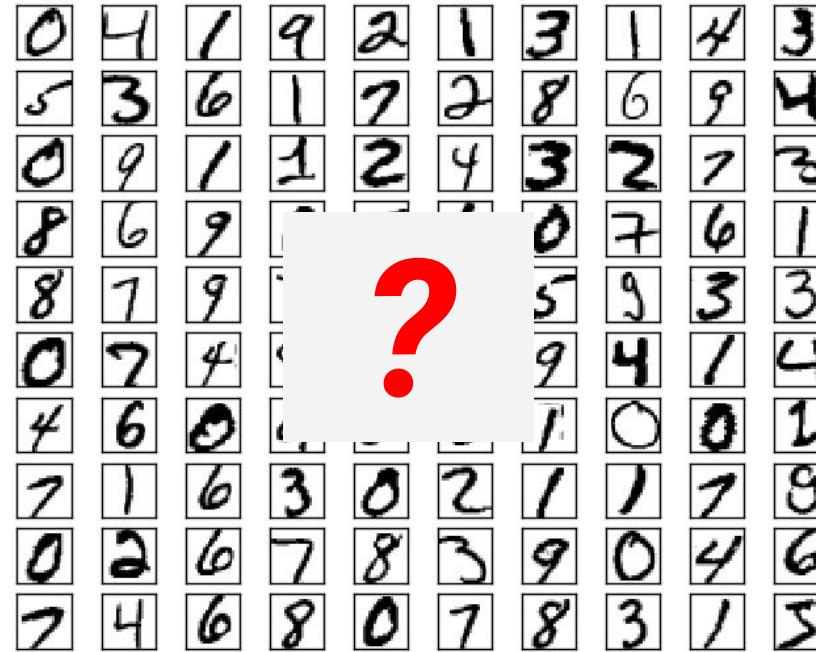
Fast Style Transfer
goo.gl/owF2z9

TensorFlow for Poets
goo.gl/xGsB9d

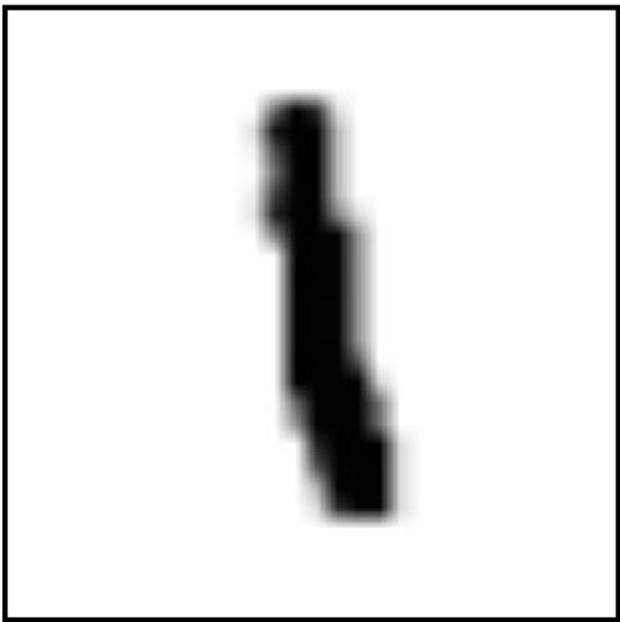
Deep Dream
goo.gl/1kBXyO

Basic and Deep MNIST

Hello “computer vision” world



We see



Computer “sees”

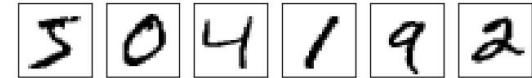
2

Dataset

Original: 65,000

1 8 2 0 2 9 9 5 5 1 5 6 0 3 4 4 6 5 4 6 5
9 2 5 0 1 1 1 0 9 0 3 1 6 4 2 3 6 1 1 1 3
2 8 4 1 7 3 3 8 8 7 9 2 2 4 1 5 9 8 7 2 3
1 8 1 8 0 3 0 1 9 9 4 1 8 2 1 2 9 7 5 9 2
0 2 7 4 3 3 0 0 3 1 9 6 5 0 5 1 7 9 3 0 4
0 5 1 3 1 5 5 6 1 8 5 1 1 4 4 6 2 2 5 0 6
6 2 1 9 2 8 6 1 9 5 2 5 4 4 2 8 3 8 2 4 5
9 1 4 8 1 8 4 5 9 9 8 3 7 6 0 0 3 0 2 6 1
7 5 8 9 6 1 8 4 1 2 5 9 1 9 7 5 4 0 8 9 9
6 5 7 1 2 6 3 2 6 5 4 8 9 7 1 3 0 3 8 3
3 2 7 7 0 8 7 4 4 7 9 6 9 0 9 8 0 4 6 0 6
4 3 8 0 9 6 3 8 0 9 9 6 8 6 8 5 7 8 6 0 2

Images



Labels

5 0 4 1 9 2

Train / test split

Original: 65,000

1 8 2 0 2 9 9 5 5 1 5 6 0 3 4 4 6 5 4 6 5
9 2 5 0 1 1 1 0 9 0 3 1 6 4 **2 3** 6 1 1 1 3
2 8 4 1 7 3 3 8 8 7 9 2 2 4 1 5 9 8 7 2 3
1 8 1 8 0 3 0 1 9 9 4 1 8 2 1 2 9 7 5 9 2
0 2 7 4 3 3 0 0 3 1 9 6 5 3 6 1 7 9 3 0 4
0 5 1 3 1 5 5 6 1 8 5 1 1 9 4 6 2 2 5 0 6
6 2 1 9 2 8 6 1 9 5 2 5 4 4 2 8 3 8 2 4 5
9 1 4 8 1 8 4 5 9 9 8 3 7 6 0 0 3 0 2 6 4
7 5 8 9 6 1 8 4 1 2 5 9 1 9 7 5 4 0 8 9 9
6 5 7 1 2 6 3 2 6 5 4 8 9 7 1 3 0 3 8 3
3 2 7 7 0 8 7 4 4 7 9 6 9 0 9 8 0 4 6 0 6
4 3 8 0 9 6 3 8 0 9 9 6 8 6 8 5 7 8 6 0 2

