TensorFlow on Android

“freedom” Koan-Sin Tan
freedom@computer.org
Aug, 6th, 2017
COSCUP 2017, Taipei, Taiwan

Who Am I

- A software engineer working for a SoC company
- An old open source user, learned to use Unix on a VAX-11/780 running 4.3BSD
- Learned a bit about TensorFlow and how it works on Android
    - Send a couple of PRs, when I was learning to use TensorFlow to classify image

TensorFlow

- “An open-source software library for Machine Intelligence”
    - An open-source library for deep neural network learning
- https://www.tensorflow.org/

tensorflow / tensorflow

Watch 5,807 | Star 65,519 | Fork 32,011

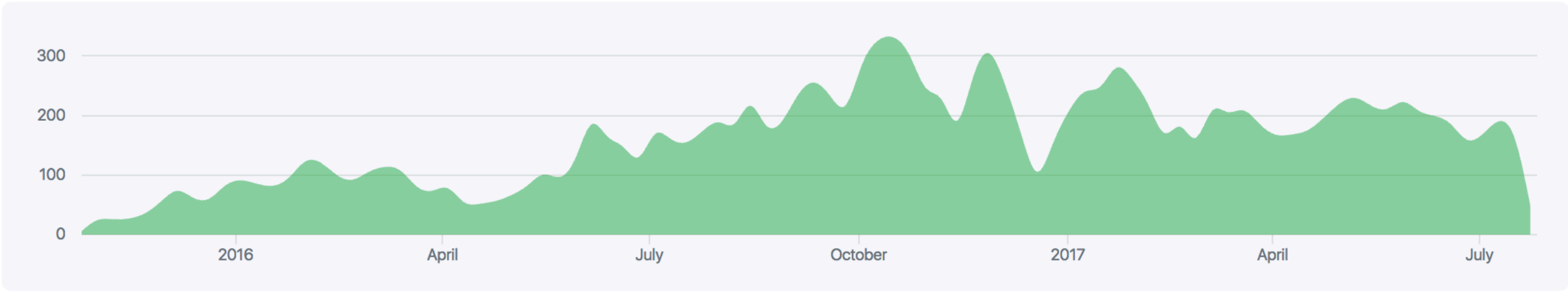
Code | Issues 871 | Pull requests 87 | Projects 0 | Insights

Contributors | Commits | Code frequency | Punch card | Network | Members | Dependents

Nov 1, 2015 – Aug 2, 2017

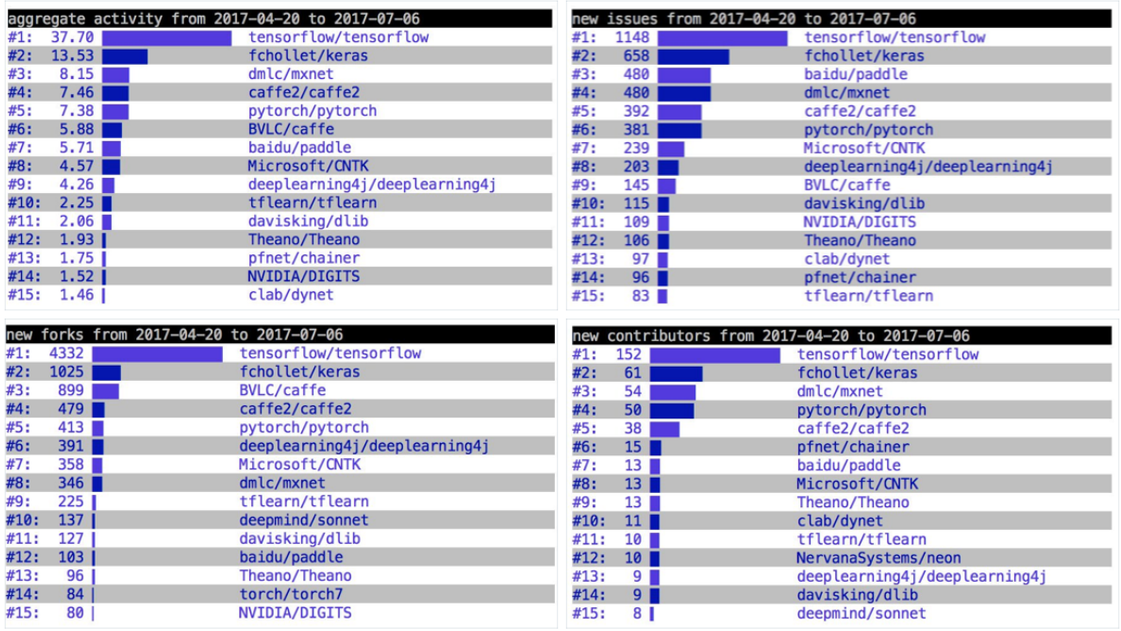
Contributions: Commits

Contributions to master, excluding merge commits



https://github.com/tensorflow/tensorflow/graphs/contributors

The state of the deep learning landscape: GitHub activity of major libraries over the past quarter (tickets, forks, and contributors).



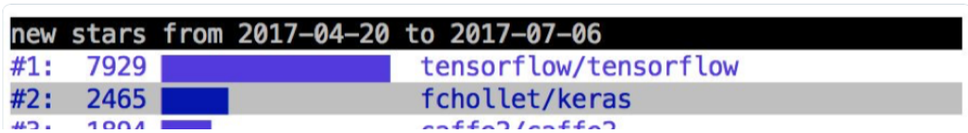
7/7/17, 12:12 AM

234 Retweets **440** Likes

François Chollet @fchollet · 7/7/17

Replying to @fchollet

And these are new GitHub stars over the period. Probably the noisiest of all metrics.



Filters ▾ | author:freedomtan | Labels | Milestones | New pull request

Clear current search query, filters, and sorts

0 Open ✓ 8 Closed | Author ▾ | Labels ▾ | Projects ▾ | Milestones ▾ | Reviews ▾ | Assignee ▾ | Sort ▾

Update the instruction for building a minimal xla benchmark ✓ awaiting testing (then merge) cla: yes — 8
#10387 by freedomtan was merged on Jun 16 • Approved

make gcc-5 on Ubuntu 16.04 happy ✗ cla: yes — 2
#10385 by freedomtan was merged on Jun 3 • Approved

Replace use of tensorflow::ops::ReadFile in label_image ✓ cla: yes — 3
#10034 by freedomtan was merged on May 20 • Approved

gif decoder returns 4-D tensor, remove the first dim ✓ cla: yes stat:awaiting response — 6
#9877 by freedomtan was merged on May 17 • Approved

There is no --logtostderror ● cla: yes — 3
#9830 by freedomtan was merged on May 12 • Approved

fix some trivial typos ● cla: yes — 1
#9565 by freedomtan was merged on May 2 • Approved

Add a simple BMP decoder and enable it on Android ✓ awaiting testing (then merge) cla: yes — 87
#9563 by freedomtan was merged on May 18 • Approved

a python implmentation of label_image ✓ awaiting testing (then merge) cla: yes — 18
#9261 by freedomtan was merged on Jun 27 • Approved

- My first impression of TensorFlow

  - Hey, that's scary. How come you see some many compiler warnings when building such popular open-source library

  - think about: WebKit, llvm/clang, linux kernel, etc.

  - Oh, Google has yet another build system and it's written in Java

How TensorFlow Works

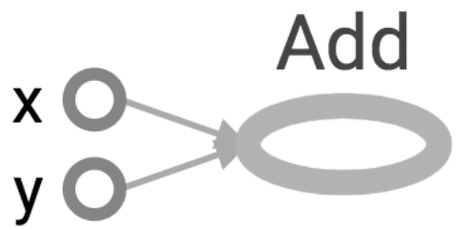
- TensorFlow is dataflow programming
  - a program modeled as an acyclic directional graph
  - node/vertex: operation
  - edge: flow of data (tensor in TensorFlow)
  - operations don’t execute right away
  - operations execute when data are available to ALL inputs

```
In [1]: import tensorflow as tf

In [2]: node1 = tf.constant(3.0)
   ...: node2 = tf.constant(4.0)
   ...: print(node1, node2)
   ...:
(<tf.Tensor 'Const:0' shape=()
dtype=float32>, <tf.Tensor 'Const_1:0'
shape=() dtype=float32>)

In [3]: sess = tf.Session()
   ...: print(sess.run([node1,
node2]))
   ...:
[3.0, 4.0]

In [4]: a = tf.add(3, 4)
   ...: print(sess.run(a))
   ...:
7
```



TensorFlow on Android

- https://www.tensorflow.org/mobile/
  - ongoing effort “to reduce the code footprint, and supporting quantization and lower precision arithmetic that reduce model size”
- Looks good
  - some questions
    - how to build ARMv8 binaries, with latest NDK?
      - --cxxopt="-std=c++11" --cxxopt="-Wno-c++11-narrowing" --cxxopt=“-DTENSORFLOW_DISABLE_META”
- Inception models (e.g., V3) are relatively slow on Android devices
- is there any benchmark or profiling tool?
- it turns out YES

README.md

TensorFlow Android Camera Demo

This folder contains an example application utilizing TensorFlow for Android devices.

