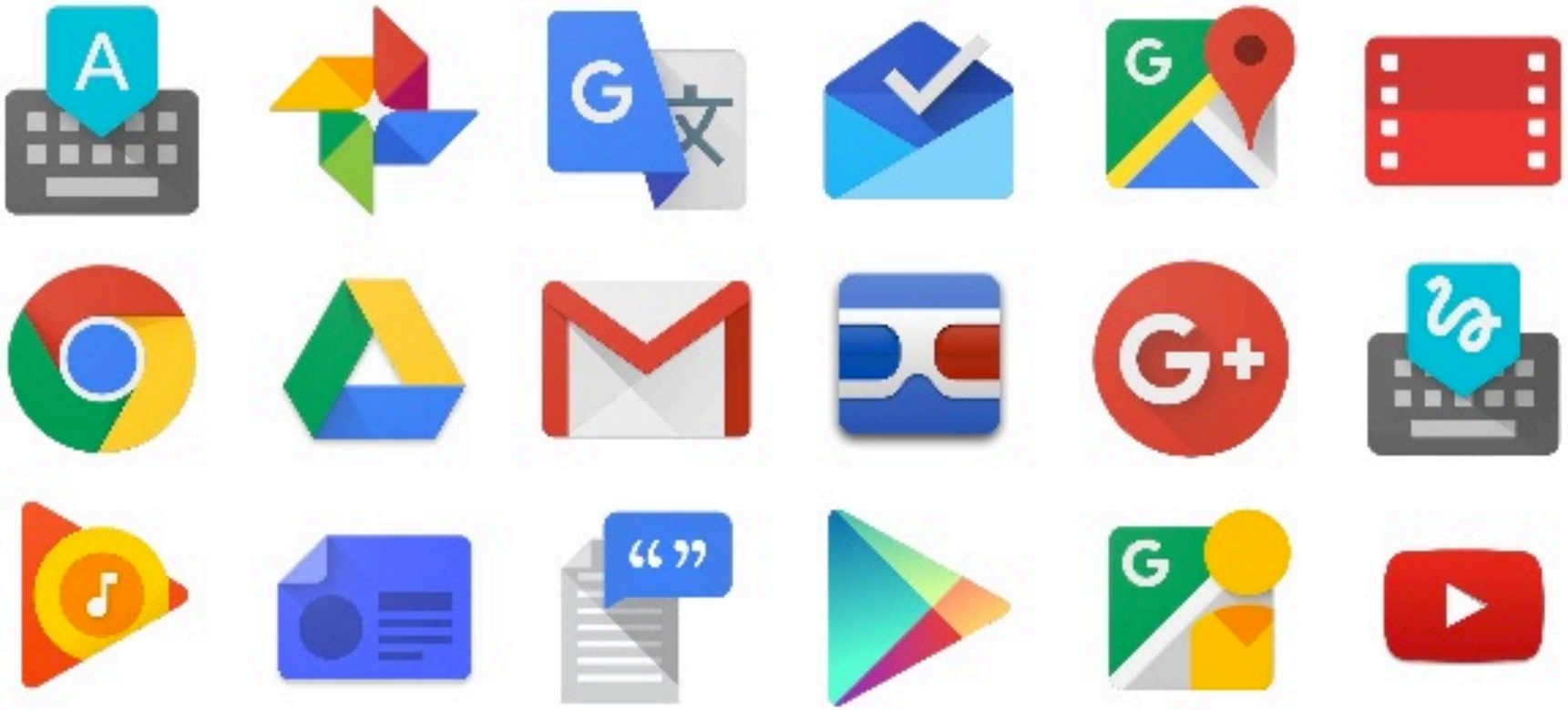
TensorFlow & TensorFlow Mobile

Muhammed Demircan und Christoph Jerolimov

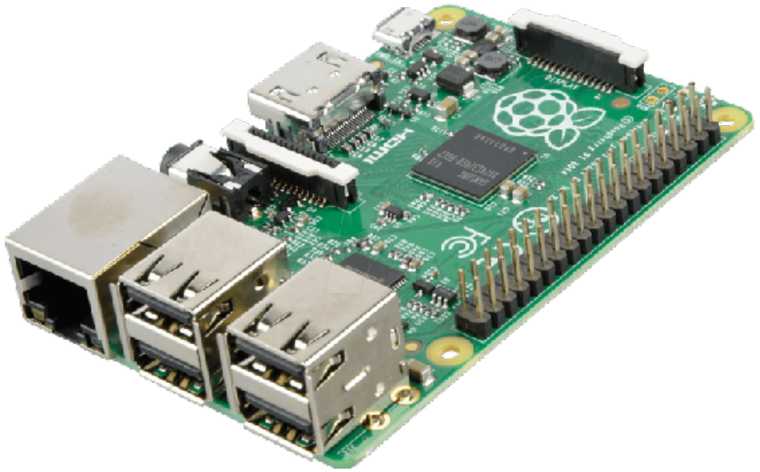
TensorFlow

Products using Machine Learning



TensorFlow

- ~~Machine Learning~~ numerical computing and graph library powered by Google
- Written in C++ and scalable from mobile devices to datacenters



TensorFlow

- Written in C++
- Bindings for Python, Haskell and Rust (Go / Java beta)
- Low level stable core API
- Highlevel APIs:
  - TensorFlow .layer and .train packages
  - TensorLayer
  - Keras

TensorFlow Dev Summit

- TensorFlow 1.0
- TensorBoard
- XLA: TensorFlow compiler
  - Just-in-time compilation
  - Ahead-of-rime compilation

Everything is a Tensor or a Graph

- Tensors are **n-dimensional arrays**
  - Number - a rank 0 tensor
  - Vector - a rank 1 tensor
  - Matrix - a rank 2 tensor
- **Construction phase:** Define a computational graph with tensors, node and edges