Train: 55,000

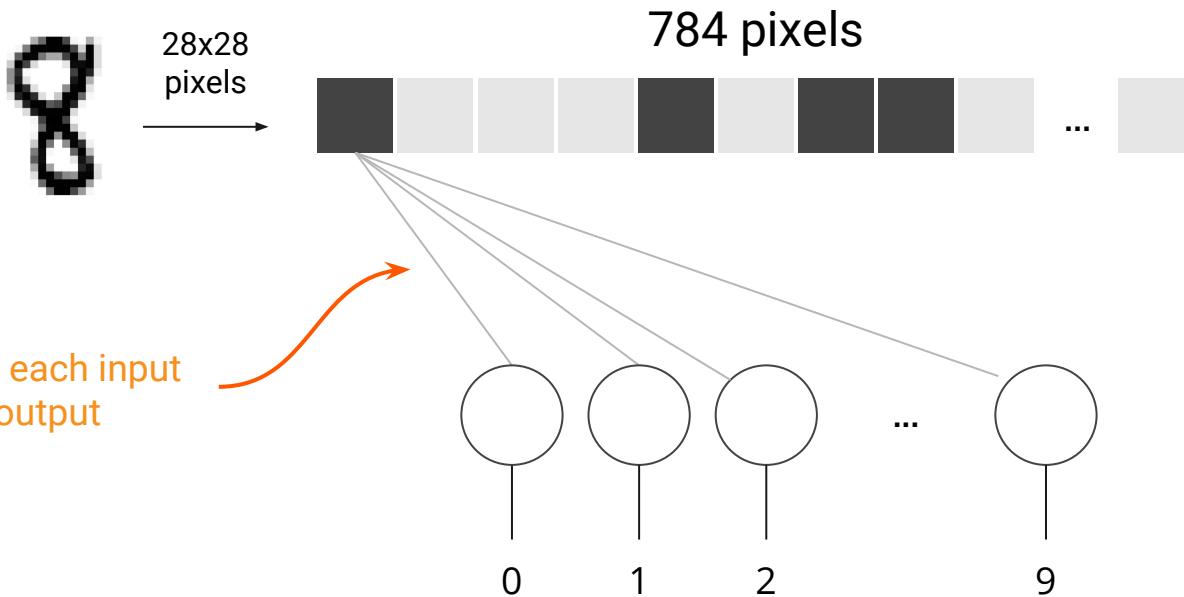
1 8 2 0 2 9 9 5 5 1 5 6 0 3 4 4
9 2 5 0 1 1 1 0 9 0 3 1 6 4 **2 3**
2 8 4 1 7 3 3 8 8 7 9 2 2 4 1 5
1 8 1 8 0 3 0 1 9 9 4 1 8 2 1 2
0 2 7 4 3 3 0 0 3 1 9 6 5 3 6 1
0 5 1 3 1 5 5 6 1 8 5 1 1 9 4 6
6 2 1 9 2 8 6 1 9 5 2 5 4 4 2 8
9 1 4 8 1 8 4 5 9 9 8 3 7 6 0 0
7 5 8 9 6 1 8 4 1 2 5 9 1 9 7 5
6 5 7 1 2 6 3 2 6 5 4 8 9 7 1
3 2 7 7 0 8 7 4 4 7 9 6 9 0 9 8
4 3 8 0 9 6 3 8 0 9 9 6 8 6 8 5