Description

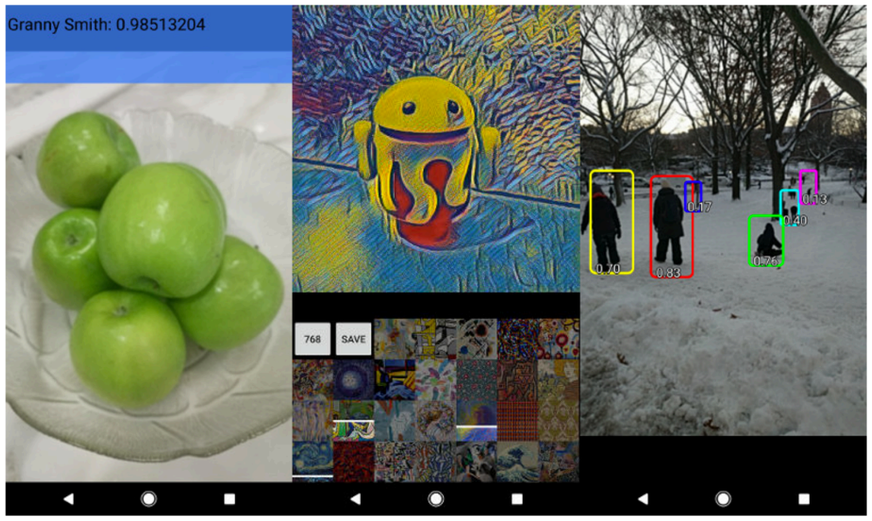
The demos in this folder are designed to give straightforward samples of using TensorFlow in mobile applications.

Inference is done using the TensorFlow Android Inference Interface, which may be built separately if you want a standalone library to drop into your existing application. Object tracking and efficient YUV -> RGB conversion are handled by `libtensorflow_demo.so`.

A device running Android 5.0 (API 21) or higher is required to run the demo due to the use of the camera2 API, although the native libraries themselves can run on API >= 14 devices.

Current samples:

1. TF Classify: Uses the Google Inception model to classify camera frames in real-time, displaying the top results in an overlay on the camera image.
2. TF Detect: Demonstrates a model based on Scalable Object Detection using Deep Neural Networks to localize and track people in the camera preview in real-time.
3. TF Stylize: Uses a model based on A Learned Representation For Artistic Style to restyle the camera preview image to that of a number of different artists.



- `bazel build -c opt --linkopt="-ldl" --cxxopt="-std=c++11" --cxxopt="-Wno-c++11-narrowing" --cxxopt="-DTENSORFLOW_DISABLE_META" --crosstool_top=//external:android/crosstool --cpu=arm64-v8a --host_crosstool_top=@bazel_tools//tools/cpp:toolchain //tensorflow/examples/android:tensorflow_demo --fat_apk_cpu=arm64-v8a`

- `bazel build -c opt --cxxopt="-std=c++11" --cxxopt="-DTENSORFLOW_DISABLE_META" --crosstool_top=//external:android/crosstool --cpu=arm64-v8a --host_crosstool_top=@bazel_tools//tools/cpp:toolchain //tensorflow/examples/android:tensorflow_demo --fat_apk_cpu=arm64-v8a`

  - in case you wanna know how to do it for with older NDK

- The TensorFlow benchmark can benchmark a compute graph and its individual options
  - both on desktop and **Android**
- however, it doesn't deal with real input(s)

README.md

TensorFlow Model Benchmark Tool

Description

A simple C++ binary to benchmark a compute graph and its individual operators, both on desktop mach
Android.

To build/install/run

On Android:

(0) Refer to https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/android to edit
to configure the android NDK/SDK.

(1) build for your specific platform, e.g.:

```
$bazel build -c opt \
  --crosstool_top=//external:android/crosstool \
  --cpu=armeabi-v7a \
  --host_crosstool_top=@bazel_tools//tools/cpp:toolchain \
  tensorflow/tools/benchmark:benchmark_model
```

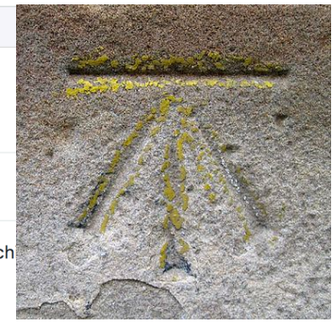
(2) Connect your phone. Push the binary to your phone with adb push (make the directory if required):

```
$adb push bazel-bin/tensorflow/tools/benchmark/benchmark_model /data/local/tmp
```

(3) Push the compute graph that you need to test. For example: adb push tensorflow_inception_graph.pb
/data/local/tmp

(4) Run the benchmark. For example:

```
$adb shell "/data/local/tmp/benchmark_model \
  --graph=/data/local/tmp/tensorflow_inception_graph.pb \
  --input_layer="input:0" \
  --input_layer_shape="1,224,224,3" \
  --input_layer_type="float" \
  --output_layer="output:0"
```



- I saw label_image when reading an article on quantization
  - label_image didn't build for Android
  - still image decoders (jpg, png, and gif) are not included
- So,
  - made it run
  - added a quick and dirty BMP decider
- To hack more quickly (compiling TensorFlow on MT8173 board running Debian is slow), I wrote a Python script to mimic what the C++ program does

[1] https://github.com/tensorflow/tensorflow/tree/master/tensorflow/examples/label_image

TensorFlow C++ and Python Image Recognition Demo

This example shows how you can load a pre-trained TensorFlow network and use it to recognize objects in images in C++. For Java see the Java README, and for Go see the godoc example.

Description

This demo uses a Google Inception model to classify image files that are passed in on the command line.

To build/install/run

The TensorFlow `GraphDef` that contains the model definition and weights is not packaged in the repo because of its size. Instead, you must first download the file to the `data` directory in the source tree:

```
$ curl -L "https://storage.googleapis.com/download.tensorflow.org/models/inception_v3_2016_08_28_frozen.pb
  tar -C tensorflow/examples/label_image/data -xz
```

Then, as long as you've managed to build the main TensorFlow framework, you should have everything you need to run this example installed already.

Once extracted, see the labels file in the data directory for the possible classifications, which are the 1,000 categories used in the Imagenet competition.

To build it, run this command:

```
$ bazel build tensorflow/examples/label_image/...
```

That should build a binary executable that you can then run like this:

```
$ bazel-bin/tensorflow/examples/label_image/label_image
```

This uses the default example image that ships with the framework, and should output something similar to this:

```
I tensorflow/examples/label_image/main.cc:206] military uniform (653): 0.834306
I tensorflow/examples/label_image/main.cc:206] mortarboard (668): 0.0218692
I tensorflow/examples/label_image/main.cc:206] academic gown (401): 0.0103579
I tensorflow/examples/label_image/main.cc:206] pickelhaube (716): 0.00800814
I tensorflow/examples/label_image/main.cc:206] bulletproof vest (466): 0.00535088
```

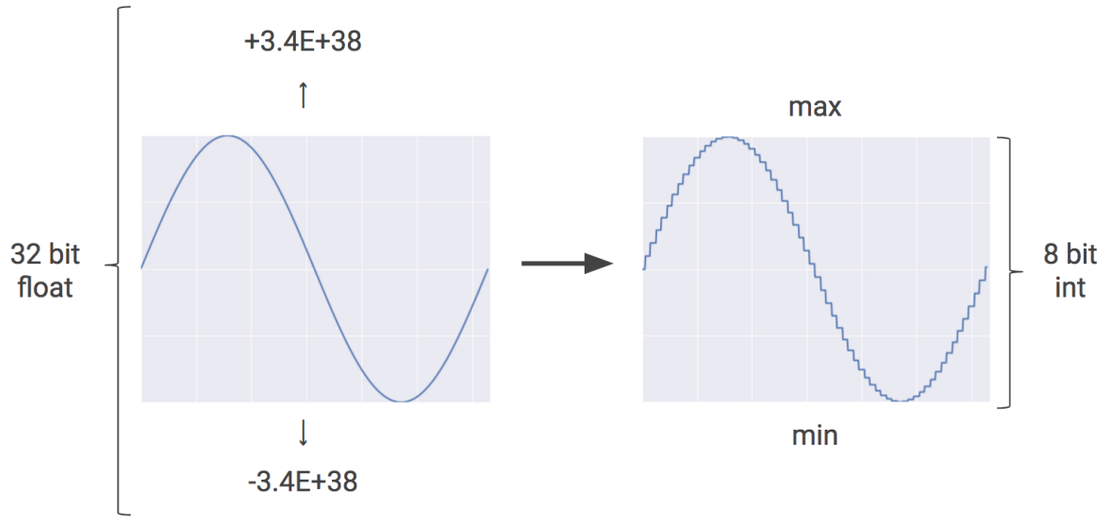
In this case, we're using the default image of Admiral Grace Hopper, and you can see the network correctly spots she's wearing a military uniform, with a high score of 0.8.

Next, try it out on your own images by supplying the --image= argument, e.g.