- **Execution phase:** Evaluate the graph with inputs with an “session”

Hello, World.


```
import tensorflow as tf

a = tf.constant(3.0, tf.float32)
b = tf.constant(4.0) # also tf.float32 implicitly

s = tf.add(a, b)     # or a + b

sess = tf.Session()

print("Tensor a + b: ", s)
# <tf.Tensor 'Add:0' shape=() dtype=float32>)

print("Result a + b: ", sess.run(s))
# 7.0
```

Hello, World.

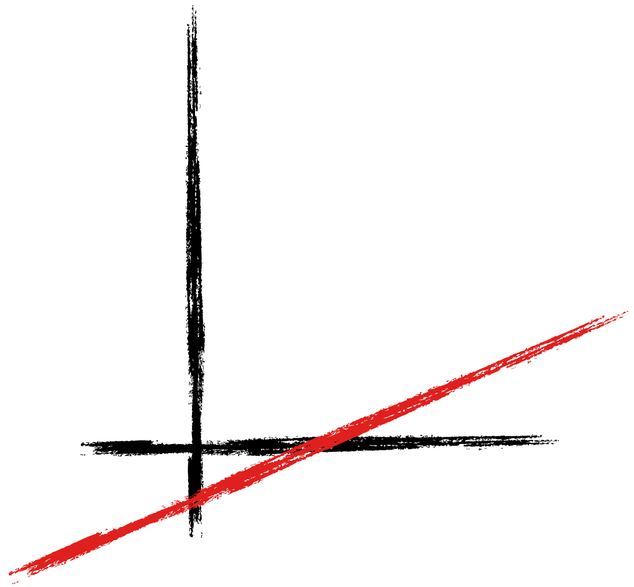

```
import tensorflow as tf

a = tf.constant(3.0, tf.float32)
b = tf.constant(4.0) # also tf.float32 implicitly

s = tf.add(a, b)     # or a + b

sess = tf.Session()

print("Tensor a + b: ", s)
# <tf.Tensor 'Add:0' shape=() dtype=float32>)

print("Result a + b: ", sess.run(s))
# 7.0
```

Tensor types

- Each tensor has a shape and value-type (float, double, ..)
- tf.constant()
  Constant and immutable value.
- tf.placeholder()
  Dummy node, not trainable, takes values in the session.
- tf.Variable()
  Variable and **trainable** value with an **initial value**.

