TensorFlow CodeLab

GDG Milano

Università degli Studi
Milano Bicocca

12 November 2016

#Google
#TensorFlow
#ML

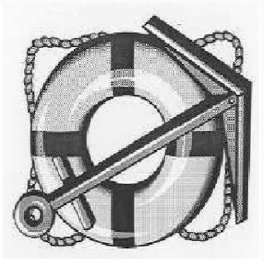
What we’ll do today

(credits: Google Developers)

Image Classifier?









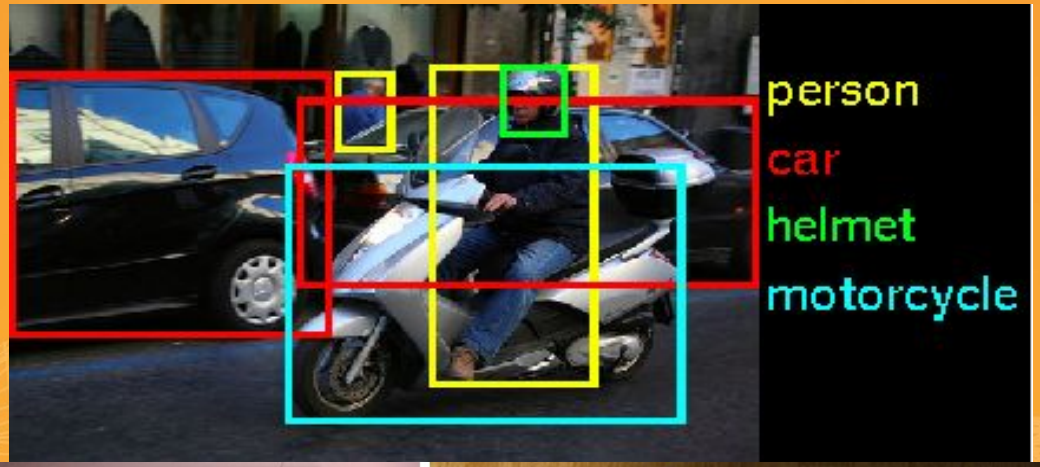


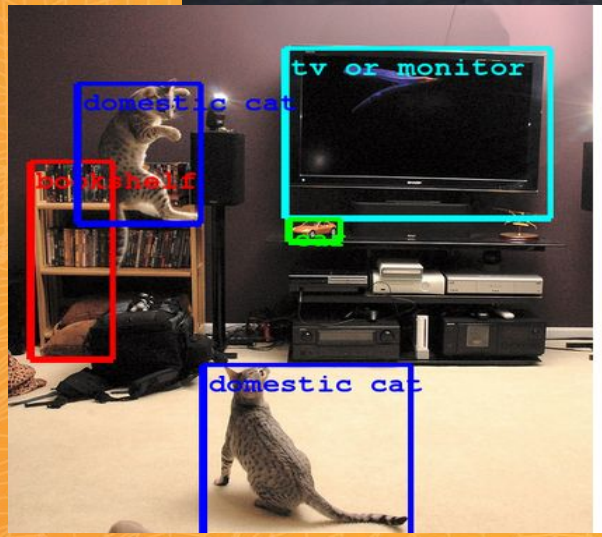


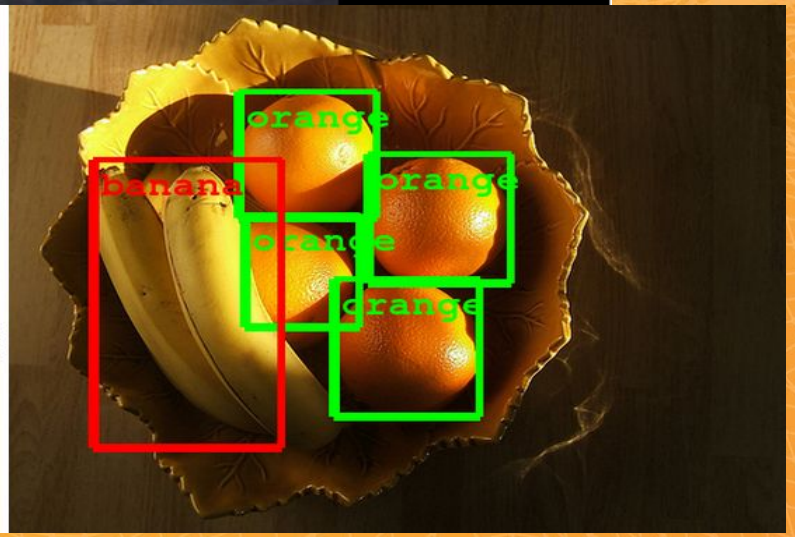


accordion (93%) | carside (92%) | pagoda (89%) | scorpion (15%) | ibis (8%) | anchor (7%)

(credits: INRIA)







(credits: Google Research)

(credits: Google)

(credits: Google)

More Precisely?

Classification

“Classification is a general process related to categorization, the process in which ideas and objects are recognized, differentiated, and understood.” - **Wikipedia**

“Statistical classification, identifies to which of a set of categories a new observation belongs, on the basis of a training set of data.” - **Wikipedia**

Statistics?
Training Set?

