

A photograph of a large audience seated in rows, all wearing VR headsets, facing a stage. The scene is dimly lit, with blue and purple lights illuminating the audience's faces and the stage area.

# Deep learning on microcontrollers

Jan Jongboom

IETF 101, London

22 March 2018

# Data Science 2017 (DSA 2017) - Summer School & Workshop

17<sup>th</sup> - 21<sup>st</sup> July, 2017 at NM-AIST, Arusha



PULSE  
LAB KAMPALA



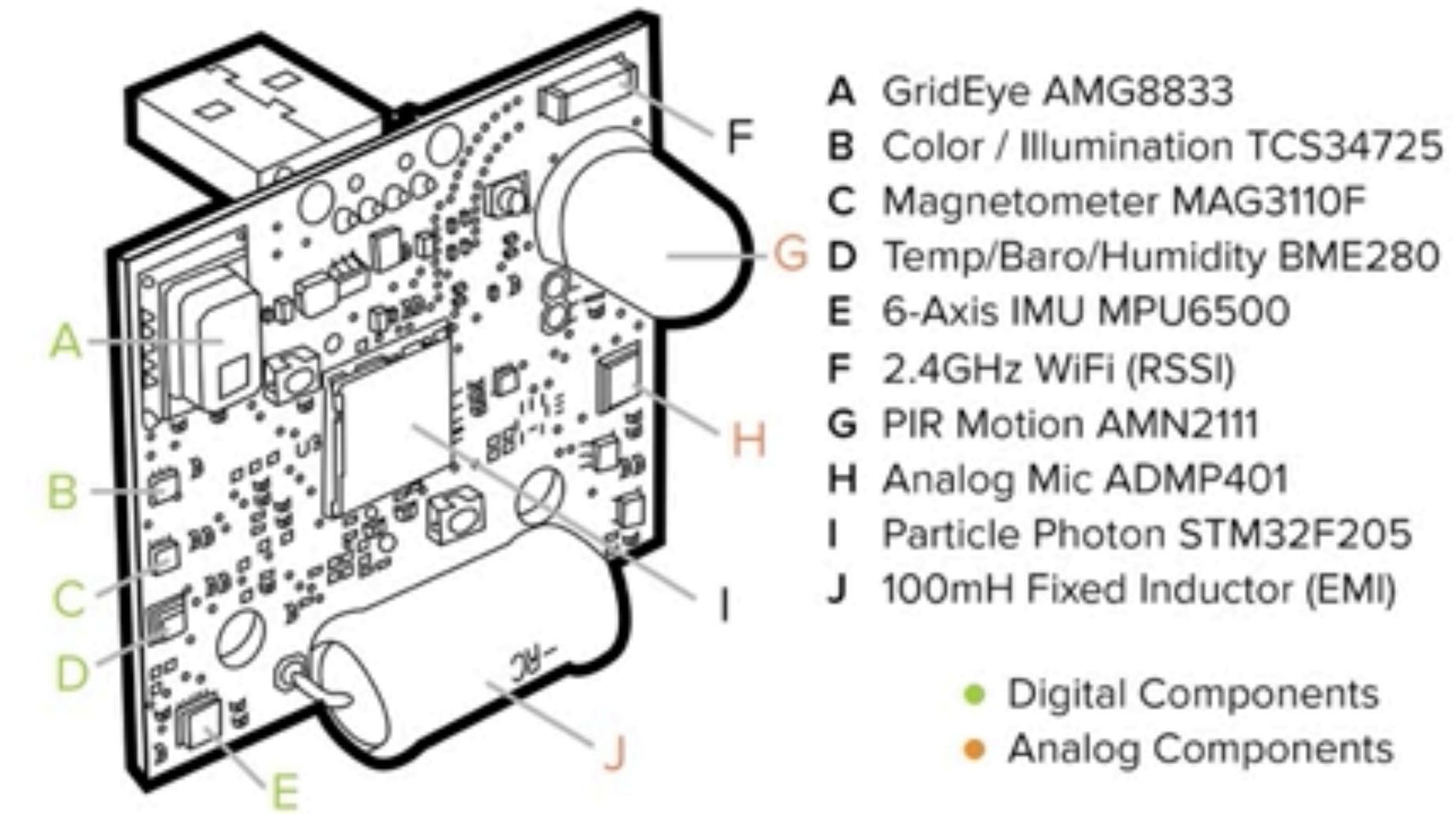
ARMmbed



# Machine learning



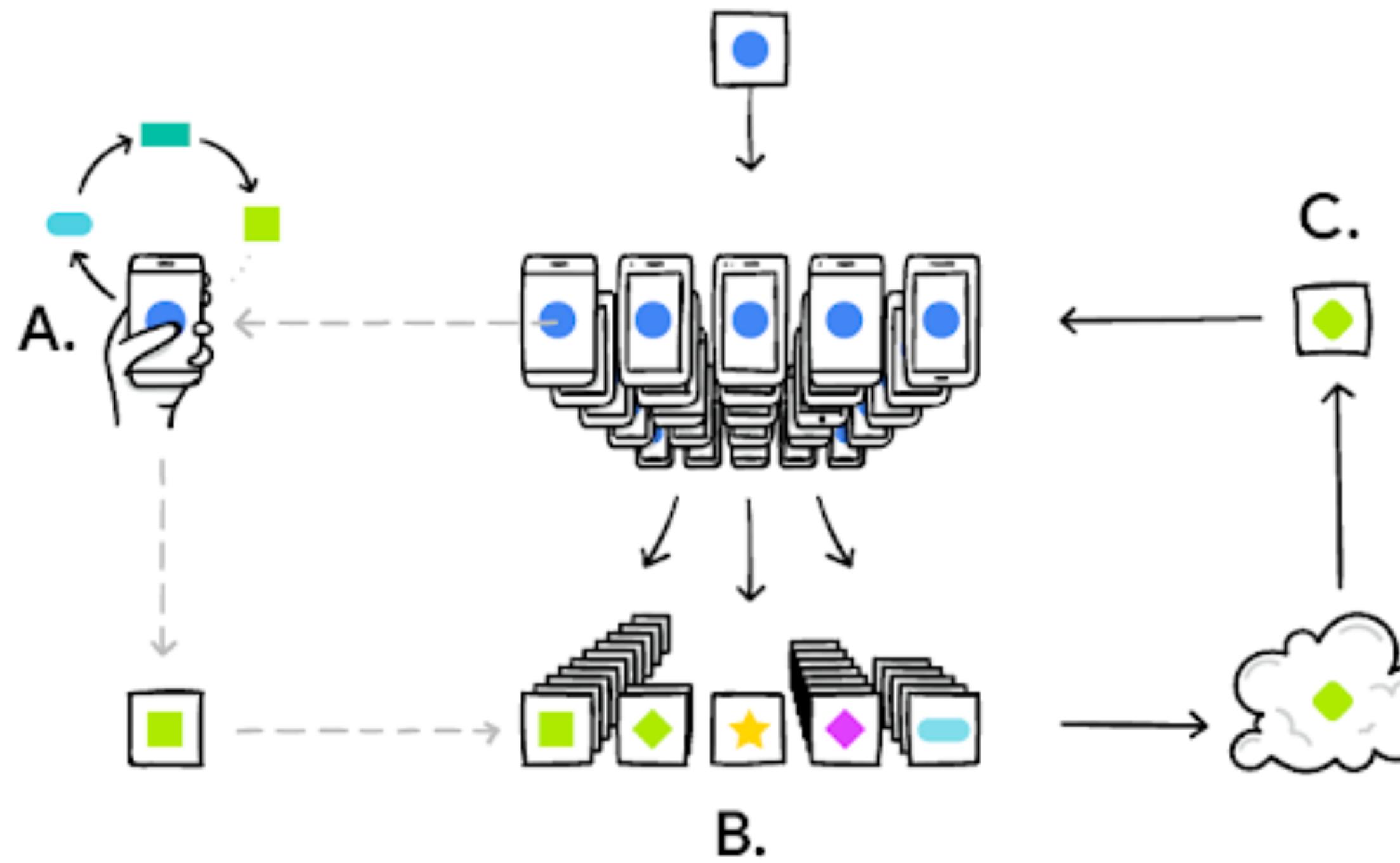
# Why machine learning on the edge?