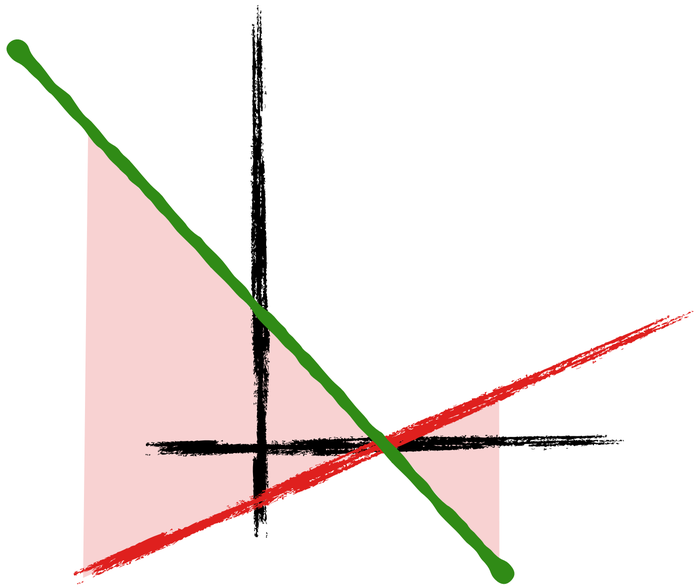
Linear Algebra

```
# Model parameters – our expectation!
W = tf.Variable([  0.3 ], tf.float32)
b = tf.Variable([ -0.3 ], tf.float32)

# Model input and output
x = tf.placeholder(tf.float32)
y = tf.placeholder(tf.float32)

linear_model = W * x + b
```