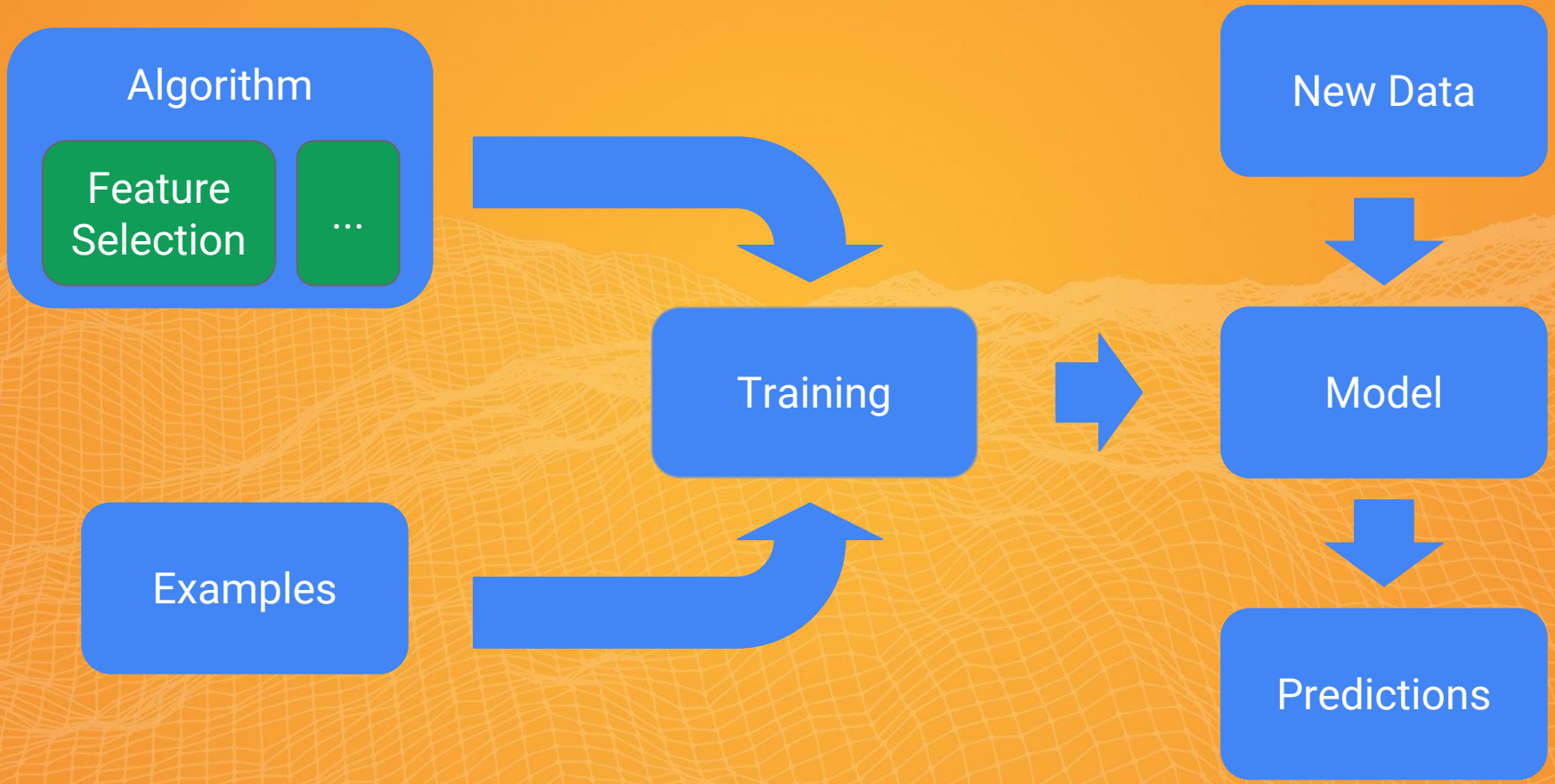
Machine Learning

“Machine learning explores the study and construction of algorithms that can learn from and make predictions on data. Such algorithms operate by building a model from example inputs in order to make data-driven predictions or decisions, rather than following strictly static program instructions” - **Wikipedia**

Predictive Model

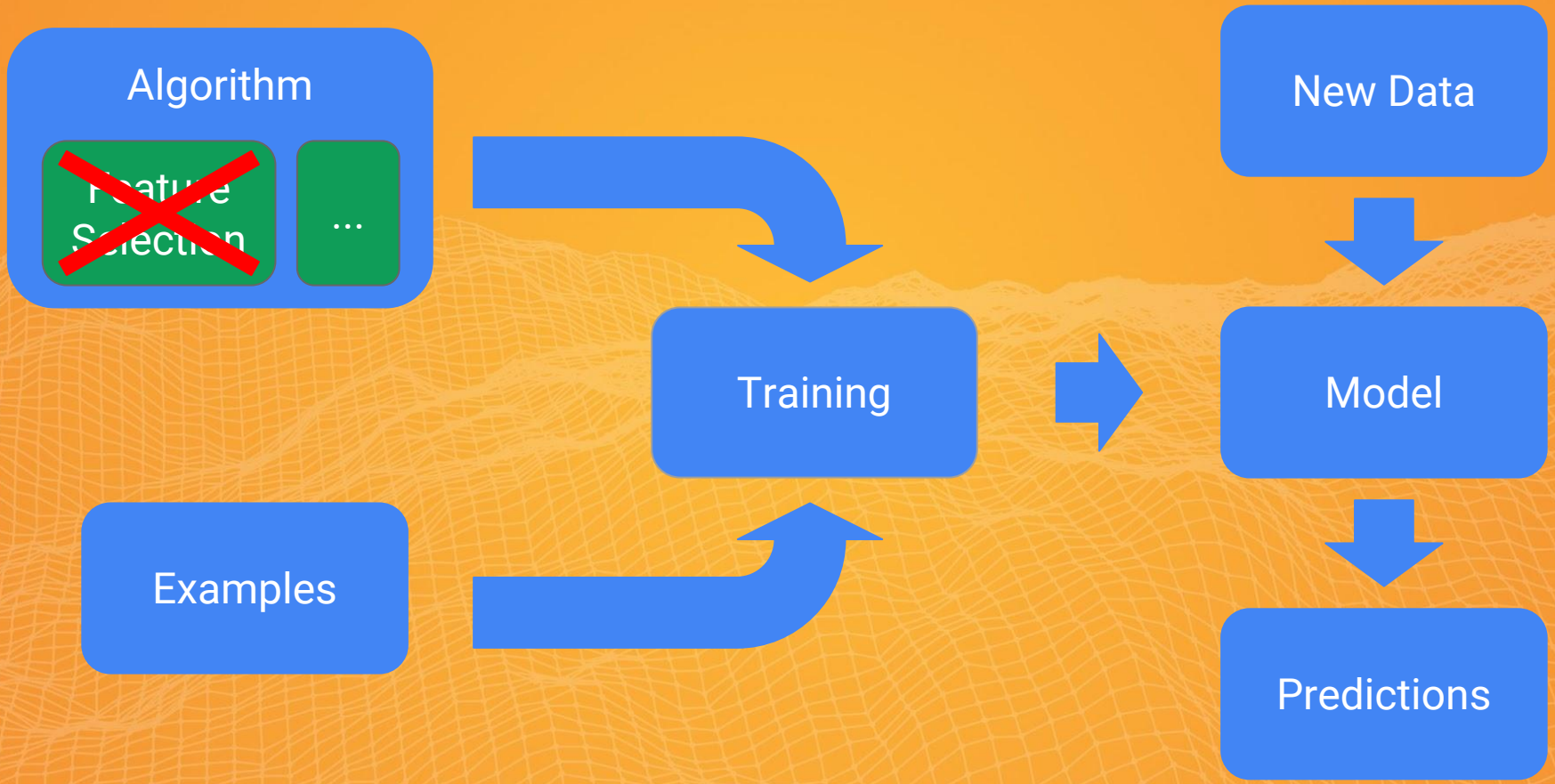
“Representation of a phenomenon.