## 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

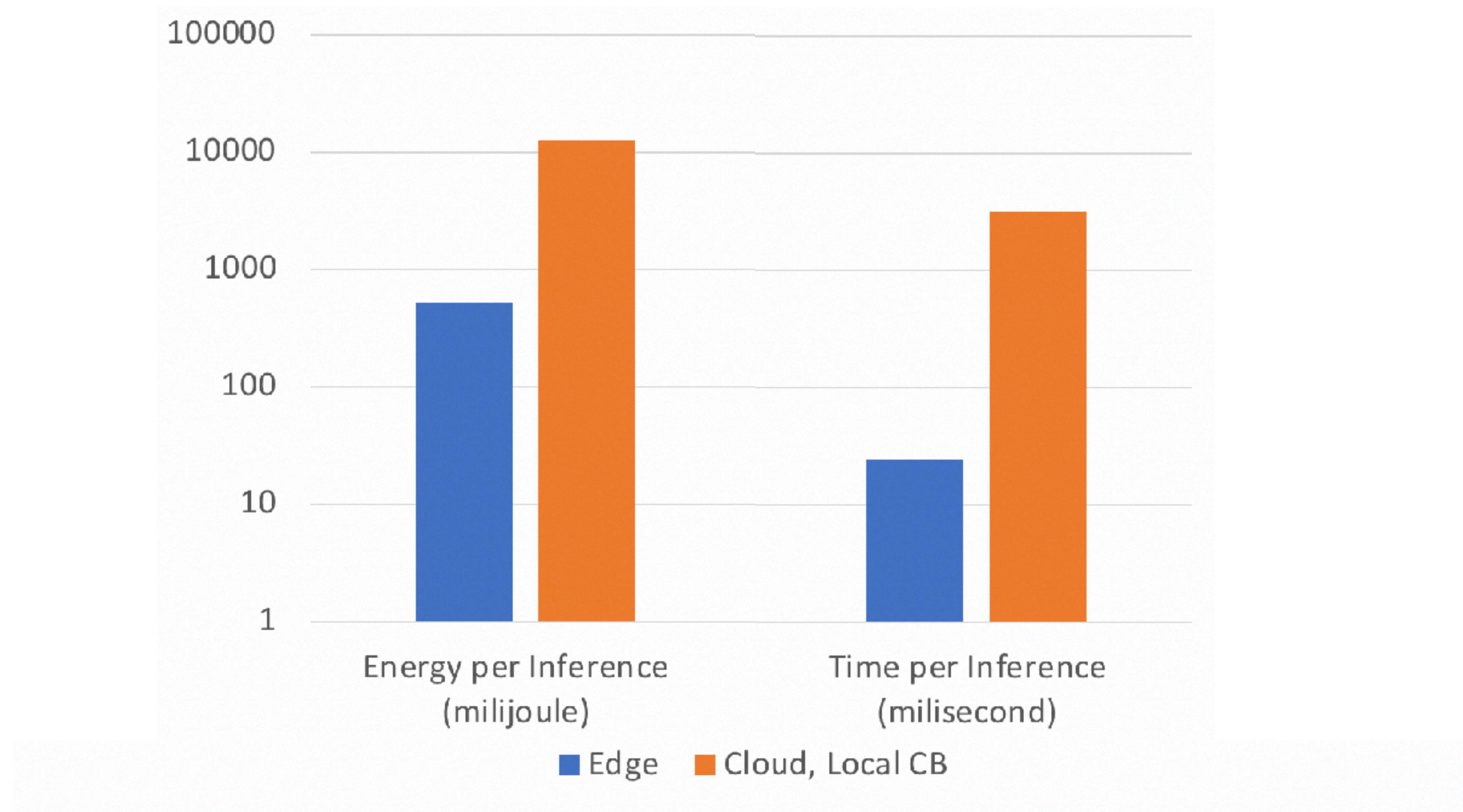
# Why machine learning on the edge?