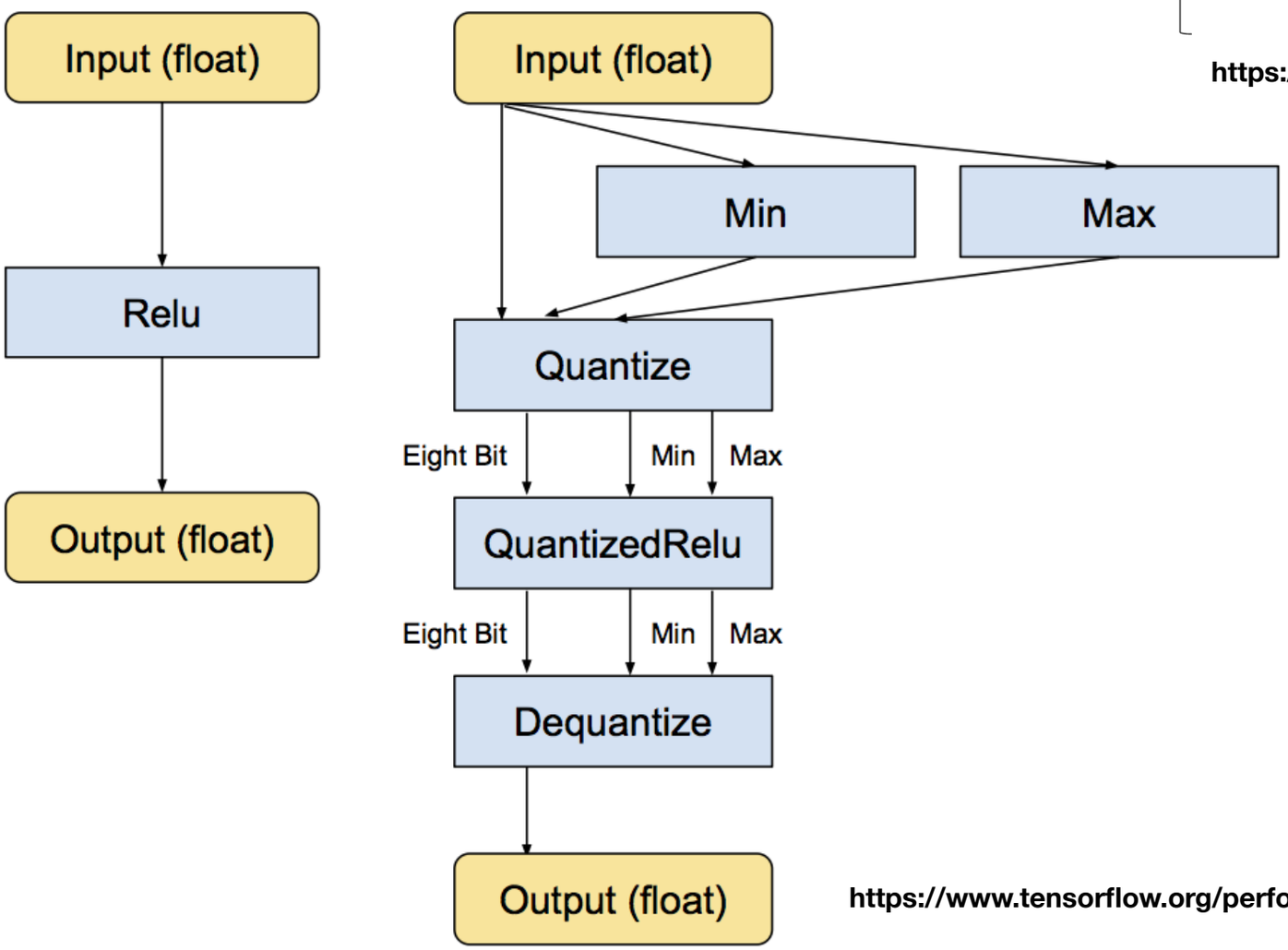
```
$ bazel-bin/tensorflow/examples/label_image/label_image --image=my_image.png
```

For a more detailed look at this code, you can check out the C++ section of the Inception tutorial.

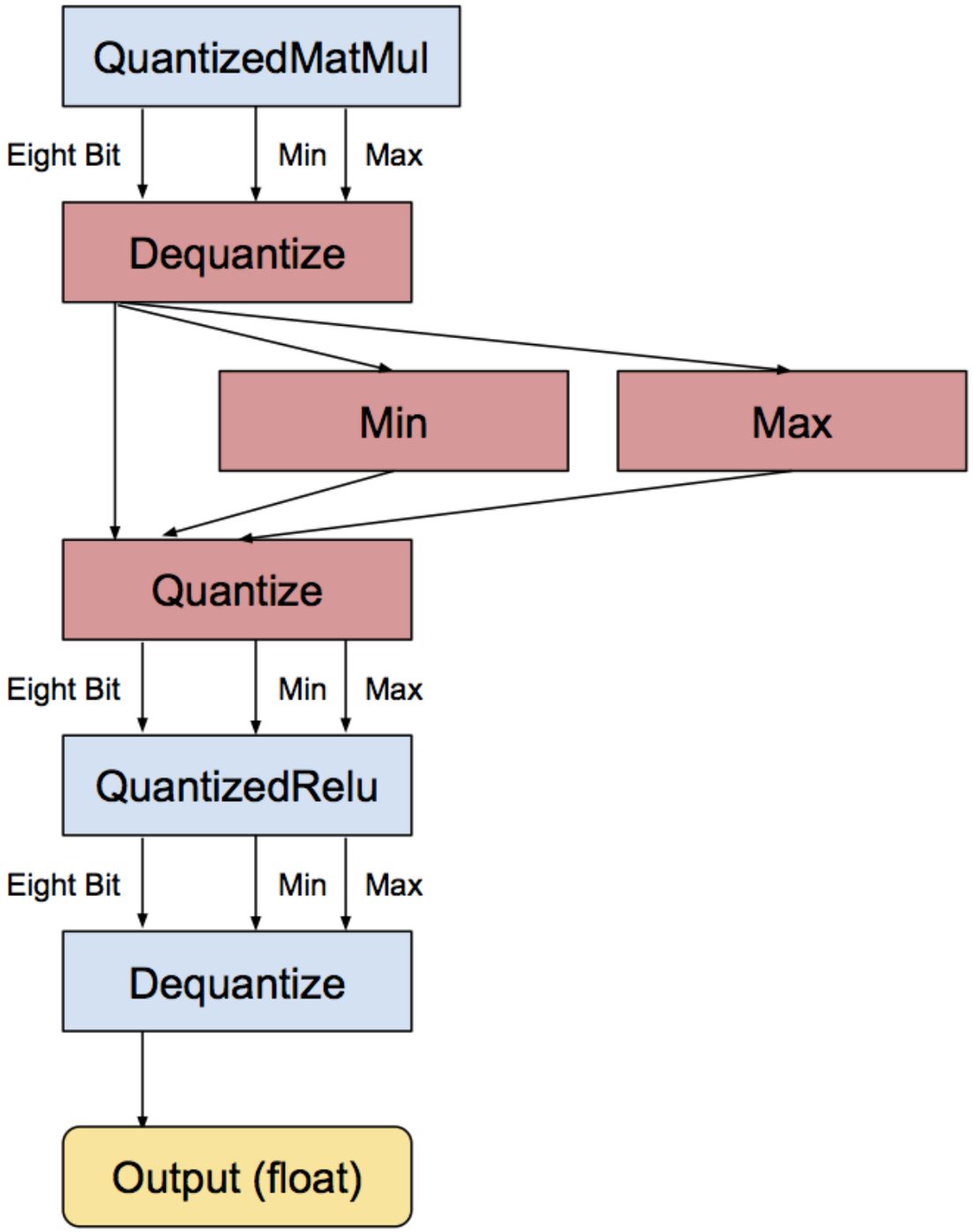
Quantization

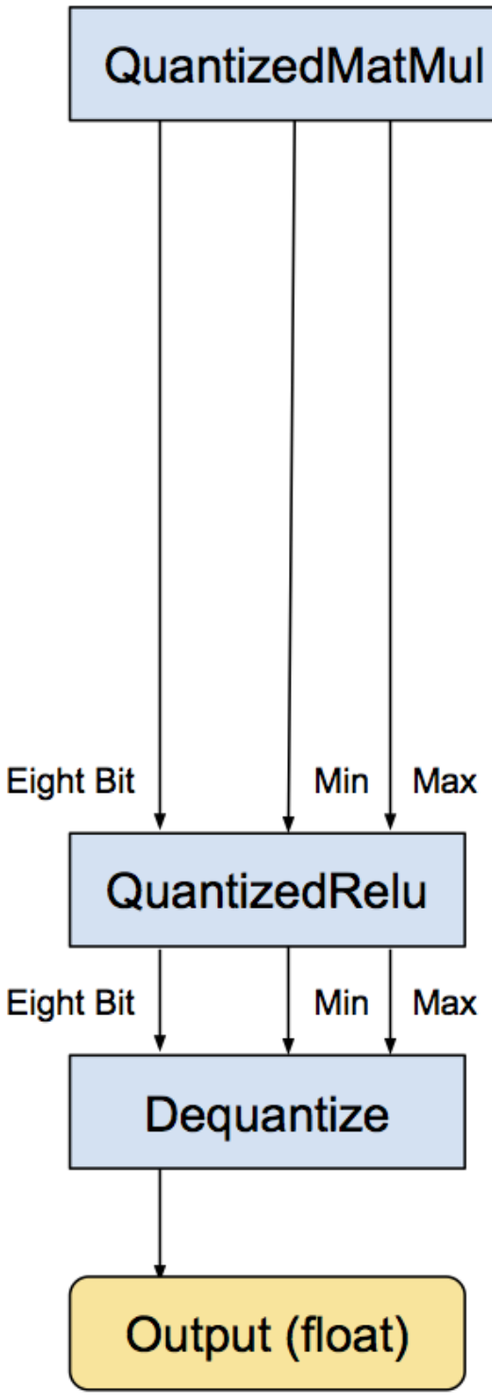


https://cloud.google.com/blog/big-data/2017/05/an-in-depth-look-at-googles-first-tensor-processing-unit-tpu