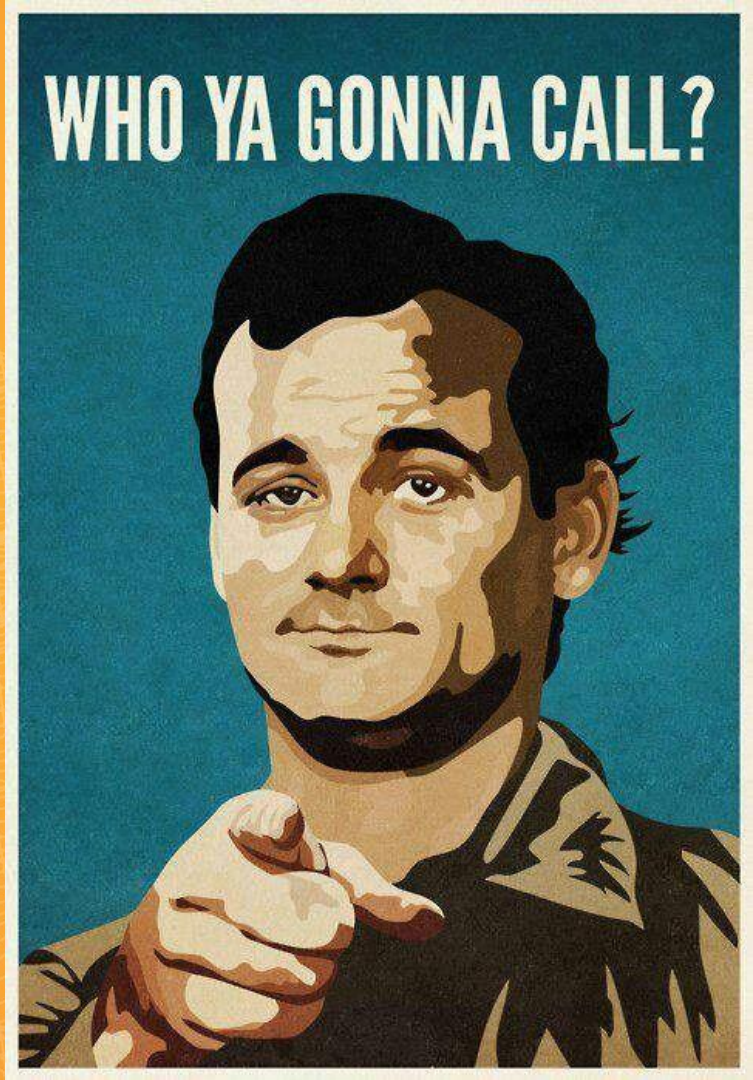
“It can be used to generate knowledge from data and to predict an outcome.”


Algorithm
Feature Selection
...
Examples
Training
New Data
Model
Predictions

Deep Learning

- A family of Machine Learning algorithms
- No feature engineering needed
- They perform better for problems like:
  - Image Recognition
  - Audio Recognition
  - Natural Language Processing


Algorithm
Feature Selection
...
Examples
Training
New Data
Model
Predictions


WHO YA GONNA CALL?

TensorFlow

TensorFlow

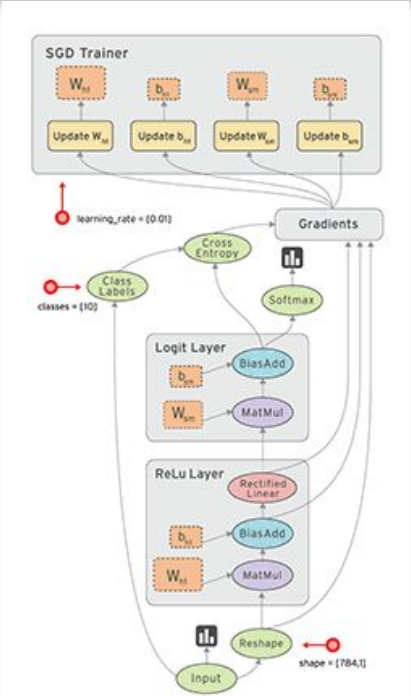
“TensorFlow is an open source software library for numerical computation using data flow graphs

Nodes in the graph represent mathematical operations, while the graph edges represent the multidimensional data arrays (tensors) communicated between them”

Data Flow Graph

Computation is defined as a directed acyclic graph (DAG) to optimize an objective function

- Graph is defined in high-level language
- Graph is compiled and optimized
- Graph is executed on available low level devices (CPU, GPU)
- Data flow through the graph




I COULD DO IT
USING NUMPY
imgflip.com


Numpy
TensorFlow
Python 1
Compiled 1
Python 2
Compiled 2
Python 3
Compiled 3
Python 1
Compiled 1
Python 2
Compiled 2
Python 3
Compiled 3

Core TensorFlow Structures

- **Graph**: a TensorFlow computation, represented as a dataflow graph.
  - collection of operations that can be executed together as a group
- **Operation**: a graph node that performs computation on tensors
- **Tensor**: a handle to one of the output of an operation
  - provides a means of computing the value in a TensorFlow Session