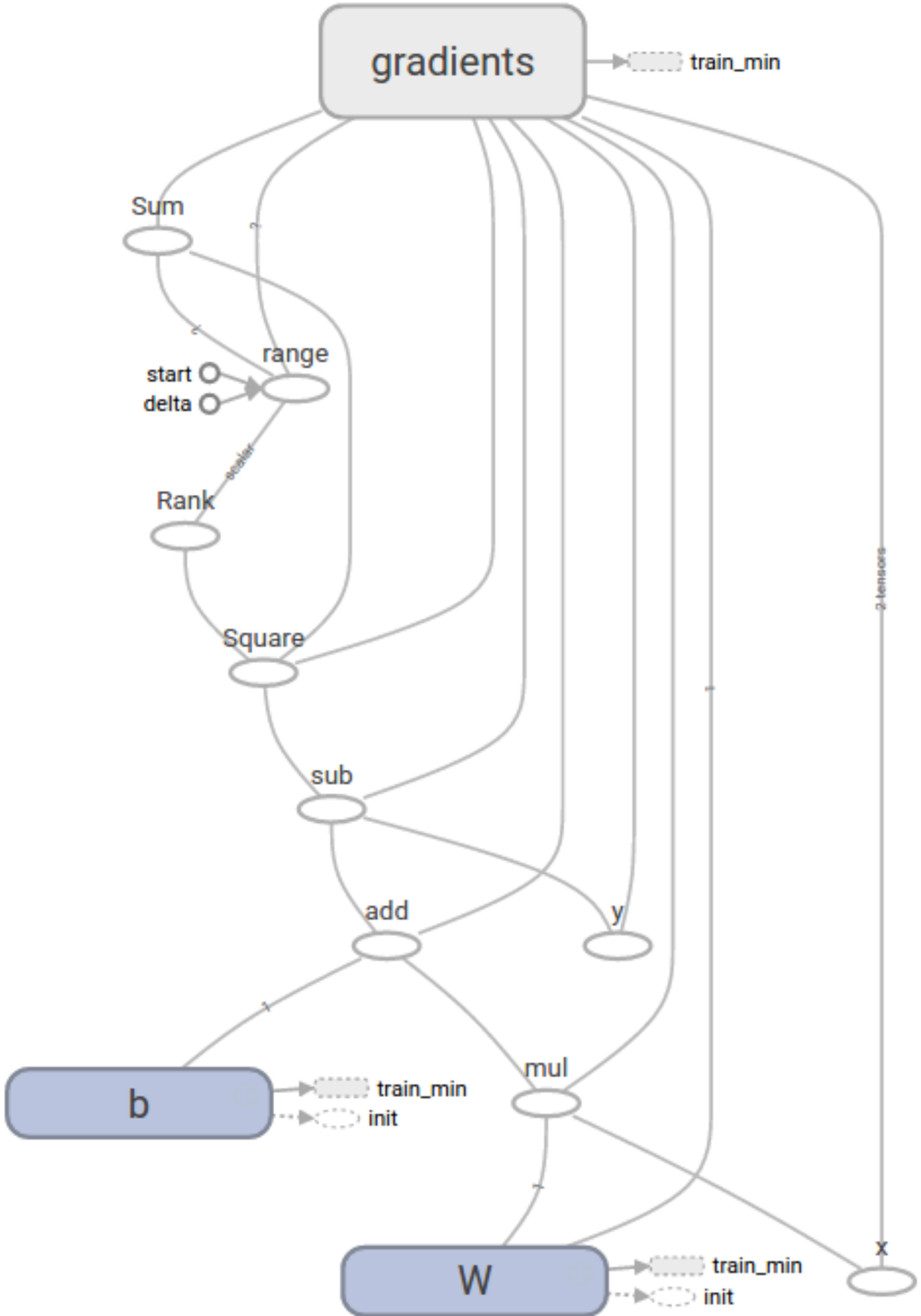
Linear Algebra

```
# Training data
x_train = [1,  2,  3,  4]
y_train = [0, -1, -2, -3]

# Loss
loss = tf.reduce_sum(tf.square(linear_model - y))
```



Linear Algebra

```
# Optimizer
optimizer = tf.train.GradientDescentOptimizer(0.01)
train = optimizer.minimize(loss)

for i in range(1000):
    sess.run(train, {x: x_train, y: y_train})

# Evaluate training accuracy
curr_W, curr_b, curr_loss = sess.run(
  [W, b, loss], {x: x_train, y: y_train}
)

print("W: %s b: %s loss: %s"%(curr_W, curr_b, curr_loss))
# W: [-0.99999] b: [ 0.99999] loss: 5.69997e-11
```


gradients
train_min
Sum
range
start
delta
Rank
scalar
Square
sub
add
y
b
train_min
init
mul
W
train_min
init
x
2-tensors