https://www.tensorflow.org/performance/quantization





https://www.tensorflow.org/performance/quantization

Quantizated nodes

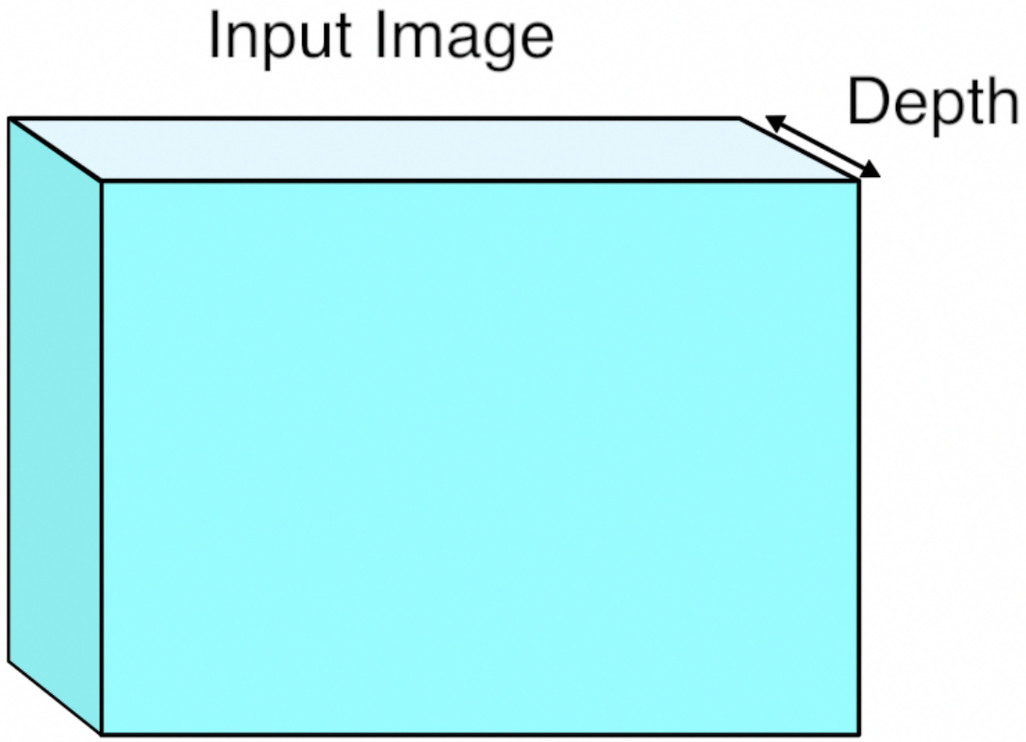
Const	1.004	0.009	0.005	0.000%	0.053%	0.000	1	InceptionV3/Conv2d_1a_3x3/weights_quint8_const
Const	1.011	0.007	0.005	0.000%	0.054%	0.000	1	InceptionV3/Conv2d_1a_3x3/weights_min
Const	1.018	0.007	0.004	0.000%	0.054%	0.000	1	InceptionV3/Conv2d_1a_3x3/weights_max
Const	1.023	0.041	0.038	0.003%	0.057%	0.000	1	InceptionV3/InceptionV3/Conv2d_1a_3x3/convolution_eightbit_reshape_dims
Const	1.064	0.007	0.005	0.000%	0.057%	0.000	1	InceptionV3/InceptionV3/Conv2d_1a_3x3/convolution_eightbit_reduction_dims
Const	1.072	0.008	0.005	0.000%	0.057%	0.000	1	InceptionV3/InceptionV3/Conv2d_1a_3x3/Relu_eightbit_reduction_dims
Const	1.080	0.008	0.005	0.000%	0.058%	0.000	1	InceptionV3/Conv2d_2a_3x3/weights_quint8_const
Const	1.087	0.007	0.004	0.000%	0.058%	0.000	1	InceptionV3/Conv2d_2a_3x3/weights_min
Const	1.093	0.007	0.004	0.000%	0.058%	0.000	1	InceptionV3/Conv2d_2a_3x3/weights_max
Const	1.099	0.008	0.005	0.000%	0.059%	0.000	1	InceptionV3/InceptionV3/Conv2d_2a_3x3/Relu_eightbit_reduction_dims
Const	1.105	0.007	0.005	0.000%	0.059%	0.000	1	InceptionV3/Conv2d_2b_3x3/weights_quint8_const
Const	1.112	0.007	0.004	0.000%	0.059%	0.000	1	InceptionV3/Conv2d_2b_3x3/weights_min
Const	1.117	0.013	0.004	0.000%	0.060%	0.000	1	InceptionV3/Conv2d_2b_3x3/weights_max
Const	1.123	0.008	0.004	0.000%	0.060%	0.000	1	InceptionV3/InceptionV3/Conv2d_2b_3x3/Relu_eightbit_reduction_dims
Const	1.130	0.014	0.005	0.000%	0.060%	0.000	1	InceptionV3/Conv2d_3b_1x1/weights_quint8_const
Const	1.137	0.007	0.004	0.000%	0.061%	0.000	1	InceptionV3/Conv2d_3b_1x1/weights_min
Const	1.142	0.049	0.005	0.000%	0.061%	0.000	1	InceptionV3/Conv2d_3b_1x1/weights_max
Const	1.149	0.010	0.007	0.000%	0.062%	0.000	1	InceptionV3/InceptionV3/Conv2d_3b_1x1/Relu_eightbit_reduction_dims
Const	1.158	0.009	0.006	0.000%	0.062%	0.000	1	InceptionV3/Conv2d_4a_3x3/weights_quint8_const
Const	1.166	0.006	0.004	0.000%	0.062%	0.000	1	InceptionV3/Conv2d_4a_3x3/weights_min
Const	1.172	0.006	0.006	0.000%	0.063%	0.000	1	InceptionV3/Conv2d_4a_3x3/weights_max
Const	1.179	0.007	0.005	0.000%	0.063%	0.000	1	InceptionV3/InceptionV3/Conv2d_4a_3x3/Relu_eightbit_reduction_dims

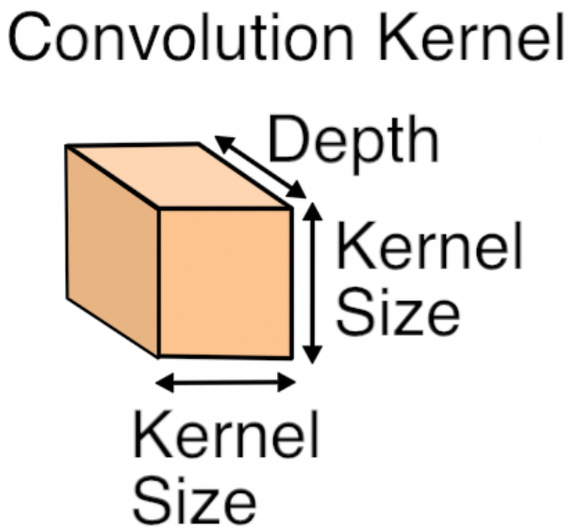
label_image

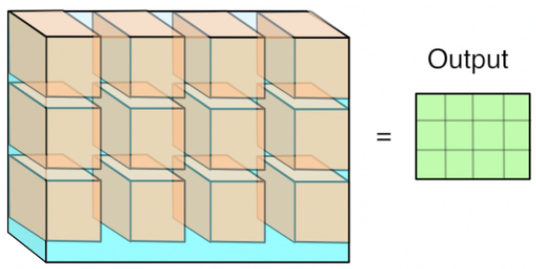
```
flounder:/data/local/tmp $ ./label_image  --graph=inception_v3_2016_08_28_frozen.pb  --
image=grace_hopper.bmp  --labels=imagenet_slim_labels.txt
can't determine number of CPU cores: assuming 4
can't determine number of CPU cores: assuming 4
native : main.cc:250 military uniform (653): 0.834119
native : main.cc:250 mortarboard (668): 0.0196274
native : main.cc:250 academic gown (401): 0.00946237
native : main.cc:250 pickelhaube (716): 0.00757228
native : main.cc:250 bulletproof vest (466): 0.0055856
flounder:/data/local/tmp $ ./label_image  --graph=quantized_graph.pb  --image=grace_hopper.bmp  --
labels=imagenet_slim_labels.txt
can't determine number of CPU cores: assuming 4
can't determine number of CPU cores: assuming 4
native : main.cc:250 military uniform (653): 0.930771
native : main.cc:250 mortarboard (668): 0.00730017
native : main.cc:250 bulletproof vest (466): 0.00365008
native : main.cc:250 pickelhaube (716): 0.00365008
native : main.cc:250 academic gown (401): 0.00365008
```

