TensorFlow

Running Deep Learning in less than 100KB on Microcontrollers

TensorFlow

Pete Warden

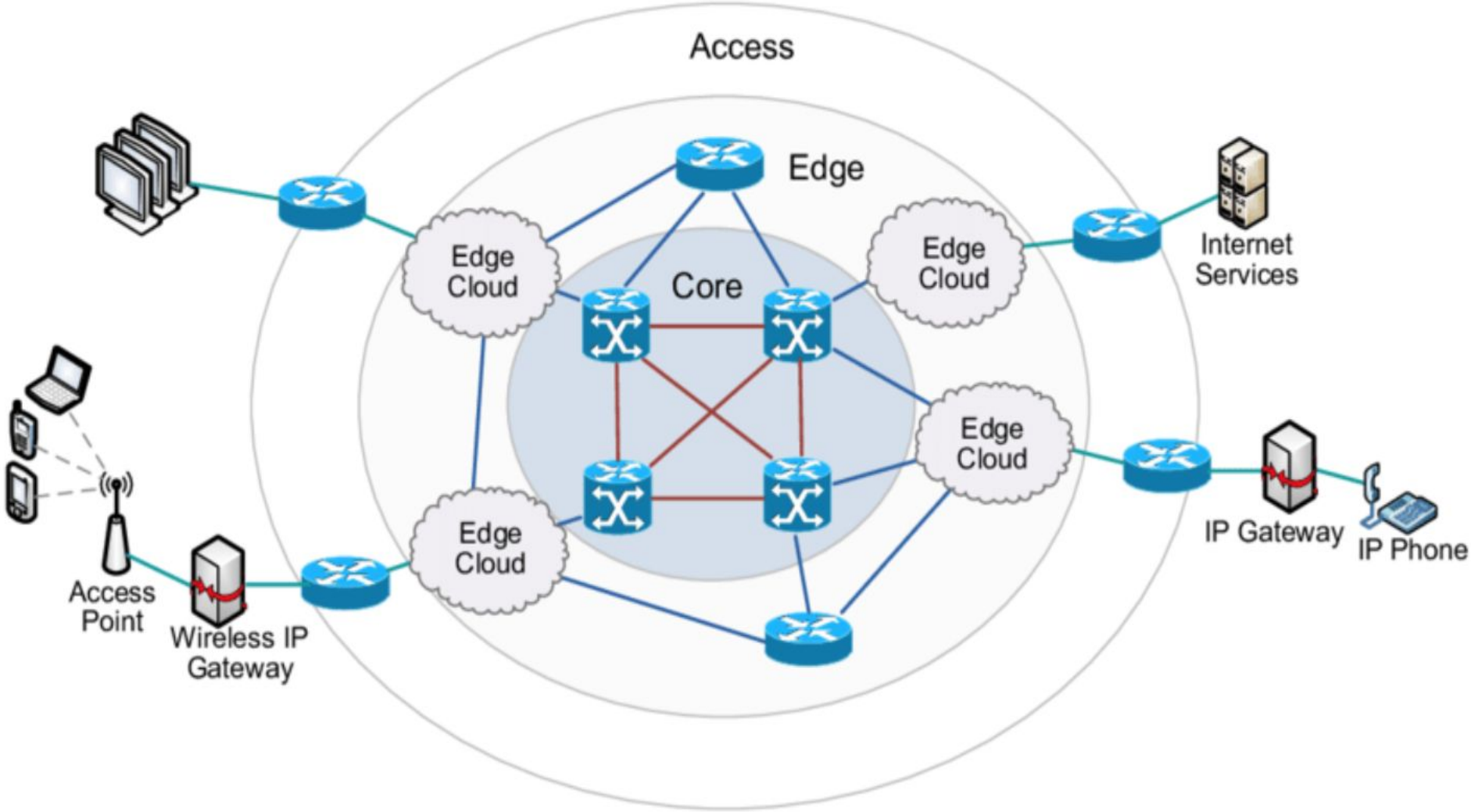
Engineer, TensorFlow
petewarden@google.com
@petewarden

TensorFlow

Why am I here?

150 Billion Devices!

Growing faster than internet users or smartphones


Access
Edge
Core
Edge Cloud
Edge Cloud
Edge Cloud
Edge Cloud
Internet Services
IP Gateway
IP Phone
Access Point
Wireless IP Gateway

TensorFlow

Why ML?

Energy!

Many devices rely on battery or energy harvesting

Transmitting data takes a lot of power, and can't improve fast enough

Capturing and processing data locally very cheap

Energy!

Most captured data is currently being wasted

ML lets us turn it into something actionable

TensorFlow

Demo

How is this done?

What are the challenges?

Less than 100KB of RAM and storage

Less than 10 million arithmetic ops per second

Can’t rely on floating point hardware

No operating system

Model Design

We needed a 20KB model

Happily common in speech world

Learned a lot about quantization

Actually just a tiny image CNN on spectrograms

Model Design

Uses fewer than 400,000 arithmetic operations

https://www.tensorflow.org/tutorials/sequences/audio_recognition

Software Design

TensorFlow Lite still > 100KB binary size

Depends on Posix and standard C/C++ libraries

Uses dynamic memory allocation

Software Design

But a lot we don’t want to lose from TensorFlow Lite:

- Existing op implementations
- Well-documented APIs and file format
- Conversion tooling

Software Design

Modularized existing code

Separated out API definitions from implementations

Used reference code

Added minimal new runtime layer for MCUs

Software Design

Focused on getting one end to end example working, rather than going broad porting whole framework at once

TensorFlow

What does this mean in practice?

```
int32 acc = 0;
for (int filter_y = 0; filter_y < filter_height; ++filter_y) {
  for (int filter_x = 0; filter_x < filter_width; ++filter_x) {
    const int in_x =
        in_x_origin + dilation_width_factor * filter_x;
    const int in_y =
        in_y_origin + dilation_height_factor * filter_y;
    // If the location is outside the bounds of the input image,
    // use zero as a default value.
    if ((in_x >= 0) && (in_x < input_width) && (in_y >= 0) &&
        (in_y < input_height)) {
      int32 input_val =
          input_data[Offset(input_shape, b, in_y, in_x, ic)];
      int32 filter_val = filter_data[Offset(
          filter_shape, 0, filter_y, filter_x, oc)];
      acc += (filter_val + filter_offset) *
             (input_val + input_offset);
    }
  }
}
```

Reference code is important

Most ML operations can be implemented simply

Most frameworks only ship with optimized versions

Understandable, but makes it very hard to extend or optimize for other platforms

What's the takeaway?

There is no killer app

Voice interfaces are the closest

Vision, accelerometer, audio sensors offer a lot

Need to connect with the right problems

Think about your domain

What could you do if your model ran on a 50 cent chip that could be peeled and stuck anywhere, and run forever?

Get it. Try it.

Code: github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/experimental/micro

Docs: tensorflow.org/lite/guide/microcontroller

Example: g.co/codelabs/sparkfunTF

TensorFlow

Pete Warden

Engineer, TensorFlow
petewarden@google.com
@petewarden