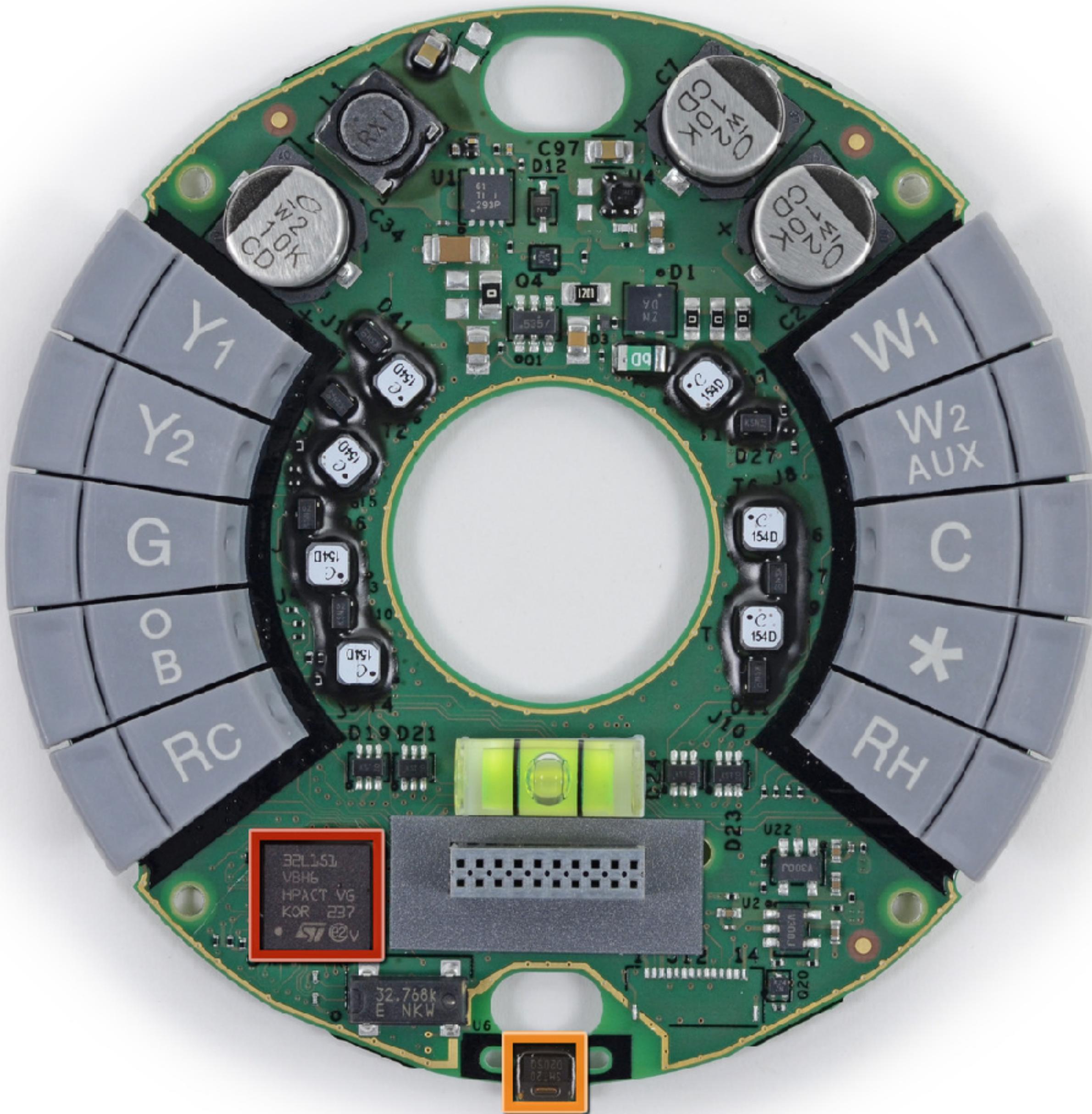
Offline self-contained systems

<https://os.mbed.com/blog/entry/streaming-data-cows-dsa2017/>

# Edge vs. Cloud



# Microcontrollers



Small (1cm<sup>2</sup>)

Cheap (~1\$)

Efficient (standby: 0.3 µA)

## Downsides

Slow (max. 100 MHz)

Limited memory (max. 256K RAM)