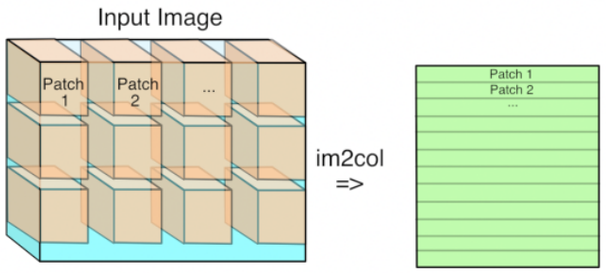
gemmlowp

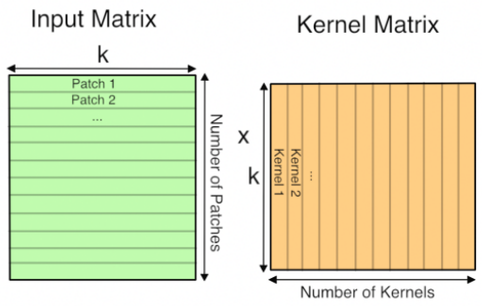
- GEMM (GEneral Matrix Multiplication)
  - The Basic Linear Algebra Subprograms (BLAS) are routines that provide standard building blocks for performing basic vector and matrix operations
    - The Level 1 BLAS (1979) perform scalar, vector and vector-vector operations,
    - the Level 2 BLAS (1988) perform matrix-vector operations, and
    - the Level 3 BLAS (1990) perform matrix-matrix operations: {**S,D,C,Z}GEMM** and others
- Lowp: low-precision
  - less than **single** precision floating point numbers (< 32-bit), well, actually, "low-precision" in gemmlowp means that the input and output matrix entries are integers on at most 8 bits
- Why GEMM
  - Optimized
  - FC, Convolution (im2col, see next page)











https://petewarden.com/2015/04/20/why-gemm-is-at-the-heart-of-deep-learning/

Quantization is tricky

- Yes, we see https://www.tensorflow.org/performance/quantization
  - tensorflow/tools/quantization/
- There are others utilities
  - tensorflow/tools/graph_transforms/
- Inception V3 model,
  - Floating point numbers

```
./benchmark_model --output_layer=InceptionV3/Predictions/Reshape_1 -input_layer_shape=1,299,299,3
```

    - avg: around 840 ms for a 299x299x3 photo
  - Quantized one

```
./benchmark_model --graph=quantized_graph.pb  --output_layer=InceptionV3/Predictions/Reshape_1 --input_layer_shape=1,299,299,3
```

    - If we tried a recent one, oops, > 1.2 seconds

```
8 ============================== Summary by node type ==============================
8                  [Node type]   [count]   [avg ms]    [avg %]    [cdf %]  [mem KB]  [times called]
8                       Conv2D        95    667.344    75.639%    75.639%  35873.957              95
8                          Mul        94    165.642    18.774%    94.413%      0.000              94
8                      AvgPool        10     14.910     1.690%    96.103%   8017.792              10
8                          Add        94     13.279     1.505%    97.608%      0.000              94
8                         Relu        94      7.526     0.853%    98.461%      0.000              94
8                      MaxPool         4      7.178     0.814%    99.275%   2834.560               4
8                     ConcatV2        15      4.399     0.499%    99.774%  10678.528              15
8                        Const       287      1.678     0.190%    99.964%      0.000             287
8                      Softmax         1      0.102     0.012%    99.975%      0.000               1
8                         NoOp         1      0.071     0.008%    99.983%      0.000               1
8                      Reshape         2      0.043     0.005%    99.988%      0.000               2
8                      _Retval         1      0.030     0.003%    99.992%      0.000               1
8                      BiasAdd         1      0.026     0.003%    99.995%      0.000               1
8                      Squeeze         1      0.024     0.003%    99.997%      0.000               1
8                         _Arg         1      0.015     0.002%    99.999%      0.000               1
8                     Identity         1      0.009     0.001%   100.000%      0.000               1
8
8 Timings (microseconds): count=50 first=883076 curr=891488 min=830478 max=1087429 avg=882624 std=38237
```

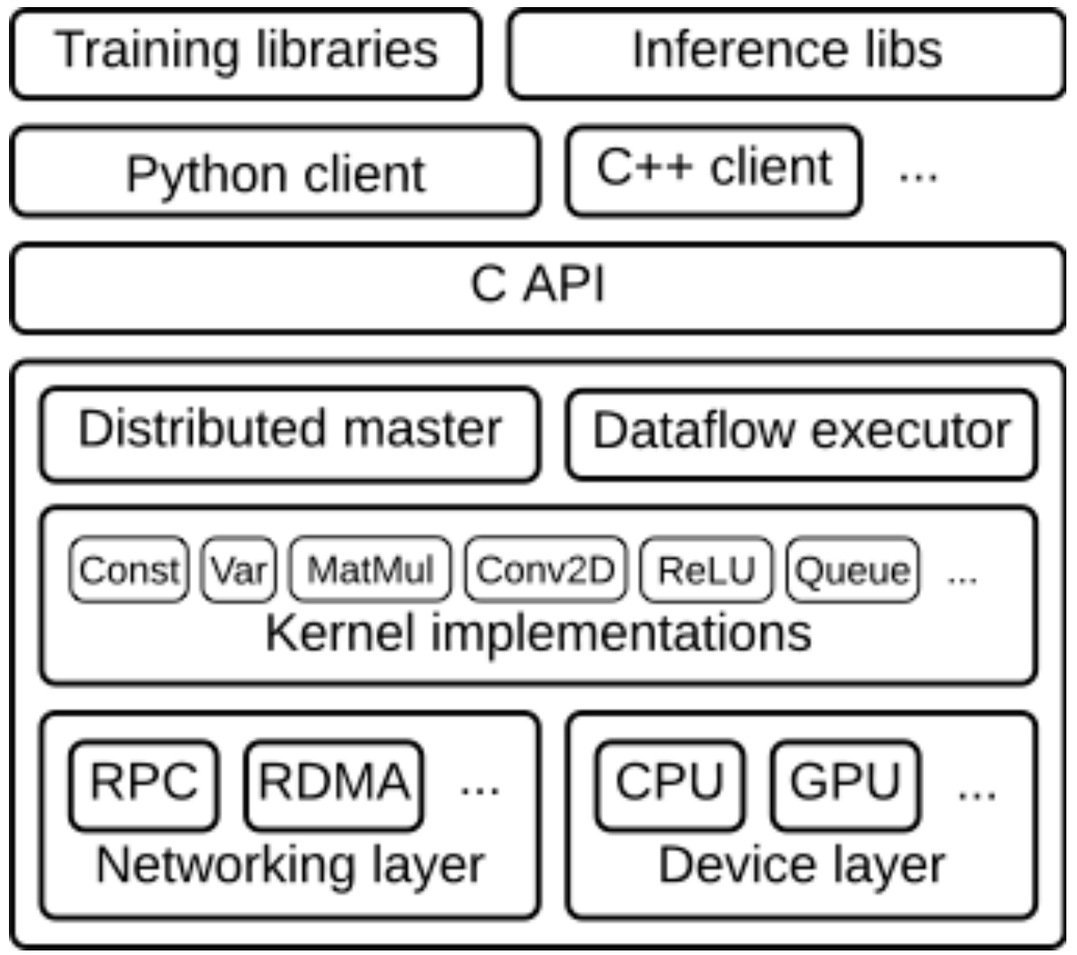
```
68 ============================== Summary by node type ==============================
68                  [Node type]   [count]   [avg ms]    [avg %]    [cdf %]  [mem KB]  [times called]
68           QuantizedConv2D        95    790.267    59.655%    59.655%  35874.715              95
68                          Mul        94    177.193    13.376%    73.031%      0.000              94
68                   QuantizeV2       136    121.860     9.199%    82.230%  18162.908             136
68                   Dequantize       143     58.238     4.396%    86.626%  45774.059             143
68                   Requantize        96     44.876     3.388%    90.014%   8970.258              96
68                          Min       136     26.907     2.031%    92.045%      0.544             136
68                          Add        94     24.626     1.859%    93.904%      0.000              94
68                          Max       136     23.206     1.752%    95.656%      0.544             136
68           QuantizedAvgPool        10     19.729     1.489%    97.145%   2004.528              10
68       RequantizationRange         96     10.133     0.765%    97.910%      0.768              96
68             QuantizedRelu         94      9.231     0.697%    98.607%   8968.240              94
68          QuantizedMaxPool          4      8.622     0.651%    99.258%    708.672               4
68                   ConcatV2        15      5.450     0.411%    99.669%  10678.528              15
68                      Const       630      2.682     0.202%    99.871%      0.000             630
68                    Reshape       108      1.251     0.094%    99.966%      0.000             108
68                       NoOp         1      0.139     0.010%    99.976%      0.000               1
68                    Softmax         1      0.135     0.010%    99.987%      0.000               1
68          QuantizedBiasAdd          1      0.063     0.005%    99.991%      4.012               1
68                    _Retval         1      0.052     0.004%    99.995%      0.000               1
68          QuantizedReshape          2      0.025     0.002%    99.997%      0.016               2
68                    Squeeze         1      0.022     0.002%    99.999%      0.000               1
68                       _Arg         1      0.017     0.001%   100.000%      0.000               1
68
68 Timings (microseconds): count=50 first=1332713 curr=1402667 min=1280080 max=1402667 avg=1.32564e+06 std=21647
```