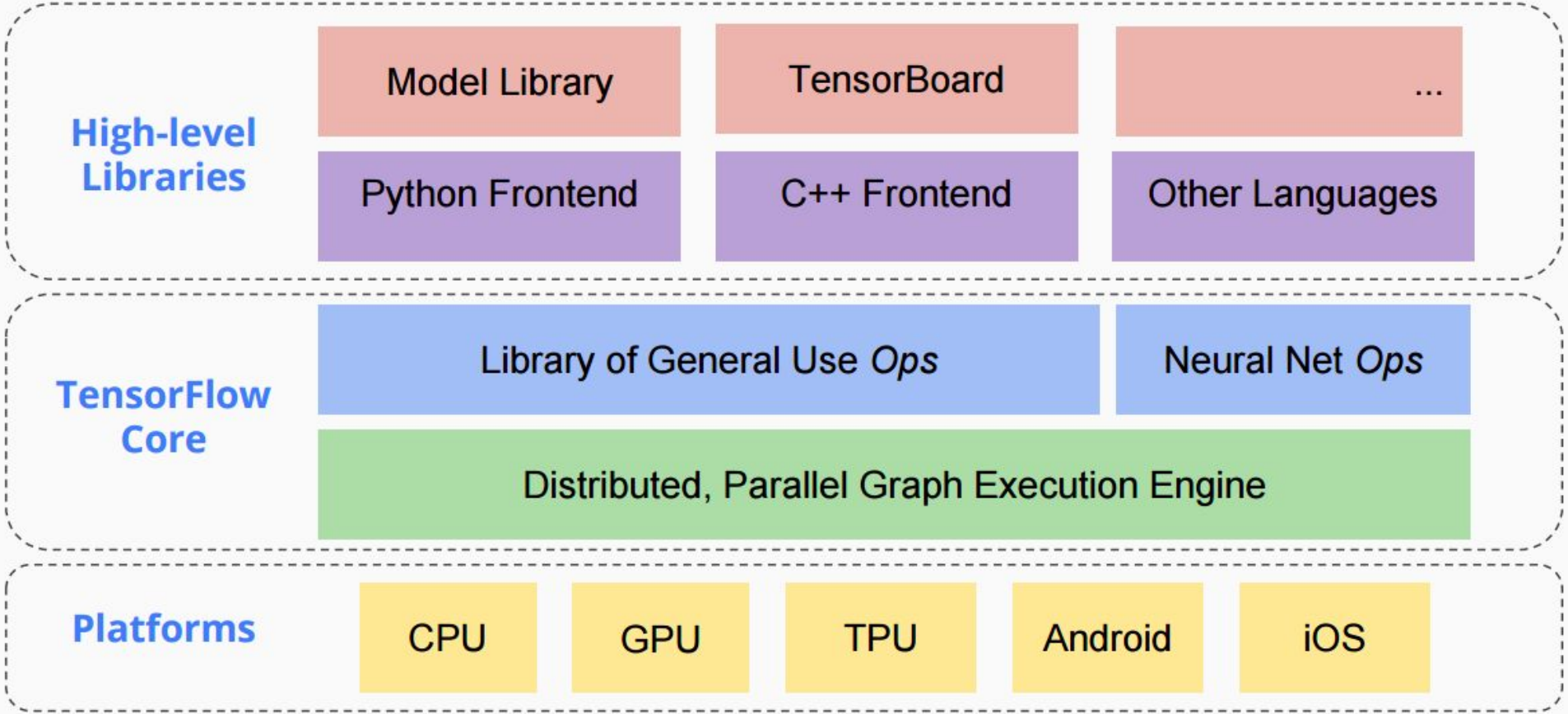
Core TensorFlow Structures

- **Constants**
- **Placeholders**: must be fed with data on execution
- **Variables**: modifiable tensors that live in TensorFlow's graph
- **Session**: encapsulates the environment in which Operations are executed and Tensors are evaluated

Tensorflow Operations

Operation	Description
tf.add	sum
tf.sub	substraction
tf.mul	multiplication
tf.div	division
tf.mod	module
tf.abs	absolute value
tf.neg	negative value
tf.inv	inverse
tf.maximum	returns the maximum
tf.minimum	returns the minimum

Operation	Description
tf.square	calculates the square
tf.round	nearest integer
tf.sqrt	square root
tf.pow	calculates the power
tf.exp	exponential
tf.log	logarithm
tf.cos	calculates the cosine
tf.sin	calculates the sine
tf.matmul	tensor product
tf.transpose	tensor transpose


High-level Libraries
Model Library
TensorBoard
...
Python Frontend
C++ Frontend
Other Languages
TensorFlow Core
Library of General Use *Ops*
Neural Net *Ops*
Distributed, Parallel Graph Execution Engine
Platforms
CPU
GPU
TPU
Android
iOS

“Hello World” Example

```
import tensorflow as tf

# Create a Constant op that produces a 1x2 matrix.  The op is
# added as a node to the default graph.
#
# The value returned by the constructor represents the output
# of the Constant op.
matrix1 = tf.constant([[3., 3.]])

# Create another Constant that produces a 2x1 matrix.
matrix2 = tf.constant([[2.],[2.]])

# Create a Matmul op that takes 'matrix1' and 'matrix2' as inputs.
# The returned value, 'product', represents the result of the matrix
# multiplication.
product = tf.matmul(matrix1, matrix2)
```

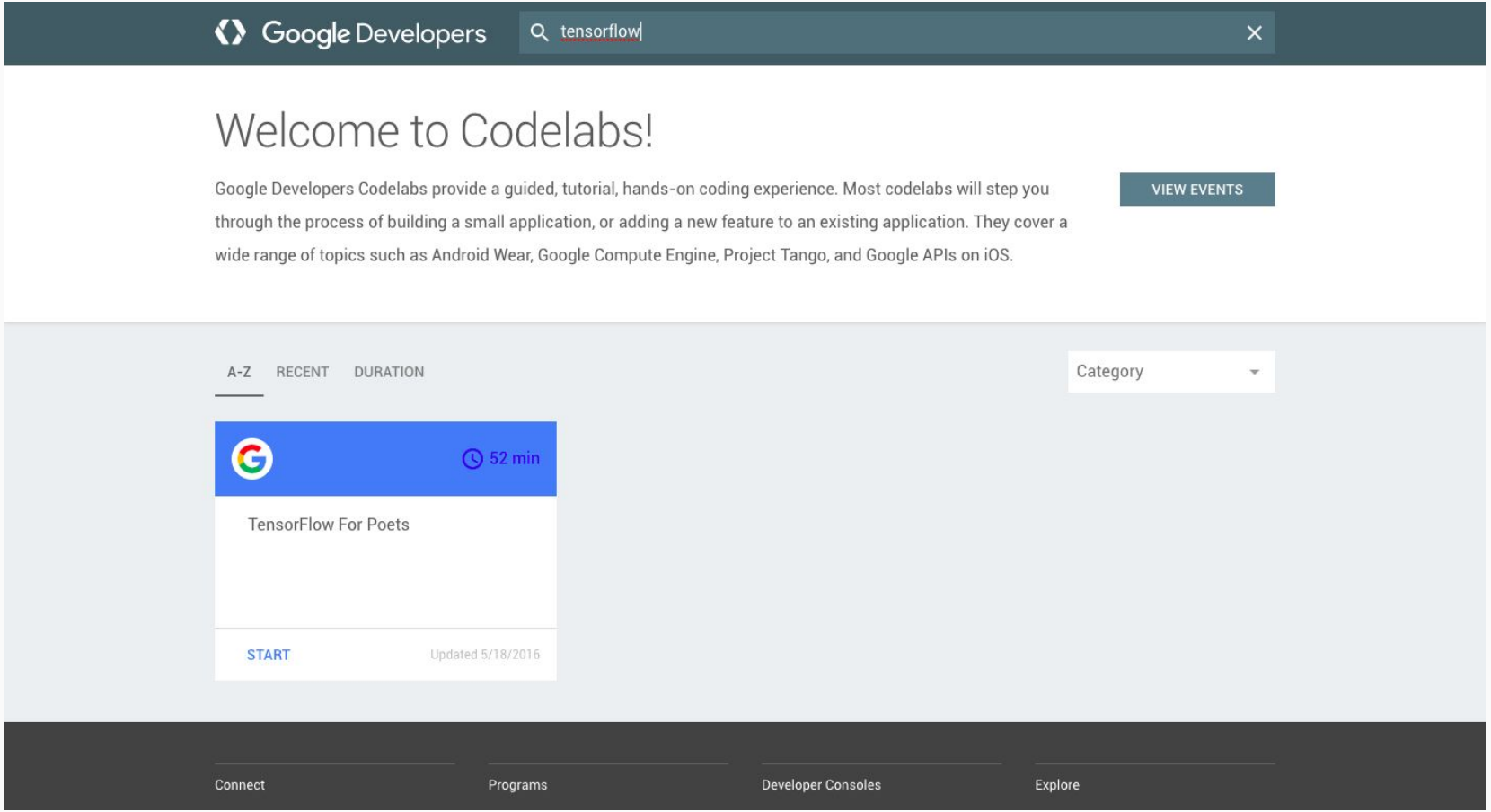
Holy Moly!