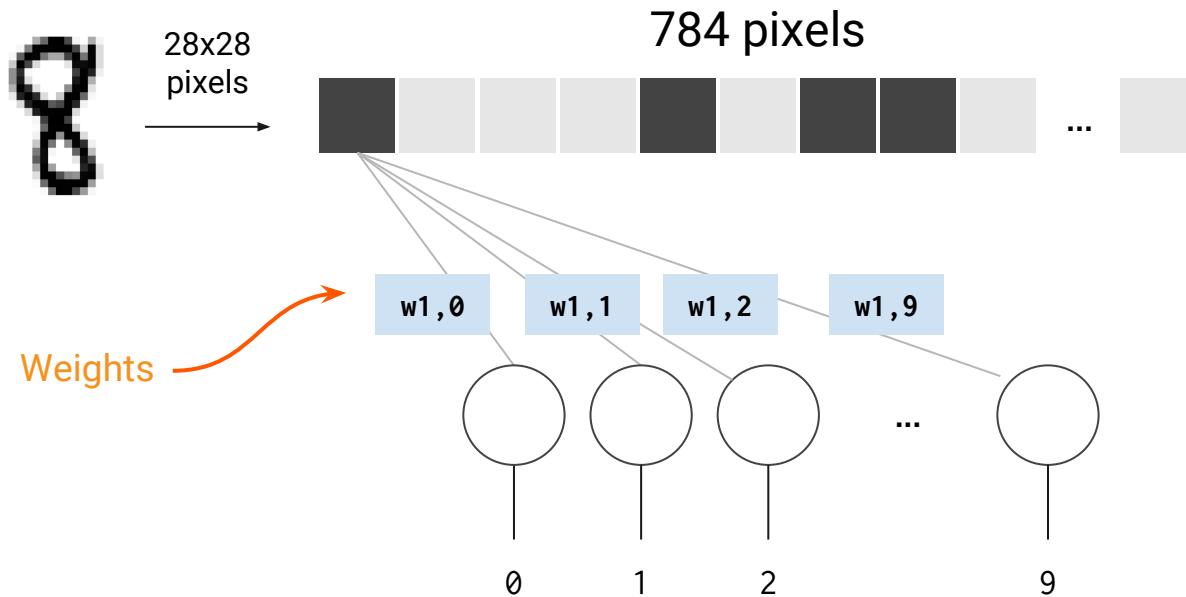
Test: 10,000

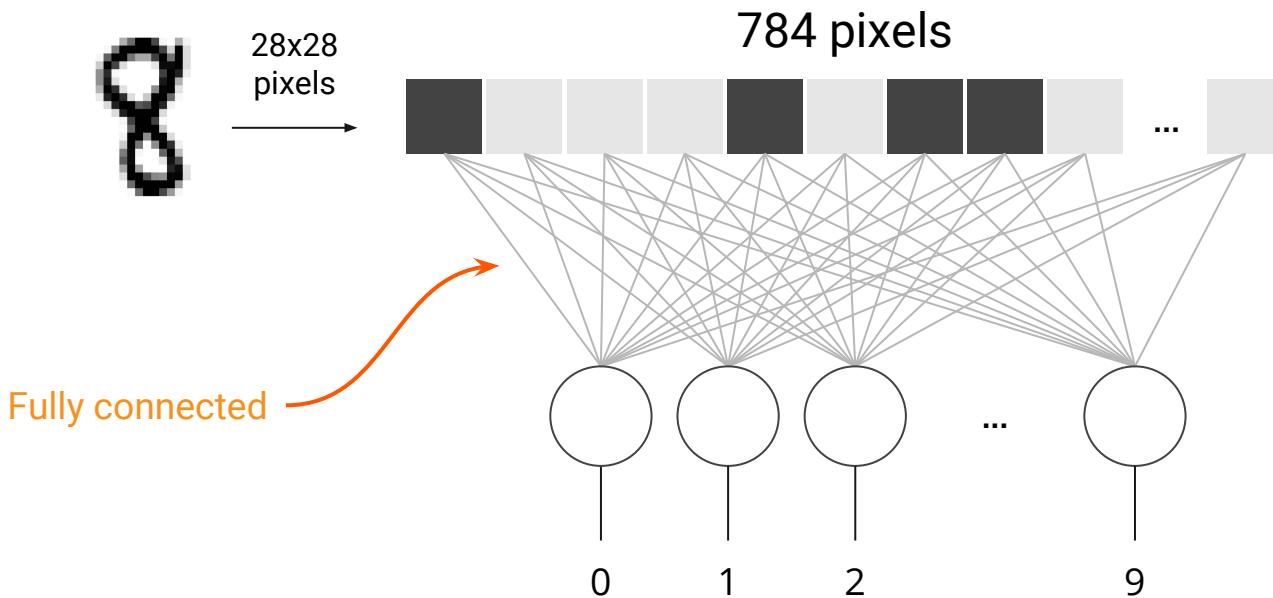
4 4 6 5
2 3 6 1
1 5 9 8
1 2 9 7
5 1 7 9
4 6 2 2
2 8 3 8
0 0 3 0
7 5 4 0
7 1 3 0
9 8 0 4
8 5 7 8