Current status of TF

- Well, public status
    - Google does have internal branches, during review process of BMP decoder, I ran into one
- CPU ARMv7 and ARMv8
    - Q hexagon DSP, https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/hvx
- Eigen and gemmlowp
- Basic XLA works
- Not all operations are supported
    - the ‘name = “mobile_srcs”’ in tensorflow/core/BUILD
        - “//tensorflow/core/kernels:android_core_ops", “//tensorflow/core/kernels:android_extended_ops" in tensorflow/core/kernel/BUILD
- C++ and Java API (the TensorFlow site lists Python, C++, Java, and GO now)
    - I am far away from Java, don't know how good the API is
    - “A word of caution: the APIs in languages other than Python are not yet covered by the API stability promises."
- You may find something like RSTensorFlow and tf-coriander, but AFAICT they are far away from complete

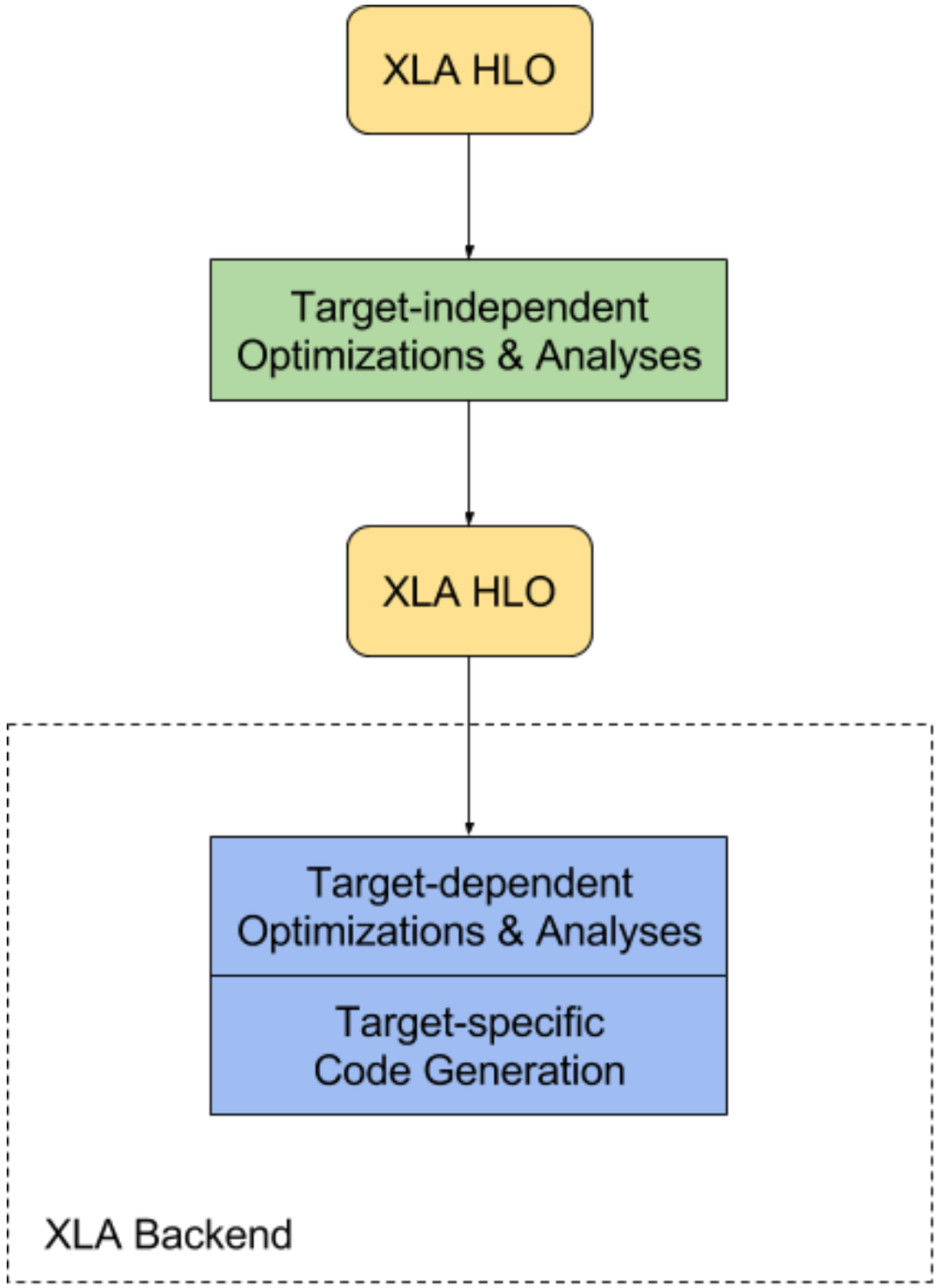
Arch including distributed training

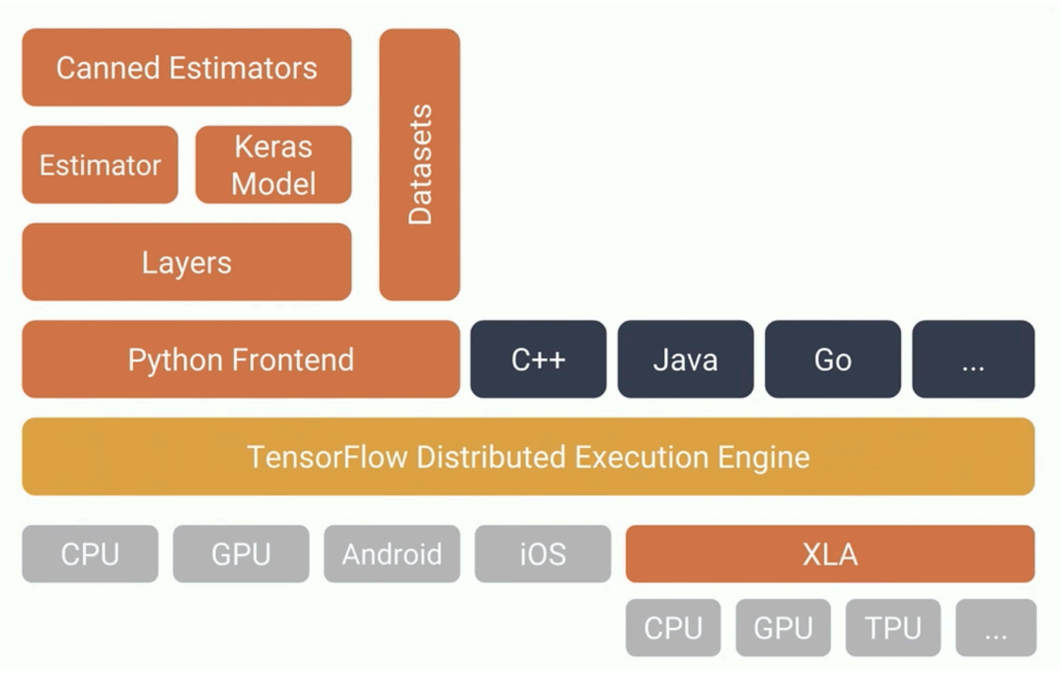
- The architecture figure of TensorFlow show important components, including distributed stuff