It’s not working!

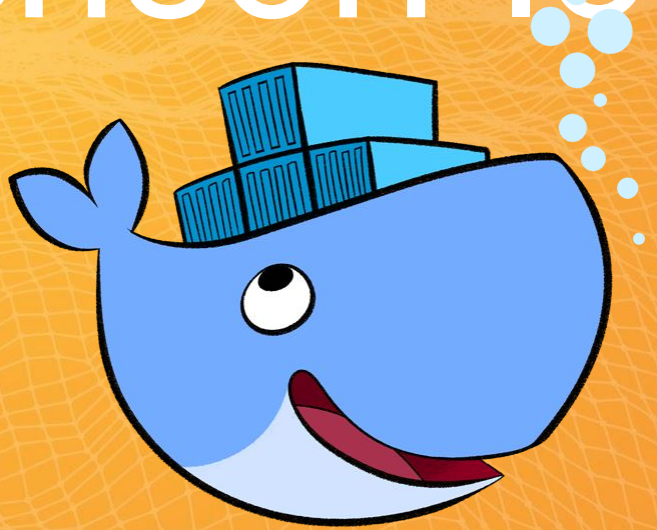
```
# Launch the default graph.
sess = tf.Session()

# To run the matmul op we call the session 'run()' method, passing 'product'
# which represents the output of the matmul op.  This indicates to the call
# that we want to get the output of the matmul op back.
#
# All inputs needed by the op are run automatically by the session.  They
# typically are run in parallel.
#
# The call 'run(product)' thus causes the execution of three ops in the
# graph: the two constants and matmul.
#
# The output of the op is returned in 'result' as a numpy `ndarray` object.
result = sess.run(product)
print(result)
# ==> [[ 12.]]

# Close the Session when we're done.
sess.close()
```


ADVENTURE TIME
CREATED BY PENDLETON WARD

https://codelabs.developers.google.com/



Install TensorFlow using



```
% docker run -it gcr.io/tensorflow/tensorflow:latest-devel
```

Check to see if your TensorFlow works

```
# python


import tensorflow as tf
hello = tf.constant('Hello, TensorFlow!')
sess = tf.Session()
print(sess.run(hello))
```

Retrieve the images

```
# ctrl-D if you're still in Docker and then:
% cd $HOME
% mkdir tf_files
% cd tf_files
% curl -O
http://download.tensorflow.org/example_images/flower_photos.tgz
% tar xzf flower_photos.tgz
```

Start Docker with local files available

```
% docker run -it -v $HOME/tf_files:/tf_files \
 gcr.io/tensorflow/tensorflow:latest-devel
```

Update TensorFlow

```
# cd /tensorflow
# git pull
```


MORE TRAINING
YOU MUST HAVE!...MMM!
quickmeme.com

```
# python tensorflow/examples/image_retraining/retrain.py \
--bottleneck_dir=/tf_files/bottlenecks \
--how_many_training_steps 500 \
--model_dir=/tf_files/inception \
--output_graph=/tf_files/retrained_graph.pb \
--output_labels=/tf_files/retrained_labels.txt \
--image_dir /tf_files/flower_photos
```

You should see something like this

```
Creating bottleneck at /tf_files/bottlenecks/daisy/530738000_4df7e4786b.jpg.txt
Creating bottleneck at /tf_files/bottlenecks/daisy/534547364_3f6b7279d2_n.jpg.txt
3400 bottleneck files created.
Creating bottleneck at /tf_files/bottlenecks/daisy/538920244_59899a78f8_n.jpg.txt
Creating bottleneck at /tf_files/bottlenecks/daisy/5434742166_35773eba57_m.jpg.txt
...
2016-11-02 23:30:52.216856: Step 100: Train accuracy = 78.0%
2016-11-02 23:30:52.217029: Step 100: Cross entropy = 0.680065
2016-11-02 23:30:52.663786: Step 100: Validation accuracy = 85.0%
...
Final test accuracy = 87.0%
Converted 2 variables to const ops.
```


WHAT AM I
DOING
HERE?
memegenerator.net

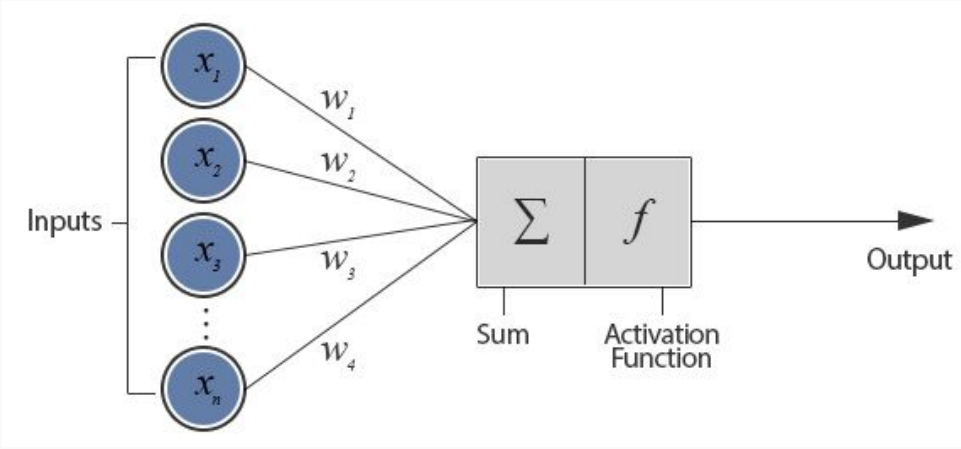
Deep Learning

- A family of Machine Learning algorithms
- No feature engineering needed
- They perform better for problems like:
  - Image Recognition
  - Audio Recognition
  - Natural Language Processing