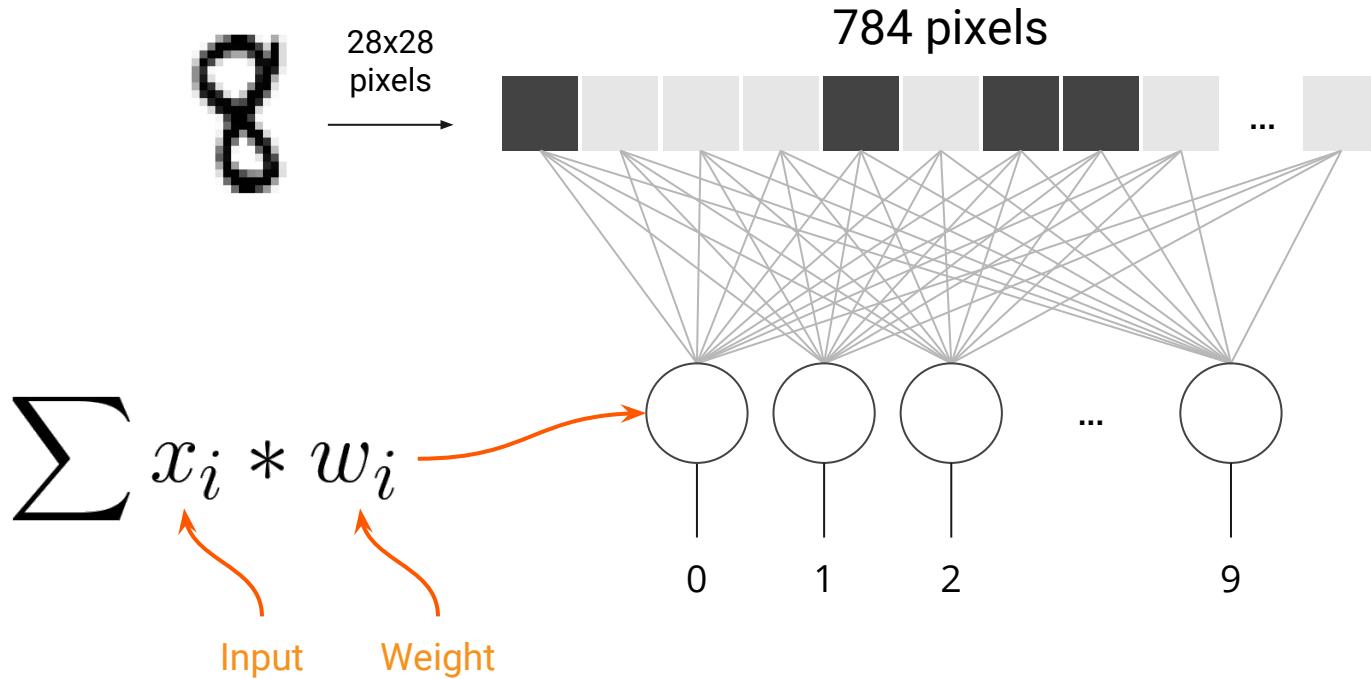
Placeholders

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









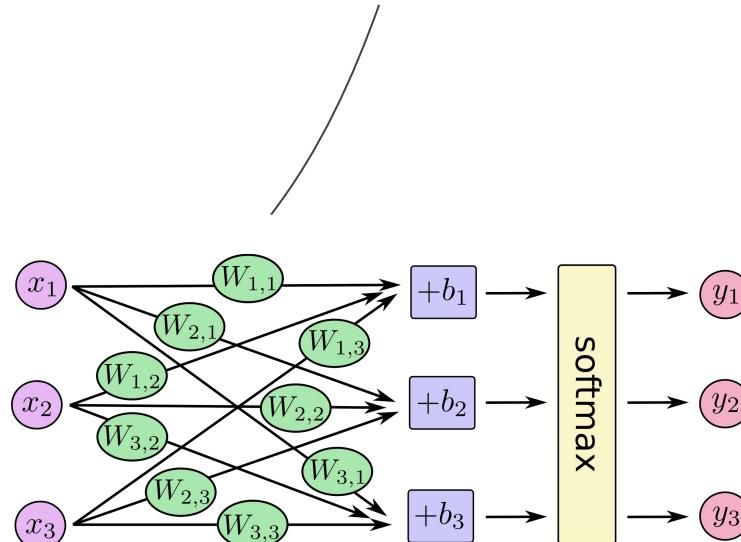
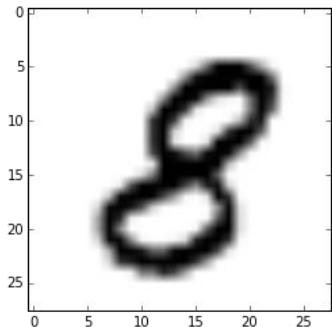
Inference...

```
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
```

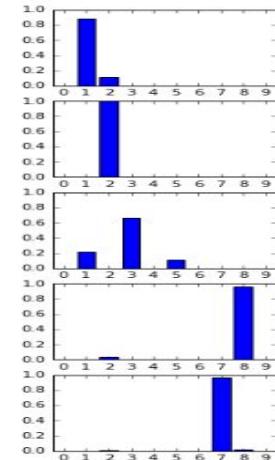
Softmax

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

input vector
(pixel data)



output vector
(probability of
each digit)



Written as a matrix multiply

Weight matrix with dimensions [748,10]