# uTensor

Machine learning for microcontrollers

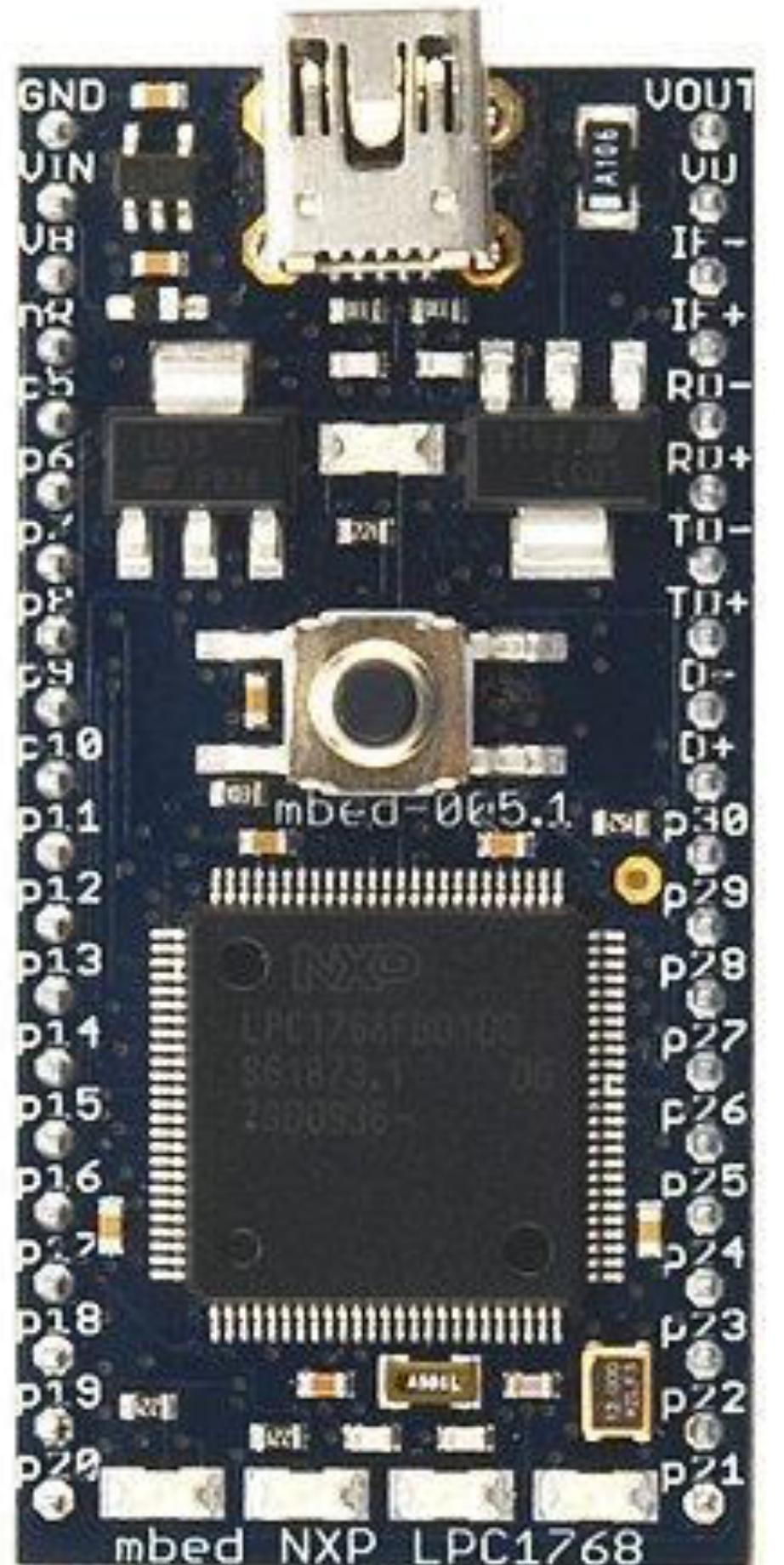
Runs in <256K RAM

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



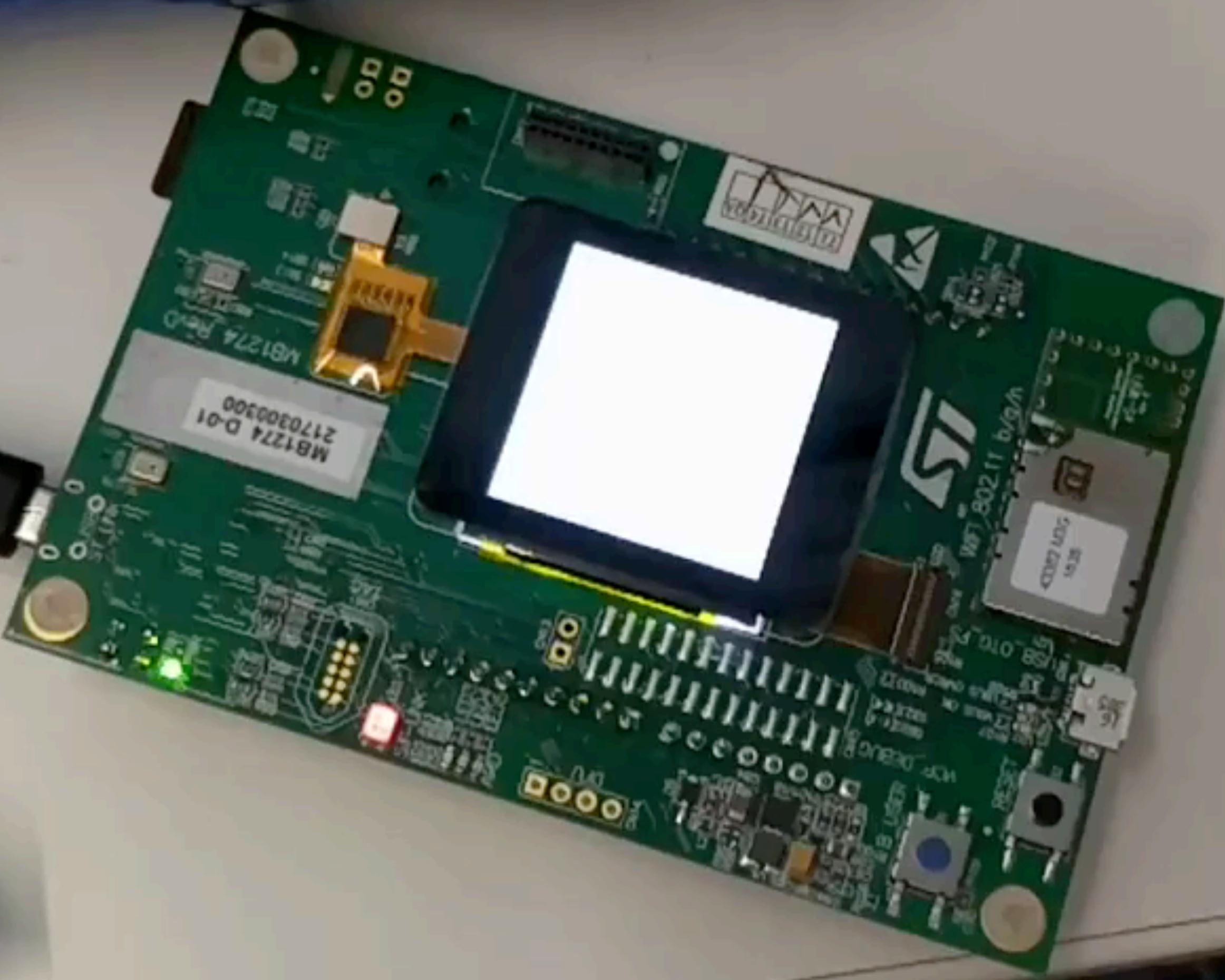