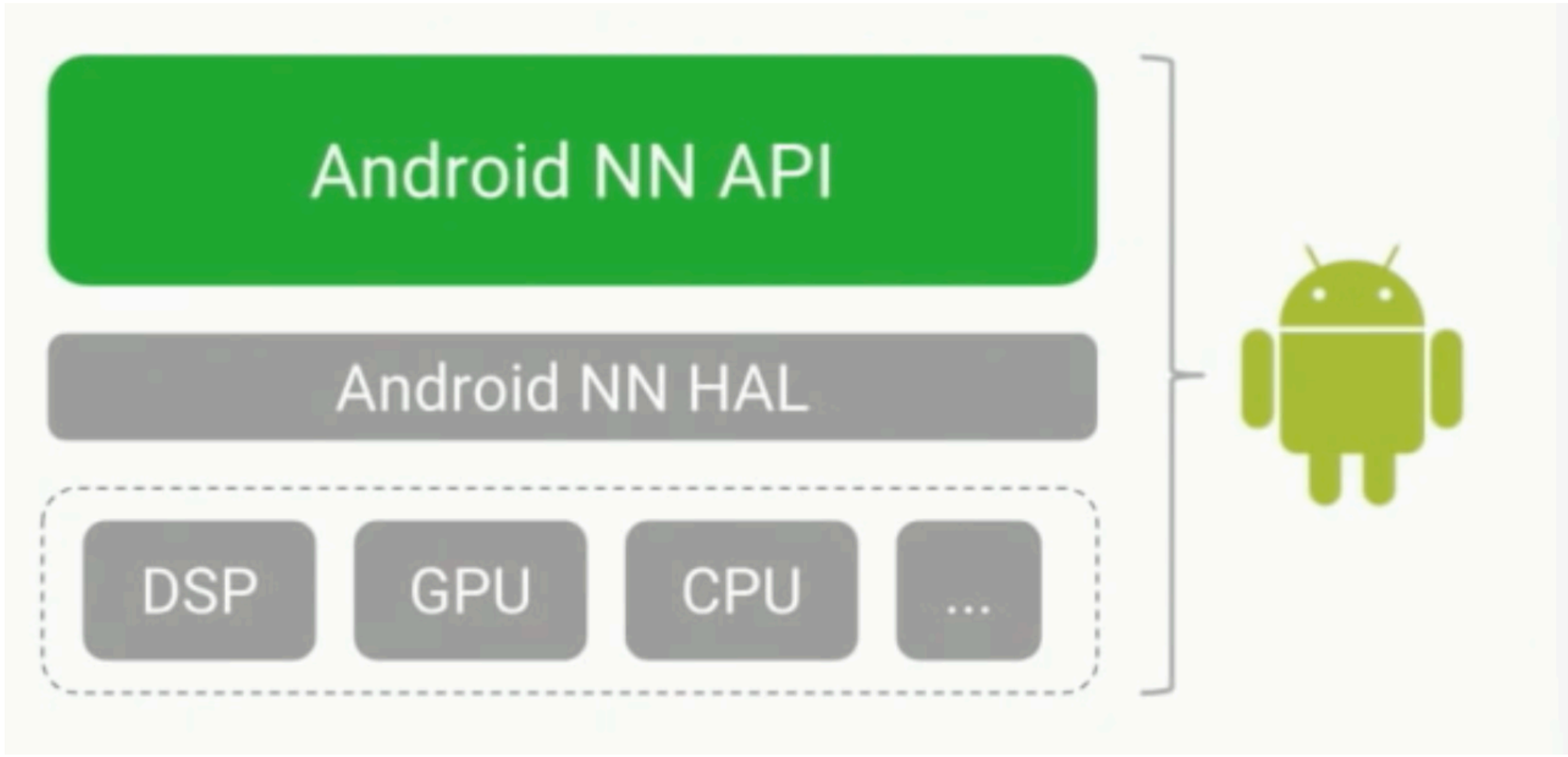


https://www.tensorflow.org/extend/architecture

TensorFlow Architecture



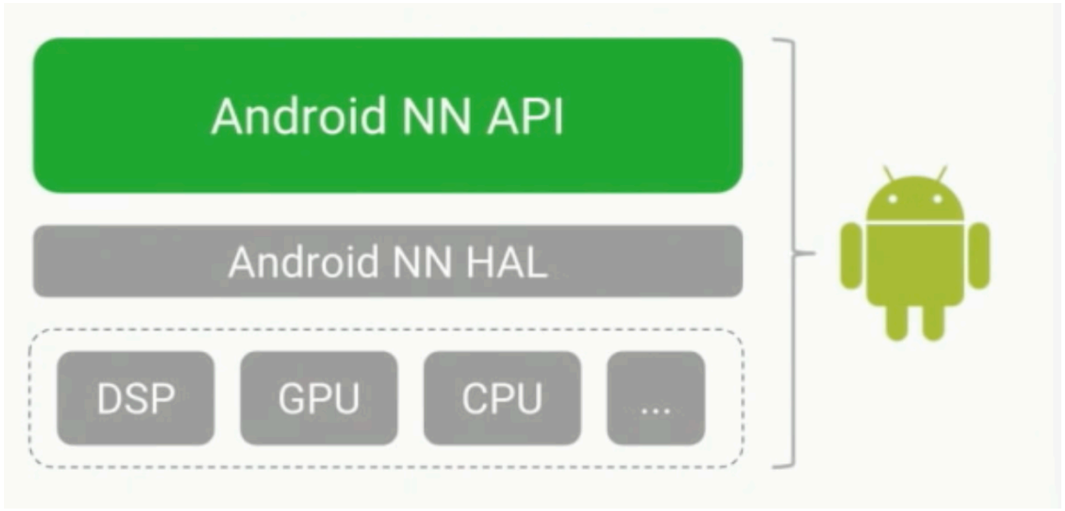




AndroidNN is coming to town

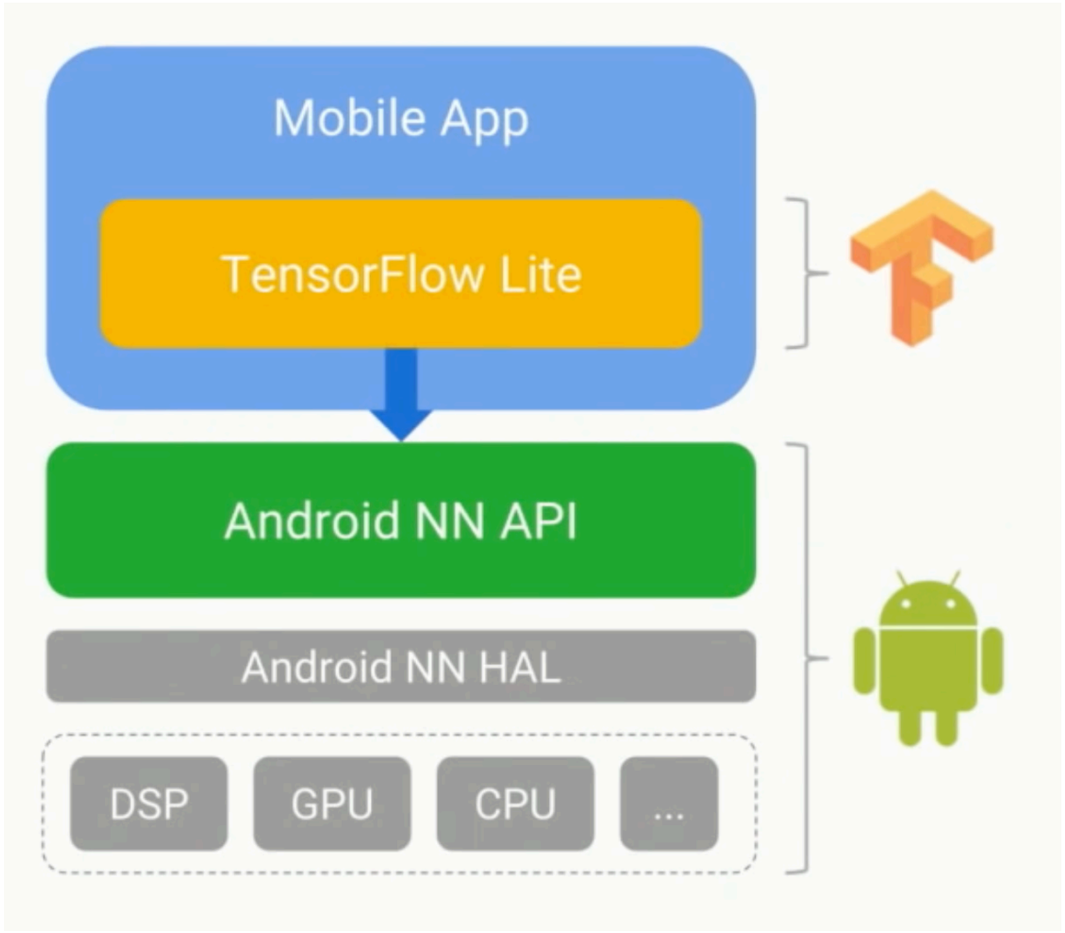
Android Neural Network API

- New API for Neural Network
- Being added to the Android framework
- Wraps hardware accelerators (GPU, DSP, ISP, etc.)



from Google I/O 2017 video

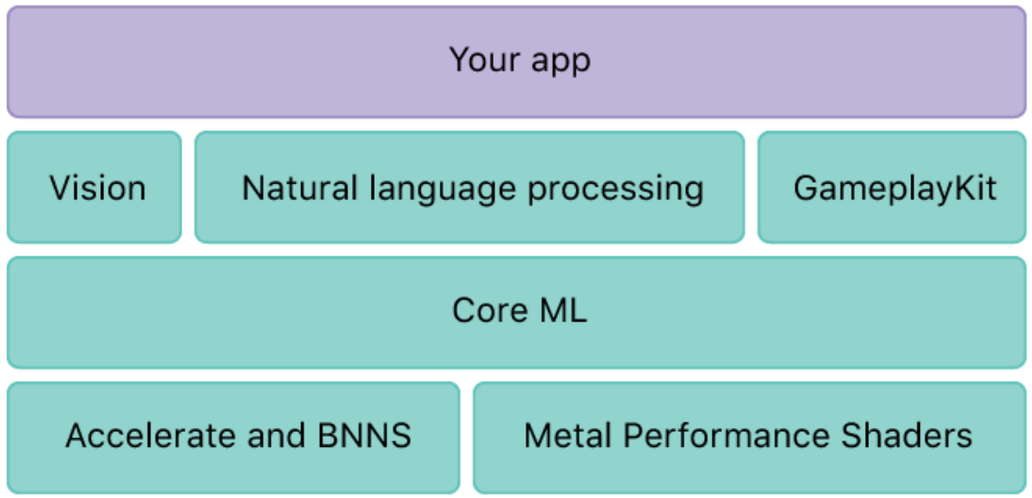
- **New TensorFlow runtime**
- Optimized for **mobile** and **embedded** apps
- Runs TensorFlow models on device
- Leverage Android NN API
- Soon to be open sourced



from Google I/O 2017 video

Comparing with CoreML stack

- No GPU/GPGPU support yet. Hopefully, Android NN will help.
- Albeit Google is so good at ML/DL and various applications, we don’t see good application framework(s) on Android yet.