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

In TensorFlow

tensor shapes: $X[batch_size, 784]$ $W[784, 10]$ $b[10]$

$Y = \text{tf.nn.softmax}(\text{tf.matmul}(X, W) + b)$

matrix multiply

broadcast

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

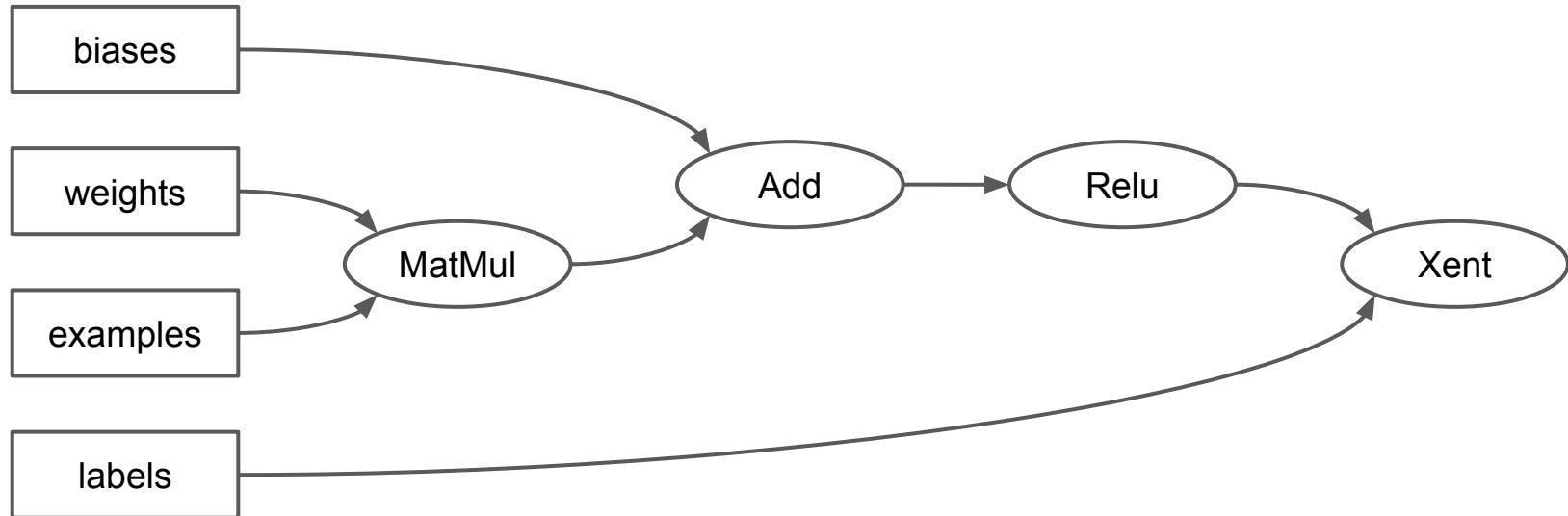
```
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))

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

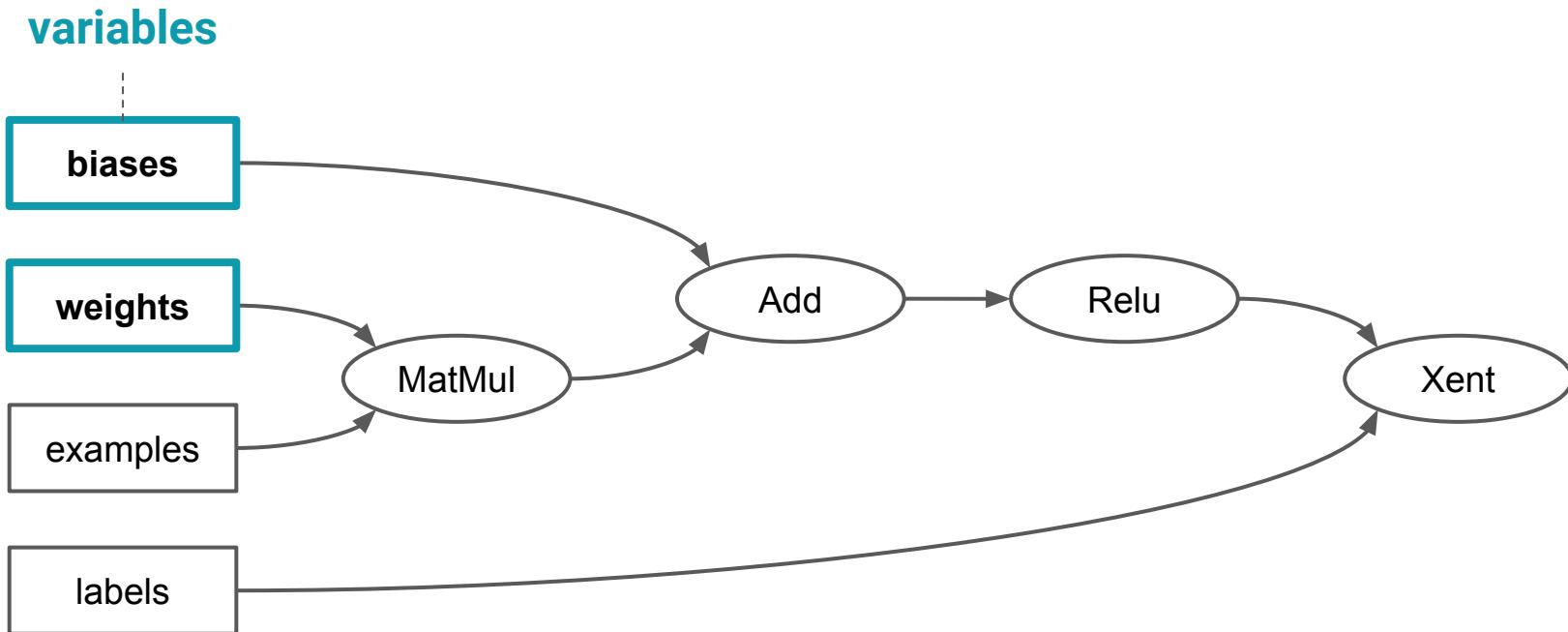
3_basic_mnist.ipynb

goo.gl/nrdsxM