W
gradients
b
train_min


W
b
init

Fit to screen

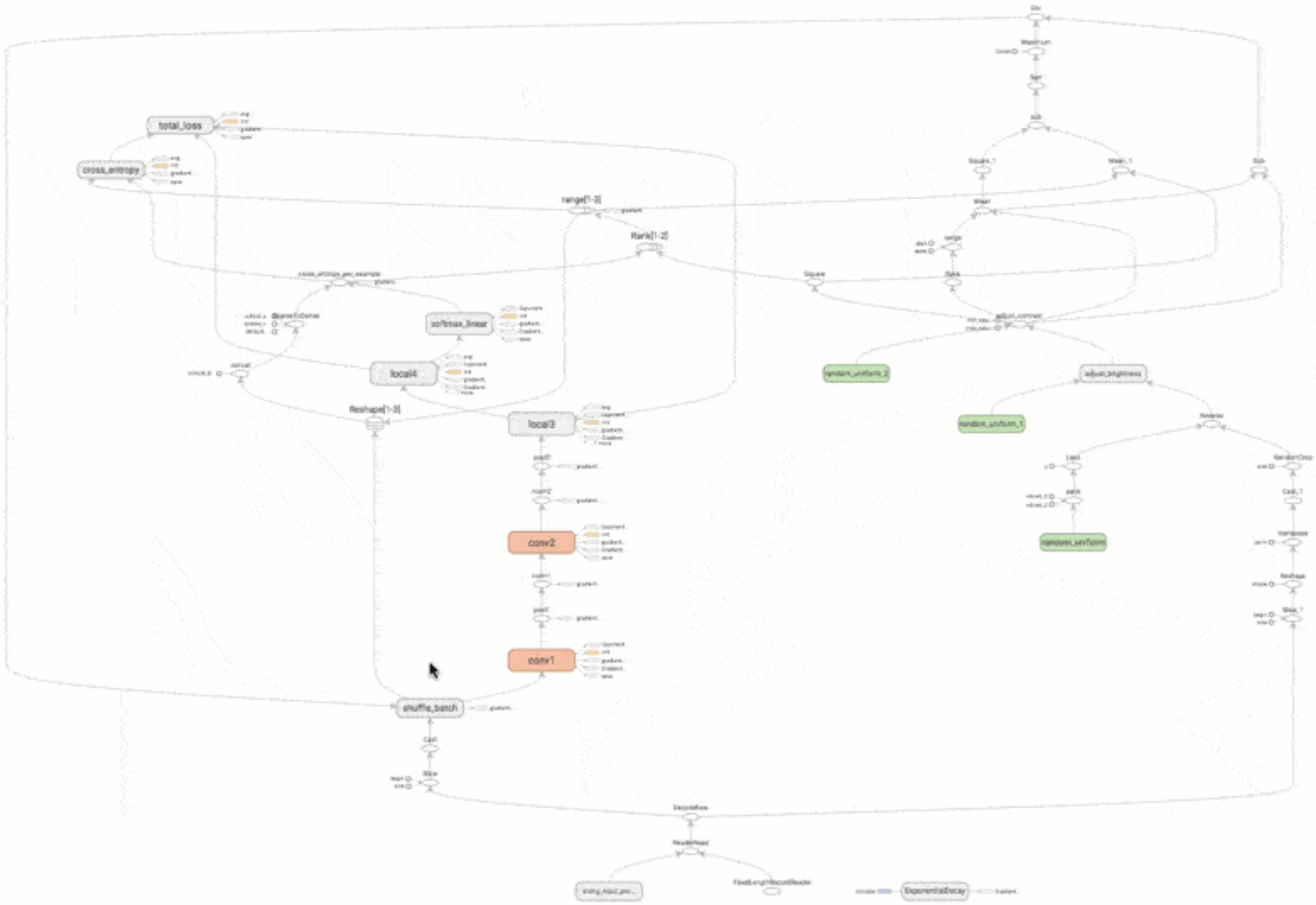
Run cifar-train

Upload Choose File

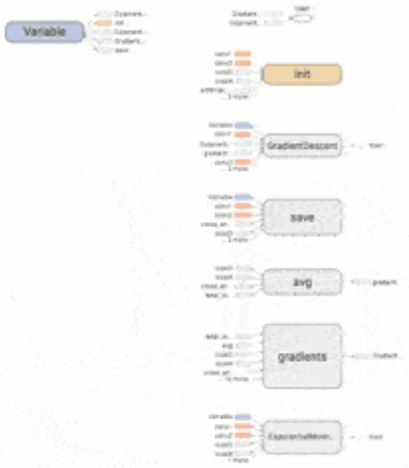
Color Structure

color: same substructure
gray: unique substructure

Main Graph



Auxiliary nodes



Graph (* = expandable)

- Namespace*
- OpNode
- Unconnected series*
- Connected series*
- Constant
- Summary
- Dataflow edge
- Control dependency edge
- Reference edge