Perceptron

- Takes ***n*** binary input and produces a single binary output
- For each input x_i there is a weight w_i that determines how relevant the input x_i is to the output
- ***b*** is the bias and defines the activation threshold

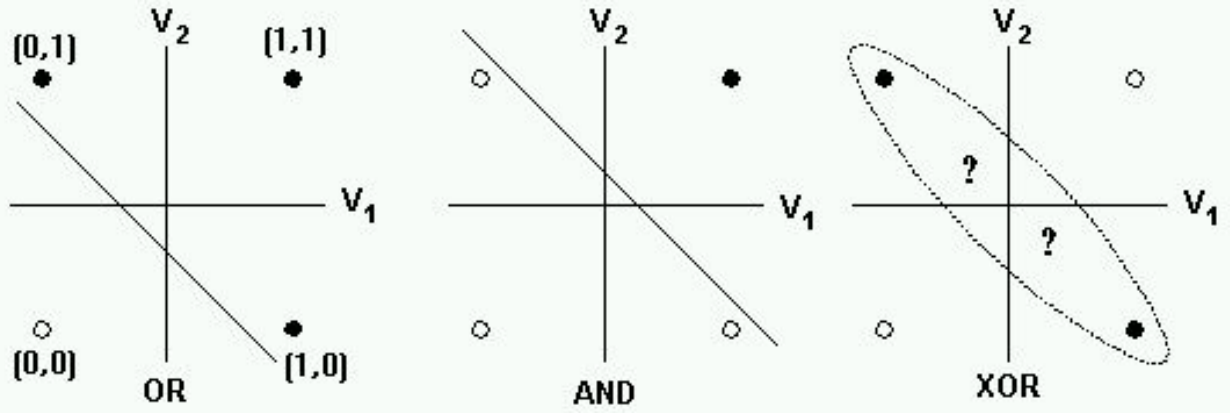
$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



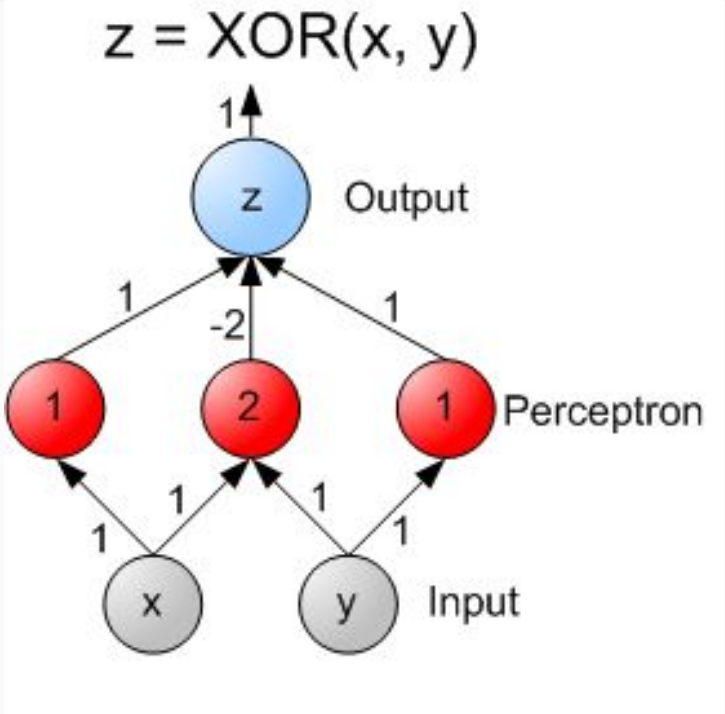
(credits: The Project Spot)

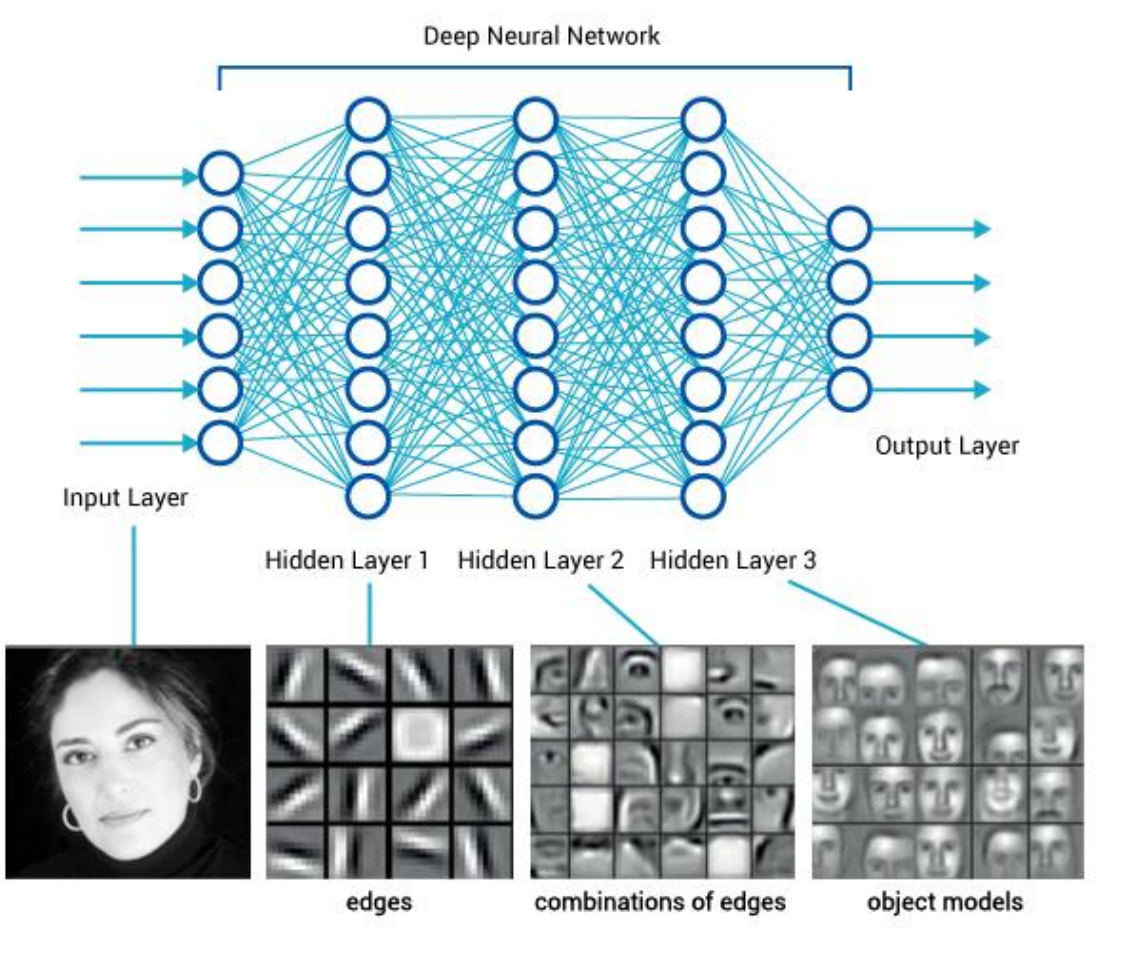
SLP Limitations

A single perceptron cannot solve linearly separable problems!