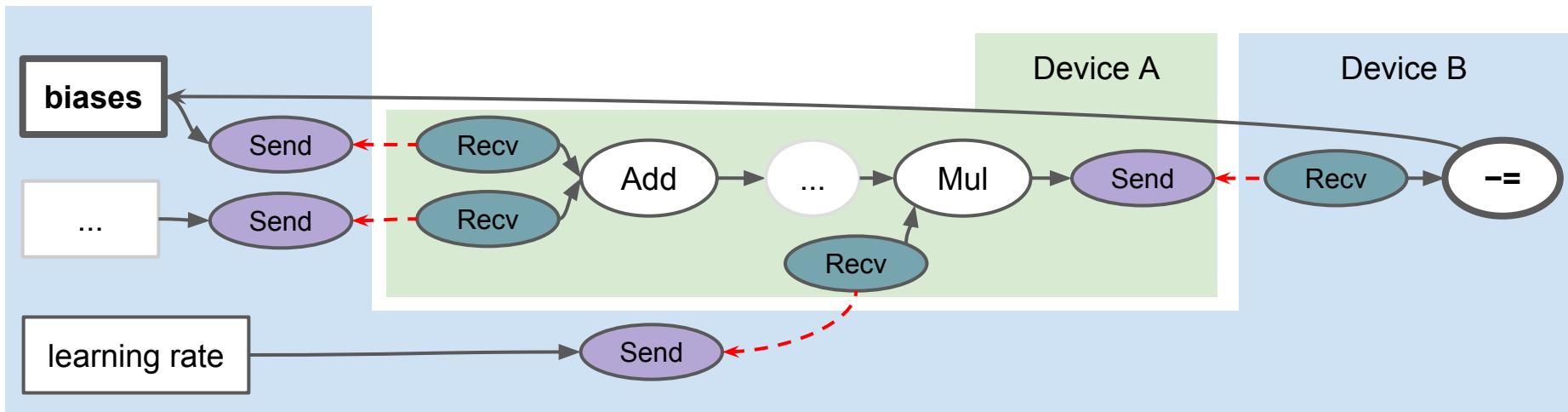
Graphs can be processed, compiled, remotely executed, assigned to devices.



Data flow, with state



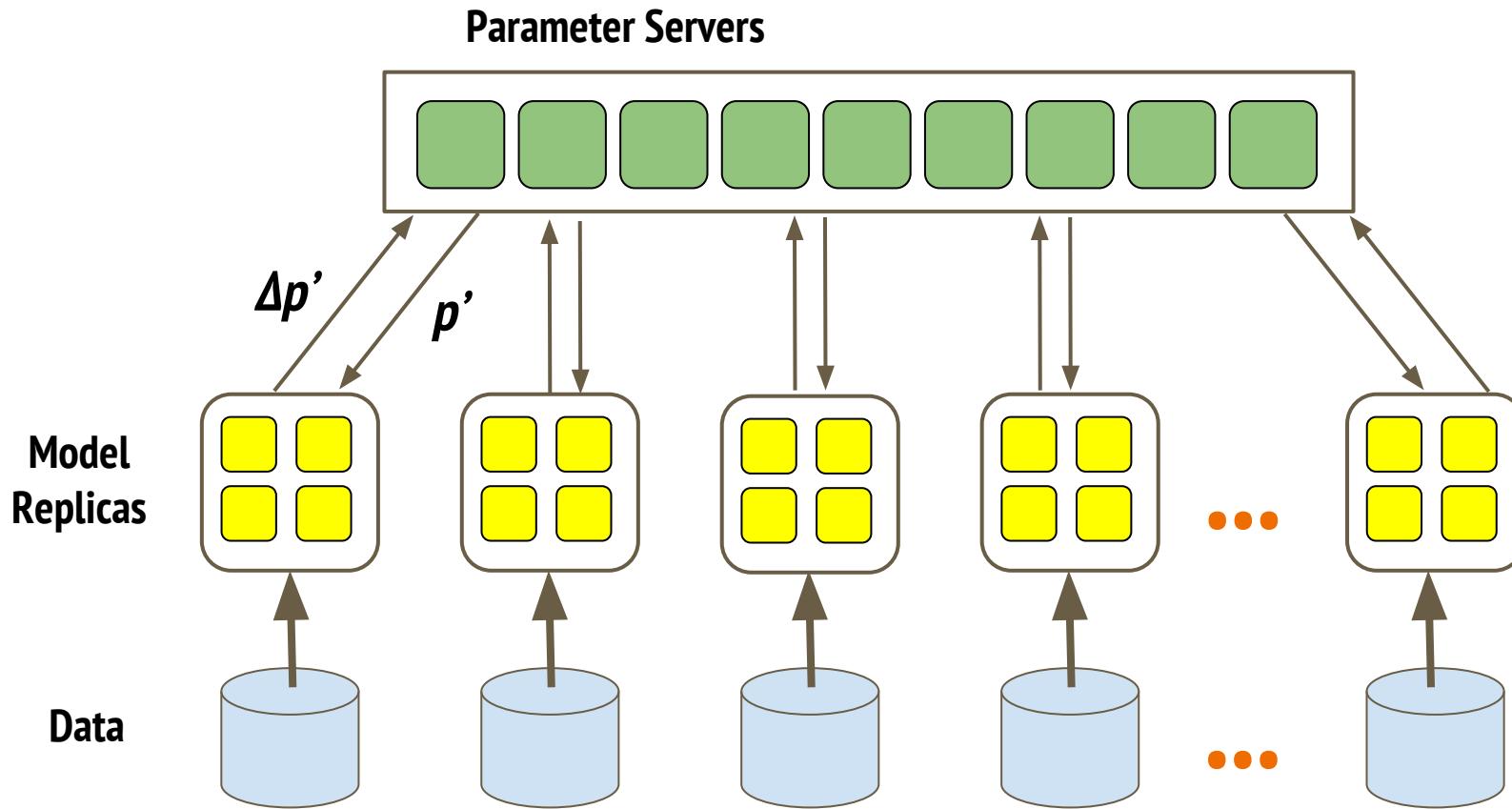
Distributed



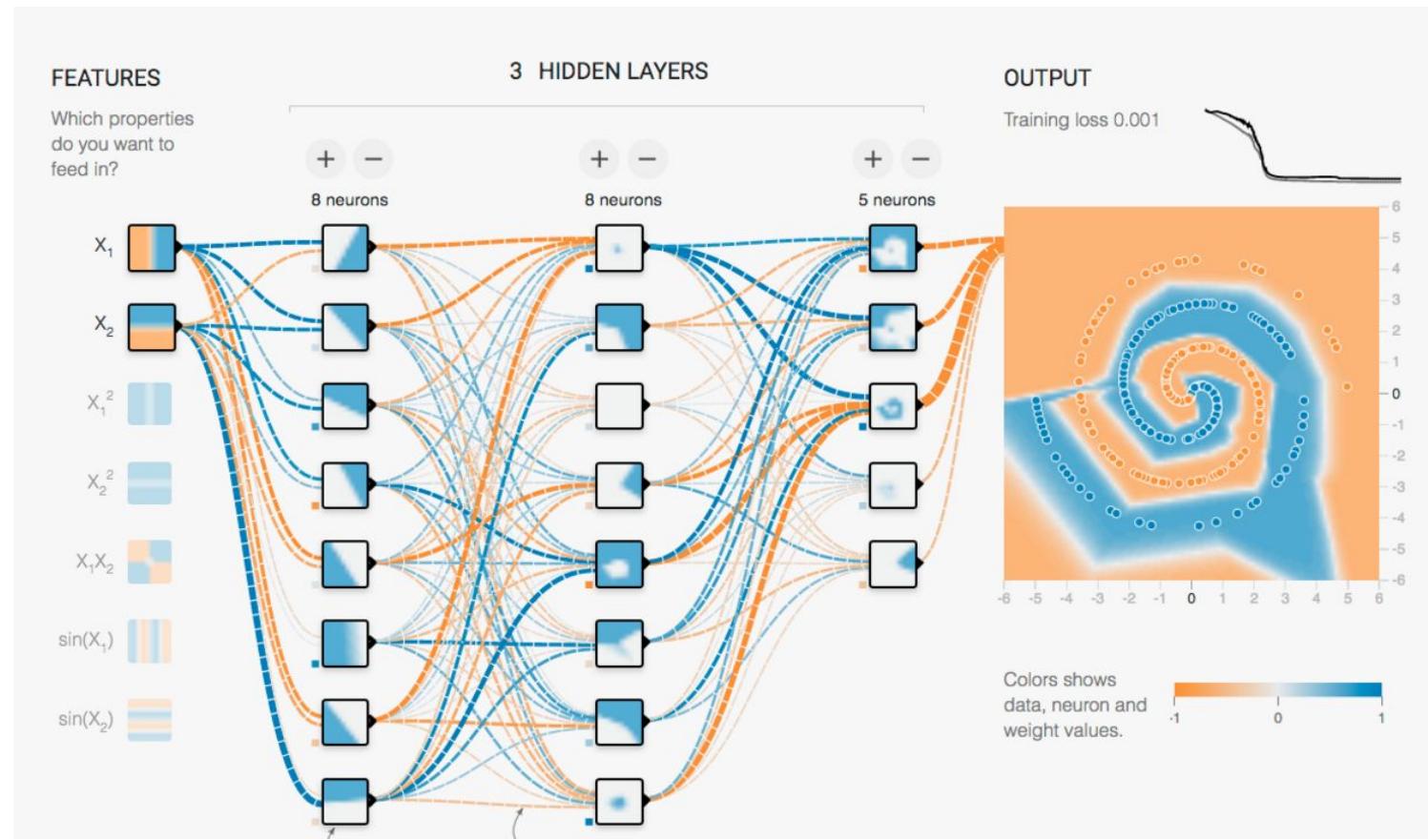
Send / receive nodes added automatically

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

Data Parallelism

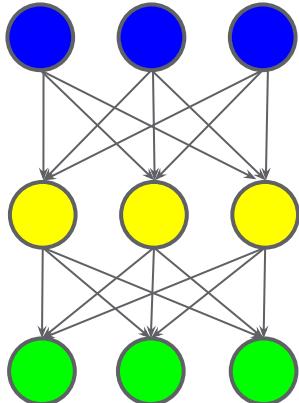


Going Deeper



Hidden layers = Hierarchies of features

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
```