TensorFlow Mobile

TensorF

swift

TensorSwift

- Written in Swift in less than 1.000 LOC
- Can classify “things” based on a learned NN.
- Pure swift, on iOS/OSX with native performance improvements

TensorSwift

```
# MINTS example NN:

W_conv1 = Tensor(shape: [5, 5, 1, 32], elements: loadArray(path, file: "W_conv1"))
b_conv1 = Tensor(shape: [32], elements: loadArray(path, file: "b_conv1"))
W_conv2 = Tensor(shape: [5, 5, 32, 64], elements: loadArray(path, file: "W_conv2"))
b_conv2 = Tensor(shape: [64], elements: loadArray(path, file: "b_conv2"))

W_fc1 = Tensor(shape: [Dimension(7 * 7 * 64), 1024], elements: loadArray(path, file: "W_fc1"))
b_fc1 = Tensor(shape: [1024], elements: loadArray(path, file: "b_fc1"))
W_fc2 = Tensor(shape: [1024, 10], elements: loadArray(path, file: "W_fc2"))
b_fc2 = Tensor(shape: [10], elements: loadArray(path, file: "b_fc2"))
```

TensorSwift

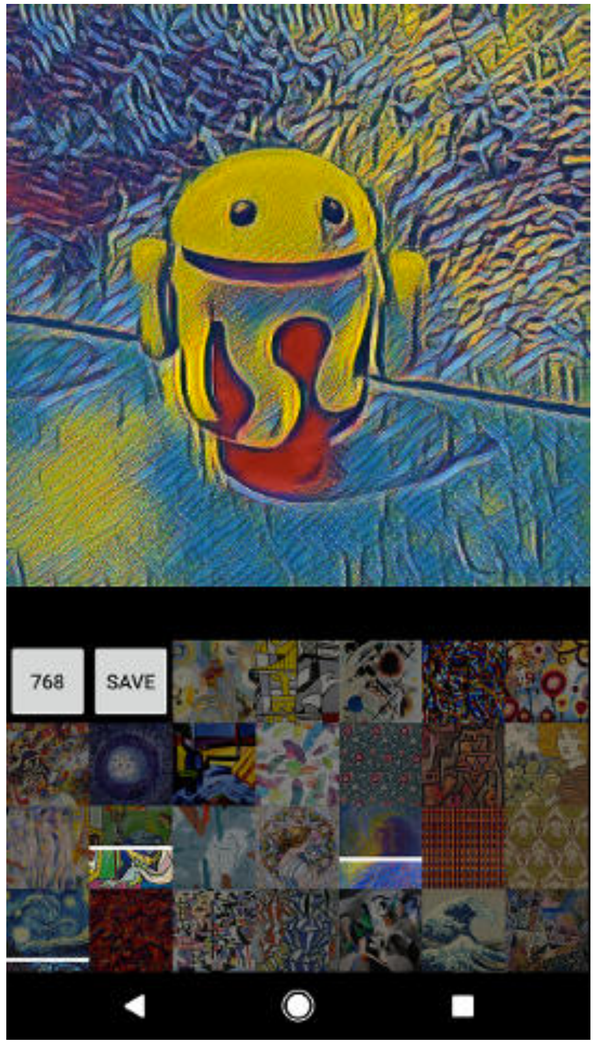
```
public func classify(_ x_image: Tensor) -> (Int, Float) {
    let h_conv1 = (x_image.conv2d(filter: W_conv1, strides: [1, 1, 1]) + b_conv1).relu()
    let h_pool1 = h_conv1.maxPool(kernelSize: [2, 2, 1], strides: [2, 2, 1])

    let h_conv2 = (h_pool1.conv2d(filter: W_conv2, strides: [1, 1, 1]) + b_conv2).relu()
    let h_pool2 = h_conv2.maxPool(kernelSize: [2, 2, 1], strides: [2, 2, 1])

    let h_pool2_flat = h_pool2.reshaped([1, Dimension(7 * 7 * 64)])
    let h_fc1 = (h_pool2_flat.matmul(W_fc1) + b_fc1).relu()

    let y_conv = (h_fc1.matmul(W_fc2) + b_fc2).softmax()

    print("elements: \(y_conv.elements)")

    return y_conv.elements.enumerated().max { $0.1 < $1.1 }!
}
```