Introduce more layers!





(credits: Amax Blog)

http://playground.tensorflow.org/

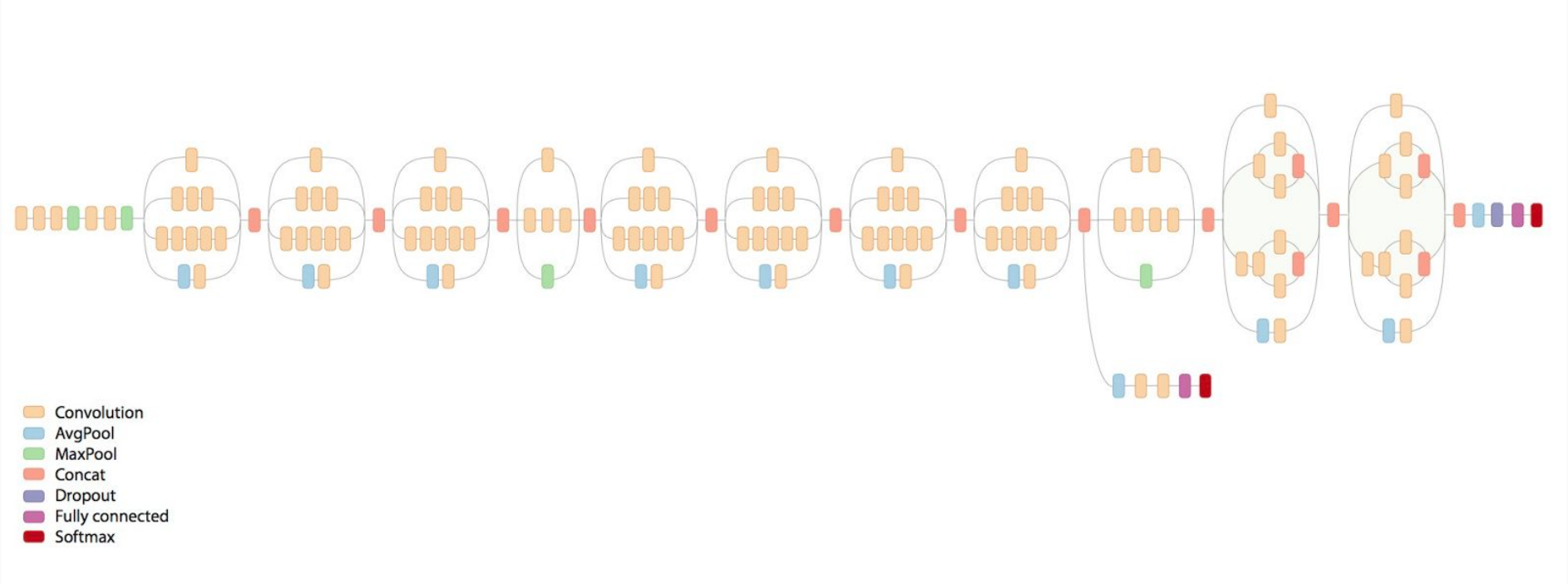

INCEPTION
THE DREAM IS REAL.

It all started with ImageNet...

- The ImageNet project is a large visual database designed for use in visual object recognition software research
- Structured in a hierarchy in which each node is depicted by over five hundred images per node
- As of 2016, over ten million URLs of images have been hand-annotated by ImageNet to indicate what objects are pictured
- Since 2010, the ImageNet project runs an annual software contest where software programs compete to correctly classify and detect objects and scenes.

Enter Inception v3

- Achieves 5.64% top-5 error
- An ensemble of four of these models achieves 3.58% top-5 error on the validation set of the ImageNet
- In the 2015 ImageNet Challenge, an ensemble of 4 of these models came in 2nd in the image classification task



(credits: Google Research Blog)

Let’s train our Inception
(or maybe not)

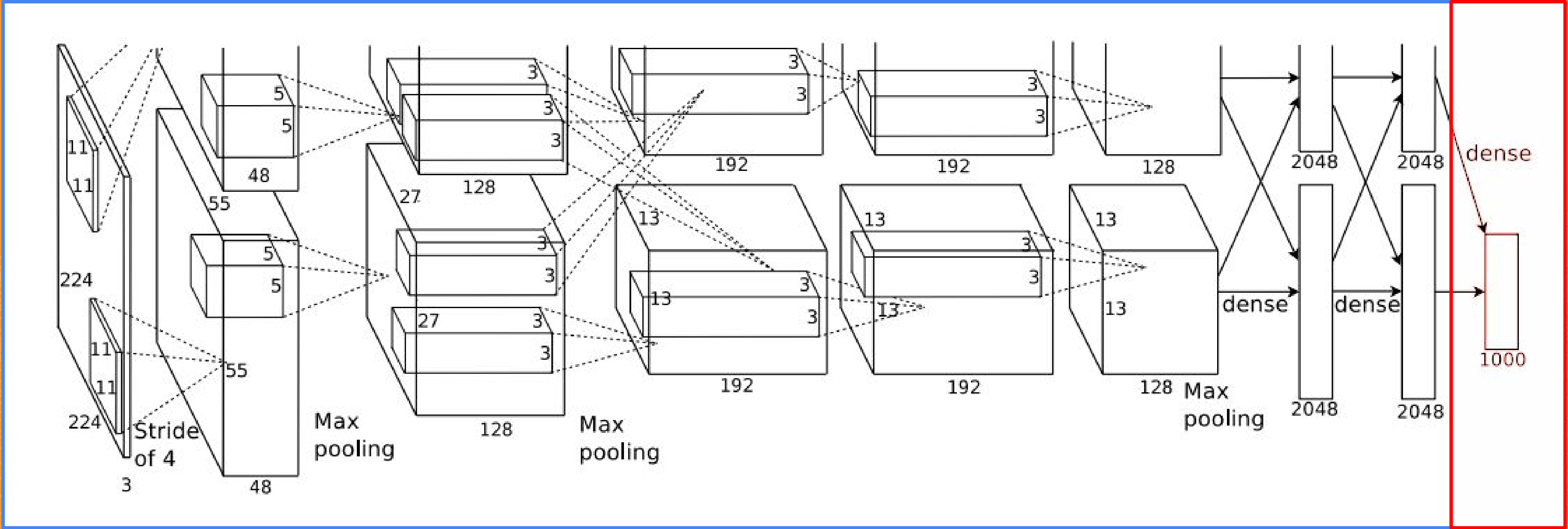
“We can train a model from scratch to its best performance on a desktop with 8 NVIDIA Tesla K40s in about 2 weeks”