**Michael Bartling**  
Arm



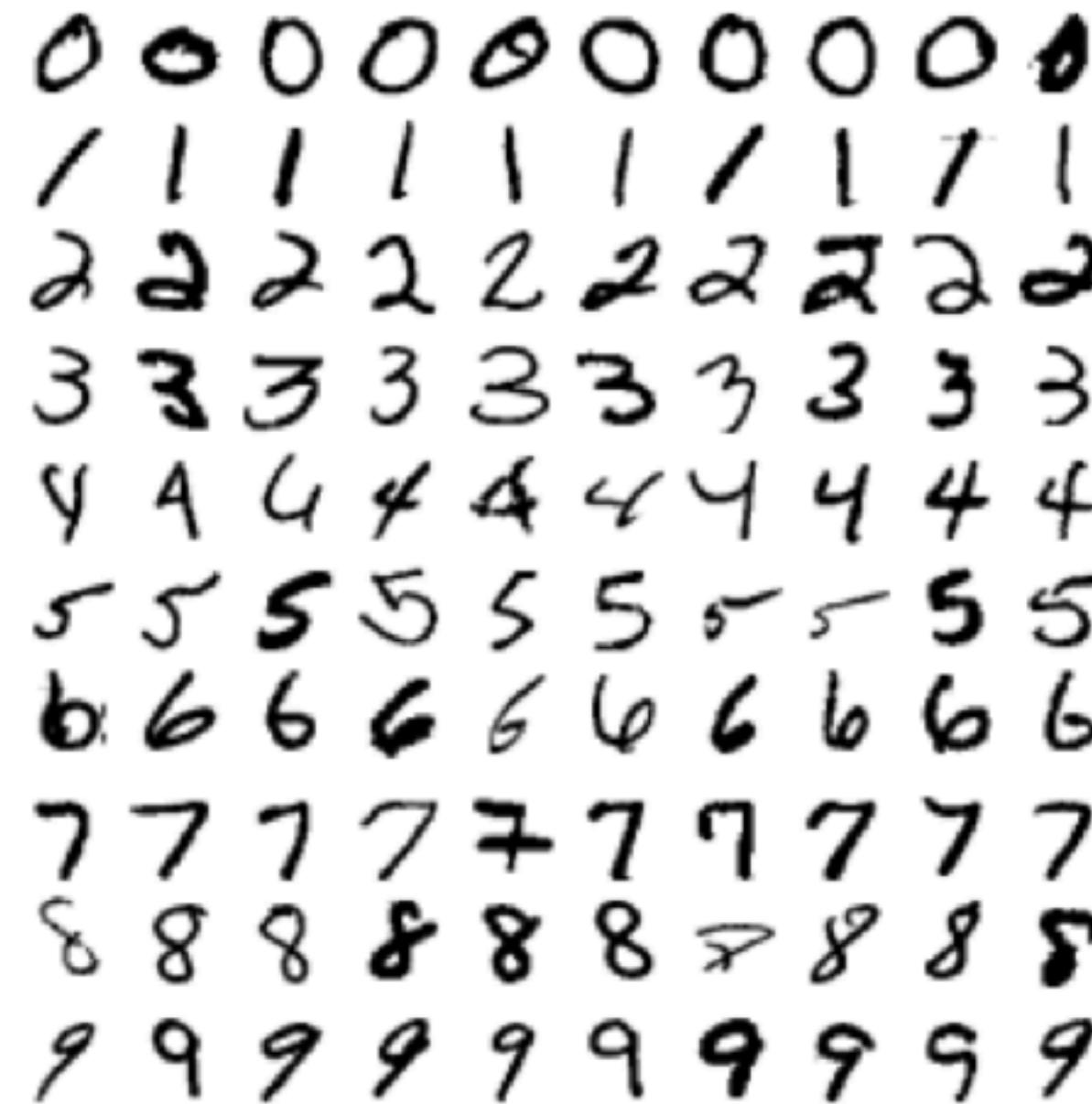
**Dboy Liao**  
Piniko



**Kazami Hsieh**  
Academia Sinica



# How?



A 10x10 grid of handwritten digits, likely from the MNIST dataset. The digits are drawn in black ink on a white background. They are arranged in two rows: the first row contains digits 0 through 4, and the second row contains digits 5 through 9. Each digit is a unique, hand-drawn style.