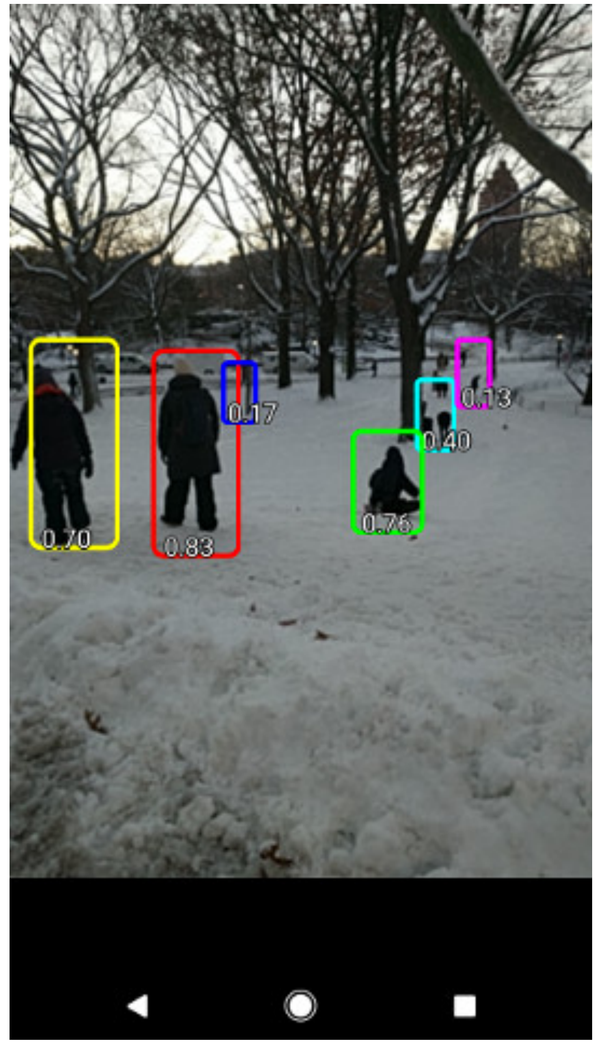
TensorFlow Mobile

TensorFlow Mobile

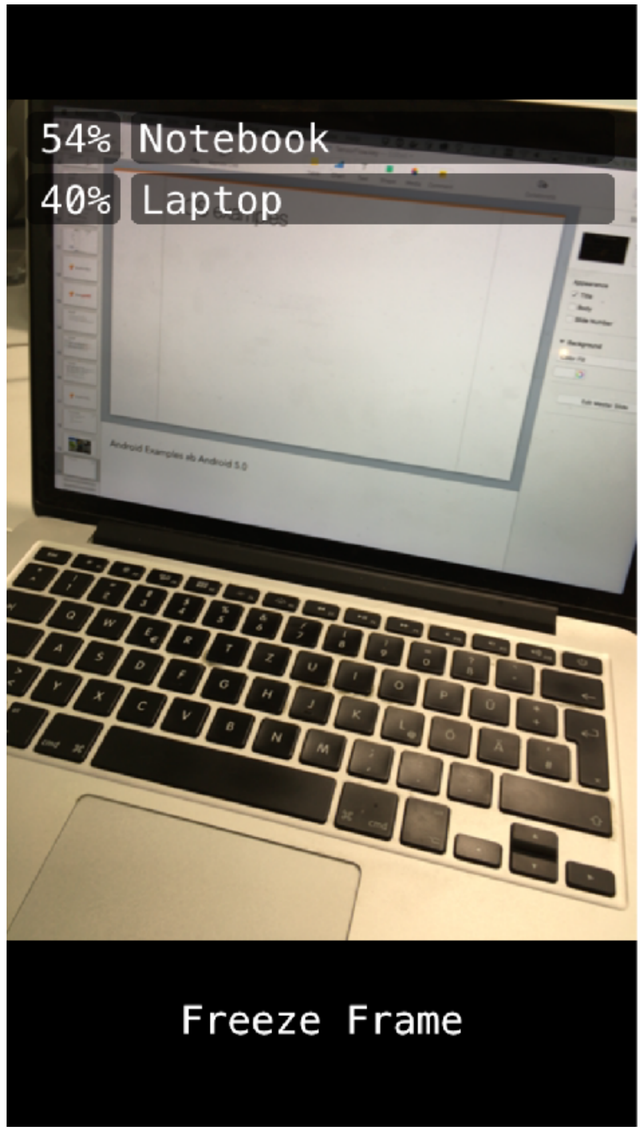
- Use the TensorFlow Core (written in C++)
  - Android 4.0.1+ (API level 14)
    - NDK
    - Bazel Buildsystem (by Google)
  - Xcode