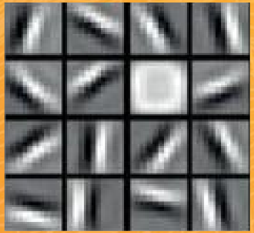
Transfer Learning

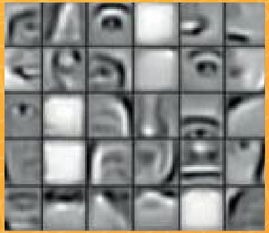
Transfer Learning

“Storing knowledge gained while solving one problem and applying it to a different but related problem

For example, the abilities acquired while learning to walk presumably apply when one learns to run, and knowledge gained while learning to recognize cars could apply when recognizing trucks” - **Wikipedia**


224
224
3
11
11
Stride of 4
55
55
48
48
5
5
Max pooling
27
27
128
128
3
3
Max pooling
13
13
192
192
128
128
dense
dense
2048
2048
2048
2048
dense
1000





```
...
Creating bottleneck at /tf_files/bottlenecks/daisy/530738000_4df7e4786b.jpg.txt
Creating bottleneck at /tf_files/bottlenecks/daisy/534547364_3f6b7279d2_n.jpg.txt
3400 bottleneck files created.
Creating bottleneck at /tf_files/bottlenecks/daisy/538920244_59899a78f8_n.jpg.txt
Creating bottleneck at /tf_files/bottlenecks/daisy/5434742166_35773eba57_m.jpg.txt
...
```


BOTTLENECK
BOTTLENECK
EVERYWHERE
MEMES & FUNNY PICS
FRABZ.COM

Bottlenecks?

- A **Bottleneck** is an informal term used for the layer just before the final output layer that actually does the classification.
- Caches the outputs of the lower layers on disk so that they don't have to be repeatedly recalculated.

```
…
2016-11-02 23:30:52.216856: Step 100: Train accuracy = 78.0%
2016-11-02 23:30:52.217029: Step 100: Cross entropy = 0.680065
2016-11-02 23:30:52.663786: Step 100: Validation accuracy = 85.0%
…
```

Training Accuracy

“The percentage of the images used in the current training batch that were labeled with the correct class”

		Predicted				
		Daisy	Dandelion	Roses	Sunflowers	Tulips
Actual	Daisy	5	1	0	1	0
	Dandelion	2	3	1	2	3
	Roses	0	2	11	0	2
	Sunflowers	1	3	0	5	2
	Tulips	0	3	1	0	3

		Predicted				
		Daisy	Dandelion	Roses	Sunflowers	Tulips
Actual	Daisy	5	1	0	1	0
	Dandelion	2	3	1	2	3
	Roses	0	2	11	0	2
	Sunflowers	1	3	0	5	2
	Tulips	0	3	1	0	3

$$\text{Accuracy} = \frac{\Sigma \text{ True positive} + \Sigma \text{ True negative}}{\Sigma \text{ Total population}}$$

Accuracy = (5 + 41) / (5 + 2 + 3 + 41)

= 0.90196078431

Validation Accuracy

“The percentage of correctly-labelled images on a randomly-selected group of images from a different set”

Cross Entropy

“A loss function that gives a glimpse into how well the learning process is progressing.

Lower numbers are better here.”


TIME FOR
ACTION
memes.com

```
import tensorflow as tf

# change this as you see fit
image_path = sys.argv[1]

# Read in the image_data (the one which will be used for testing the NN)
image_data = tf.gfile.FastGFile(image_path, 'rb').read()

# Loads label file, strips off carriage return
label_lines = [line.rstrip() for line
                   in tf.gfile.GFile("/tf_files/retrained_labels.txt")]

# Import a serialized GraphDef. GraphDef is a Graph definition, saved as a
ProtoBuf
with tf.gfile.FastGFile("/tf_files/retrained_graph.pb", 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    _ = tf.import_graph_def(graph_def, name='')
```

```
with tf.Session() as sess:
    # Feed the image_data as input to the graph and get first prediction
    softmax_tensor = sess.graph.get_tensor_by_name('final_result:0')

    predictions = sess.run(softmax_tensor, \
             {'DecodeJpeg/contents:0': image_data})

    # Sort to show labels of first prediction in order of confidence
    top_k = predictions[0].argsort()[-len(predictions[0]):][::-1]

    for node_id in top_k:
        human_string = label_lines[node_id]
        score = predictions[0][node_id]
        print('%s (score = %.5f)' % (human_string, score))
```

Surprise!
Everything is ready!

```
# ctrl-D to exit Docker and then:
% curl -L https://goo.gl/tx3dqg > $HOME/tf_files/label_image.py
```

```
# Restart your Docker image
% docker run -it -v \
  $HOME/tf_files:/tf_files gcr.io/tensorflow/tensorflow:latest-devel
```

```
# Run the Python file you created on a daisy:


# python /tf_files/label_image.py \
  /tf_files/flower_photos/daisy/21652746_cc379e0eea_m.jpg
```

You should see something like this

daisy (score = 0.99071)

sunflowers (score = 0.00595)

dandelion (score = 0.00252)

roses (score = 0.00049)

tulips (score = 0.00032)

```
# Run the Python file you created on a rose:


# python /tf_files/label_image.py \
  /tf_files/flower_photos/roses/2414954629_3708a1a04d.jpg
```

What do you see?


NOW
IT'S YOUR TURN

Trying Other Parameters

Parameter	Description
--learning_rate	How large a learning rate to use when training.
--train_batch_size	How many images to train on at a time.
--how_many_training_steps	How many training steps to run before ending.
--random_scale	A percentage determining how much to randomly scale up the size of the training images by.
--flip_left_right	Whether to randomly flip half of the training images horizontally.
--random_crop	A percentage determining how much of a margin to randomly crop off the training images.
--random_brightness	A percentage determining how much to randomly multiply the training image input pixels up or down by.
--testing_percentage	What percentage of images to use as a test set.
--validation_percentage	What percentage of images to use as a validation set.

Training on Your Own Categories

What about:

http://www.multimedia-computing.de/flickrlogos/data/

http://vision.stanford.edu/aditya86/ImageNetDogs/

https://gdgmilanoslack.herokuapp.com