Convolution

1 <small>$\times 1$</small>	1 <small>$\times 0$</small>	1 <small>$\times 1$</small>	0	0
0 <small>$\times 0$</small>	1 <small>$\times 1$</small>	1 <small>$\times 0$</small>	1	0
0 <small>$\times 1$</small>	0 <small>$\times 0$</small>	1 <small>$\times 1$</small>	1	1
0	0	1	1	0
0	1	1	0	0

Image

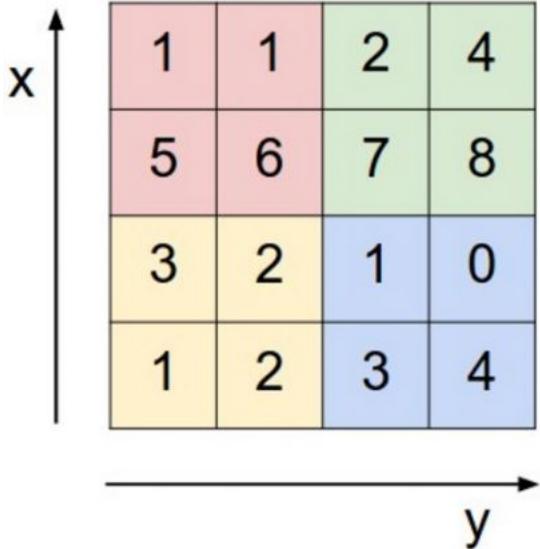
4		

Convolved Feature

Convolution with 3×3 Filter. Source:

http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution

Single depth slice



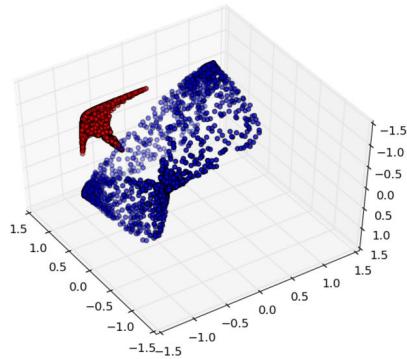
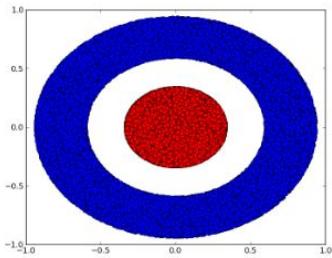
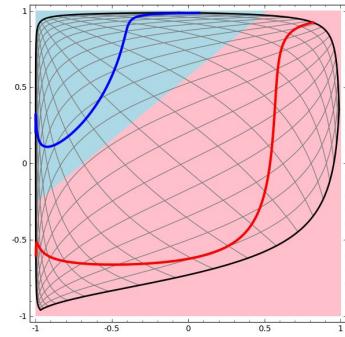
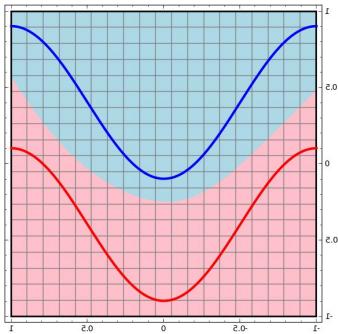
max pool with 2x2 filters
and stride 2

A 2x2 grid representing the output of max pooling. The values are:

6	8
3	4

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

Learn more: colah.github.io



From: [Neural Networks, Manifolds, and Topology](https://colah.github.io/posts/2014-04-neural-networks-manifolds-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

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

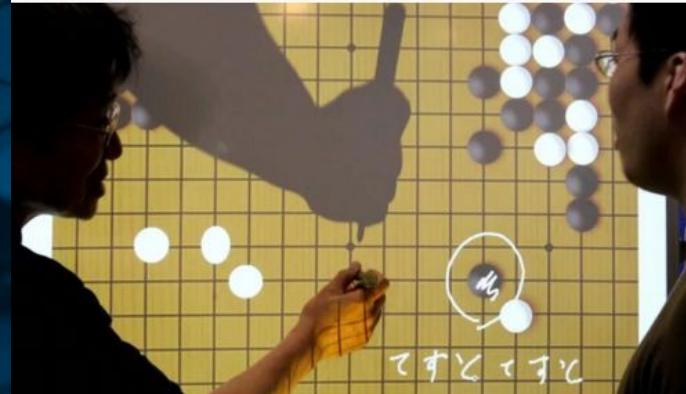
- How did they do it?
- What is the game Go?

Facebook trains AI to beat humans at Go



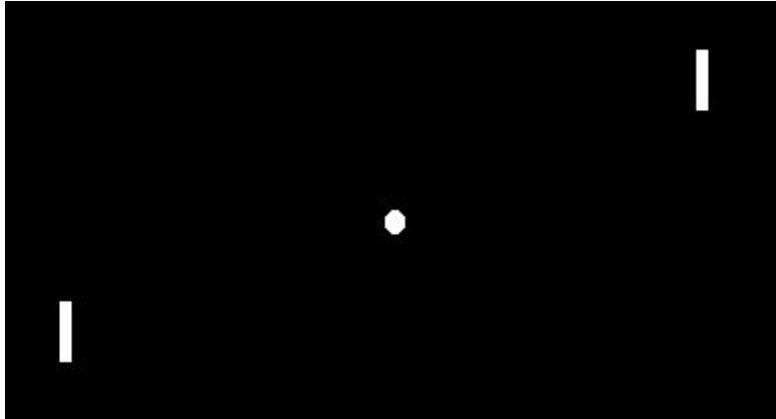
Google's AI just cracked the game that supposedly no computer could beat

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.



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

Detection of Diabetic Eye Disease

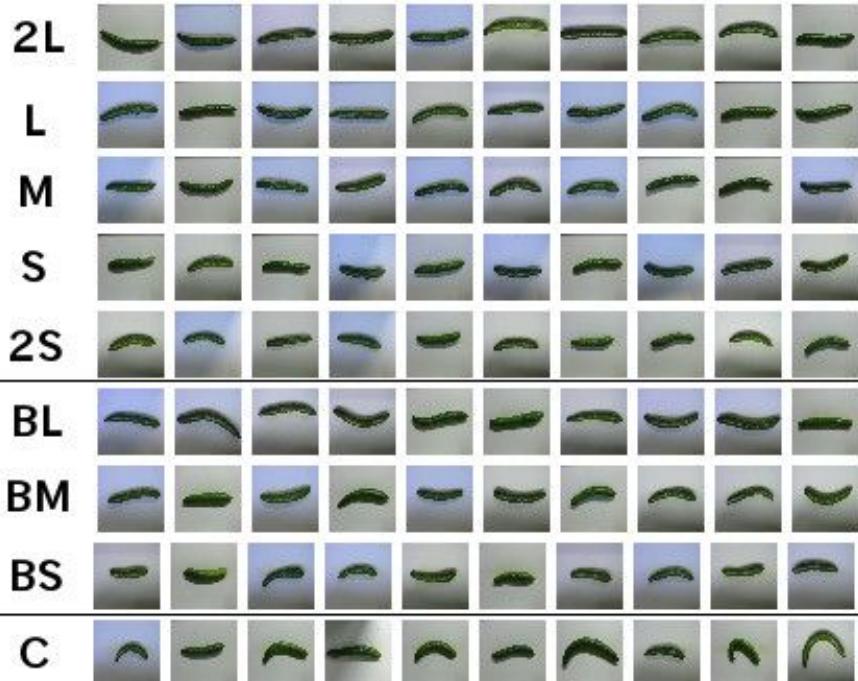


<https://research.googleblog.com/2016/11/deep-learning-for-detection-of-diabetic.html>



<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/>

The background features a wireframe landscape graphic with a grid of white lines forming a series of rolling hills or mountains against a solid orange background.

Questions?