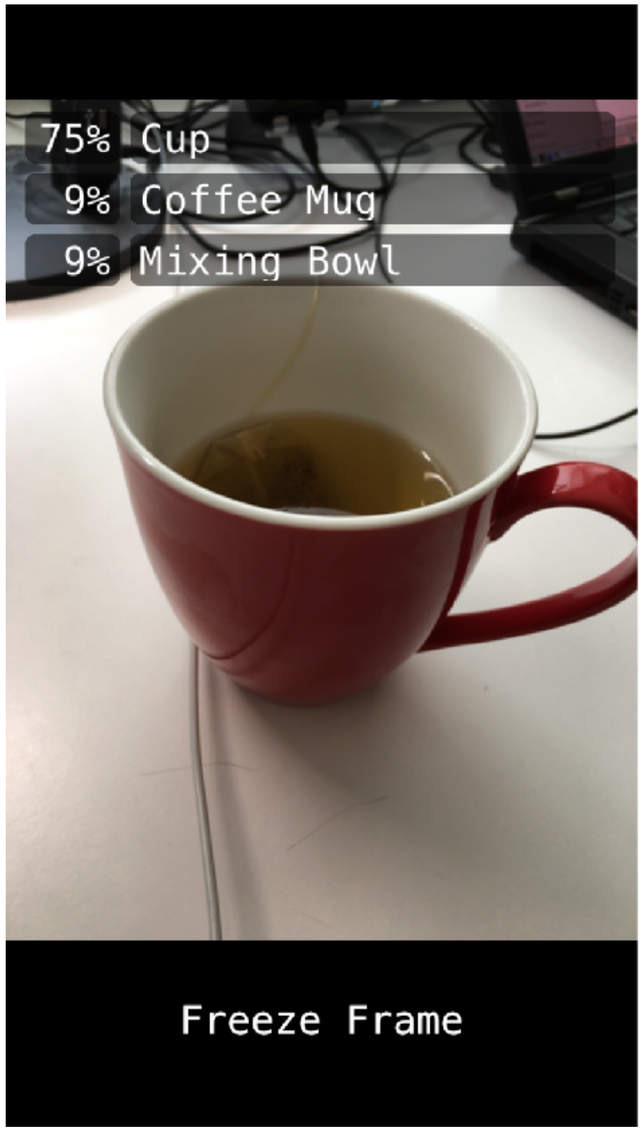
Android examples





iOS examples





More resources

- https://www.tensorflow.org/
- https://www.oreilly.com/learning/hello-tensorflow
- https://www.youtube.com/user/GoogleDevelopers
- TensorFlow Dev Summit 2017 Playlist
  https://www.youtube.com/playlist?list=PLOU2XLYxmsIKGc_NBoIhTn2Qhraji53cv
- Bayesian Deep Learning Workshop NIPS 2016
  https://www.youtube.com/channel/UC_LBLWLfKk5rMKDOHoO7vPQ