Deep learning
on microcontrollers

Jan Jongboom
IETF 101, London
22 March 2018

https://twitter.com/billyd/status/701804370560729089
Machine learning
Why machine learning on the edge?

Sensor fusion

http://www.gierad.com/projects/supersensor/
Why machine learning on the edge?

Federated learning

https://research.googleblog.com/2017/04/federated-learning-collaborative.html
Why machine learning on the edge?

LPWANs
Why machine learning on the edge?

Offline self-contained systems

Edge vs. Cloud

![Bar chart comparing energy and time per inference between Edge and Cloud, Local CB. The chart shows that Edge has lower energy per inference and higher time per inference compared to Cloud, Local CB.](image-url)
Microcontrollers

Small (1cm²)
Cheap (~1$)
Efficient (standby: 0.3 μA)

Downsides

Slow (max. 100 MHz)
Limited memory (max. 256K RAM)
uTensor

Machine learning for microcontrollers

Runs in <256K RAM

TensorFlow compatible

Built on top of Mbed OS 5
  (file systems, drivers, 150 boards compatible)

Open source, Apache 2.0 license
uTensor Team

Neil Tan
Arm

Michael Bartling
Arm

Dboy Liao
Piniko

Kazami Hsieh
Academia Sinica
How?

MNIST data set

Training set: 60,000 images

Every drawing is downsampled to 28x28 pixels

Supervised learning through backpropagation
Multi-layer perceptron (MLP) classification

Matrix multiplication (weight), then activation function

Potential outputs

Input Layer 784

Hidden Layer 1 128 (relu)

Hidden Layer 2 64 (relu)

Output Layer 10 (softmax)

Loss Layer (cross-entropy)

Output

Neuron

28x28 = 784

© 2018 Arm Limited
How?
Quantization

8-bit integers instead of 32-bit floats

Only during classification

79.9% accuracy vs. 80.3% accuracy (CIFAR-10)

TensorFlow requires floating-point de-quantization between layers

https://petewarden.com/2016/05/03/how-to-quantize-neural-networks-with-tensorflow/
Memory usage

Matrix multiplication in first hidden layer dominates RAM usage:

Input elements: 784
Number of neurons (1st layer): 128
Number of weight (input to 1st layer): 128 * 784
Resulting values (Pre-activation function): 128
Data type: 8-bit integer (1 byte)

1 byte * (784 + (128 * 784) + 128) = 98.891 kB
Other tricks

Paging of memory for larger models (sacrifices speed)

Graph in ROM (requires pre-processing) (MNIST: 26K)

Take advantage in sparsity of data, sacrifice accuracy (*TBD*)
Operators

Add, Subtract
Min, Max, ArgMax
ReLU, Matrix multiplication, Reshape, Quantization
Convolution (WIP)
Pooling (WIP)
Tensors

RAM tensor
Flash tensor
Sparse tensor
Networked tensor

Tensors can be paged to fit larger networks
Workflow

Matrix multiplication data from trained model

Tensor graph
Developing using the simulator

Arm Mbed OS simulator

```
/*
 * This is a demo application for uTensor - an AI interference library for
 * deep learning on small microcontrollers.
 * It’s trained to recognize handwritten digits via the MNIST data set.
 * See https://github.com/utensor/utensor
 */

#include "mbed.h"
#include "tensor.hpp"
#include "deep_mnist_mlp.hpp"
#include "emscripten.h"
#include "C12832.h"

C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);

EventQueue queue;
InterruptIn btn(BUTTON1);

void run_mlp() {
  EM_ASM(
    // this writes the content of the canvas (in the simulator) to /fs/tmp.idx
    window.dumpCanvasToTmpFile();
  );
  // invoke the MLP algorithm against the temp file (just saved from canvas)
  int prediction = runMLP("/fs/tmp.idx");
  lcd.cls();
  lcd.locate(3, 13);
  lcd.printf("Predicted: %d", prediction);
}
```
CMSIS-NN

New neural network kernel functions
Leverages the DSP/SIMD functions in silicon
See speedup of 4-5x
Hardware acceleration for convolution, pooling, etc.
uTensor will be built on top of CMSIS-NN
Recap

1. Buy a development board (http://os.mbed.com/platforms)
2. Clone uTensor (https://github.com/uTensor/uTensor)
3. ???
4. PROFIT!!!
Thank you!

https://labs.mbed.com

Jan Jongboom, Arm