Simple CoreML Exercise

- Simple app to use InceptionV3 to classify image from Photo roll or camera
    - in Objective-C
    - in Swift
- Work continuously on camera
    - in Objective-C
    - in Swift

Depthwise Separable Convolution

- CNNs with depthwise separable convolution such as Mobilenet [1] changed almost everything

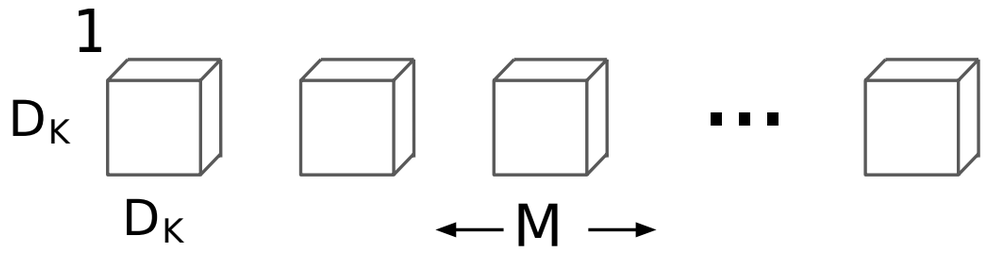
- Depthwise separable convolution “factorize” a standard convolution into a depthwise convolution and a 1 × 1 convolution called a pointwise convolution. Thus it greatly reduces computation complexity.

- Depthwise separable convolution is not that that new [2], but pure depthwise separable convolution-based networks such as Xception and MobileNet demonstrated its power

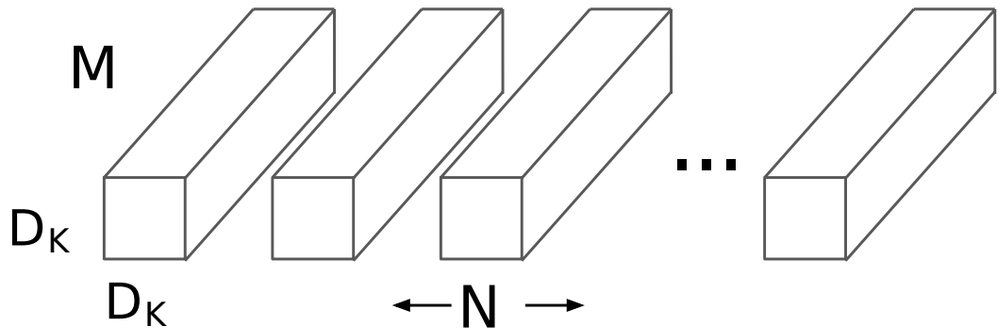
[1] https://arxiv.org/abs/1704.04861

[2] L. Sifre. “Rigid-motion scattering for image classification”, PhD thesis, 2014

Depthwise Separable Convolution



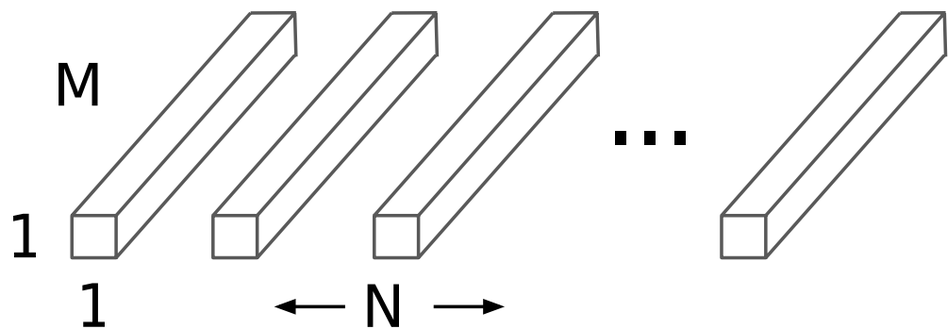
depthwise convolution filters



standard convolution filters

$$\frac{D_K \cdot D_K \cdot M \cdot D_F \cdot D_F + M \cdot N \cdot D_F \cdot D_F}{D_K \cdot D_K \cdot M \cdot N \cdot D_F \cdot D_F}$$

$$= \frac{1}{N} + \frac{1}{D_K^2}$$



https://arxiv.org/abs/1704.04861 **1×1 Convolutional Filters (Pointwise Convolution)**

MobileNet

- D_K: kernel size
- D_F: input size
- M: input channel size
- N: output channel size

https://arxiv.org/abs/1704.04861

Table 1. MobileNet Body Architecture

Type / Stride	Filter Shape	Input Size
Conv / s2	$3 \times 3 \times 3 \times 32$	$224 \times 224 \times 3$
Conv dw / s1	$3 \times 3 \times 32$ dw	$112 \times 112 \times 32$
Conv / s1	$1 \times 1 \times 32 \times 64$	$112 \times 112 \times 32$
Conv dw / s2	$3 \times 3 \times 64$ dw	$112 \times 112 \times 64$
Conv / s1	$1 \times 1 \times 64 \times 128$	$56 \times 56 \times 64$
Conv dw / s1	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 128$	$56 \times 56 \times 128$
Conv dw / s2	$3 \times 3 \times 128$ dw	$56 \times 56 \times 128$
Conv / s1	$1 \times 1 \times 128 \times 256$	$28 \times 28 \times 128$
Conv dw / s1	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 256$	$28 \times 28 \times 256$
Conv dw / s2	$3 \times 3 \times 256$ dw	$28 \times 28 \times 256$
Conv / s1	$1 \times 1 \times 256 \times 512$	$14 \times 14 \times 256$
$5\times$ Conv dw / s1	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
$5\times$ Conv / s1	$1 \times 1 \times 512 \times 512$	$14 \times 14 \times 512$
Conv dw / s2	$3 \times 3 \times 512$ dw	$14 \times 14 \times 512$
Conv / s1	$1 \times 1 \times 512 \times 1024$	$7 \times 7 \times 512$
Conv dw / s2	$3 \times 3 \times 1024$ dw	$7 \times 7 \times 1024$
Conv / s1	$1 \times 1 \times 1024 \times 1024$	$7 \times 7 \times 1024$
Avg Pool / s1	Pool 7×7	$7 \times 7 \times 1024$
FC / s1	1024×1000	$1 \times 1 \times 1024$
Softmax / s1	Classifier	$1 \times 1 \times 1000$

MobileNet on Nexus 9

- “largest” Mobilenet model http://download.tensorflow.org/models/mobilenet_v1_1.0_224_frozen.tgz

  - benchmark_model: ./benchmark_model --graph=frozen_graph.pb —output_layer=MobilenetV1/Predictions/Reshape_1

  - around 120 ms

- Smallest one

  - mobilenet_v1_0.25_128: ~25 ms

```
flounder:/data/local/tmp $ ./label_image  --graph=mobilenet_10_224.pb   --image=grace_hopper.bmp  --
labels=imagenet_slim_labels.txt  --output_layer=MobilenetV1/Predictions/Reshape_1 --input_width=224 --
input_height=224
can't determine number of CPU cores: assuming 4
can't determine number of CPU cores: assuming 4
native : main.cc:250 military uniform (653): 0.871238
native : main.cc:250 bow tie (458): 0.0575709
native : main.cc:250 ping-pong ball (723): 0.0113924
native : main.cc:250 suit (835): 0.0110482
native : main.cc:250 bearskin (440): 0.00586033
```

```
flounder:/data/local/tmp $ ./label_image  --graph=mobilenet_025_128.pb   --image=grace_hopper.bmp  --
labels=imagenet_slim_labels.txt  --output_layer=MobilenetV1/Predictions/Reshape_1 --input_width=128 --
input_height=128
can't determine number of CPU cores: assuming 4
can't determine number of CPU cores: assuming 4
native : main.cc:250 suit (835): 0.310988
native : main.cc:250 bow tie (458): 0.197784
native : main.cc:250 military uniform (653): 0.121169
native : main.cc:250 academic gown (401): 0.0309299
native : main.cc:250 mortarboard (668): 0.0242411
```

Recap

- TensorFlow may not be great on Android yet
- New techniques and NN models are changing status quo
- Android NN, XLA, MobileNet
- big.LITTLE and other system software optimization may still be needed

Questions?

Backup

MobileNet on iPhone

- Find a Caffe model, e.g., the one

- Or, use a converted one