0	0	0	0	0	0	0	0	0	0
1	1	1	1	1	1	1	1	1	1
2	2	2	2	2	2	2	2	2	2
3	3	3	3	3	3	3	3	3	3
4	4	4	4	4	4	4	4	4	4
5	5	5	5	5	5	5	5	5	5
6	6	6	6	6	6	6	6	6	6
7	7	7	7	7	7	7	7	7	7
8	8	8	8	8	8	8	8	8	8
9	9	9	9	9	9	9	9	9	9

MNIST data set

Training set: 60,000 images

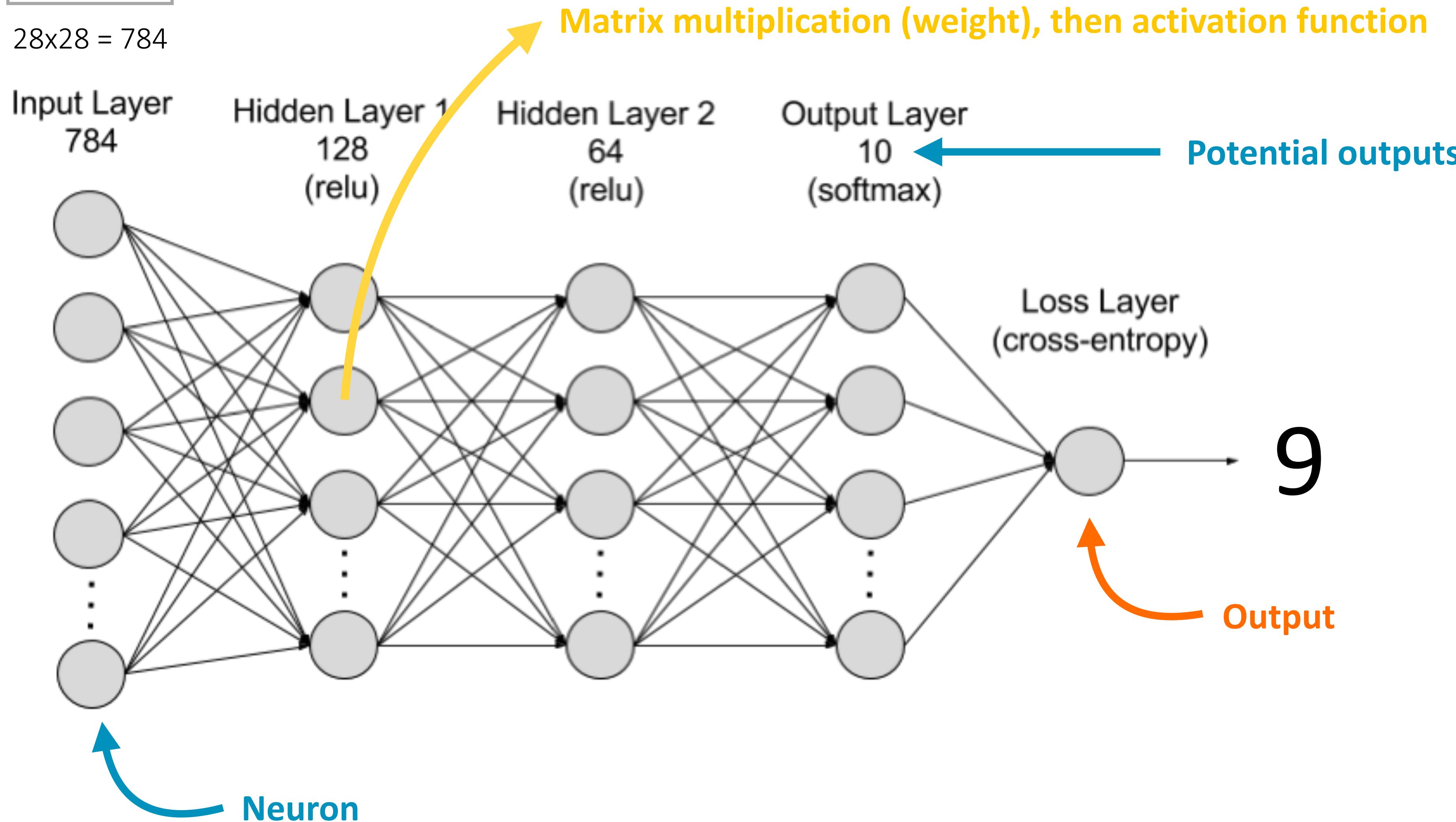
Every drawing is downsampled to 28x28 pixels

Supervised learning through backpropagation

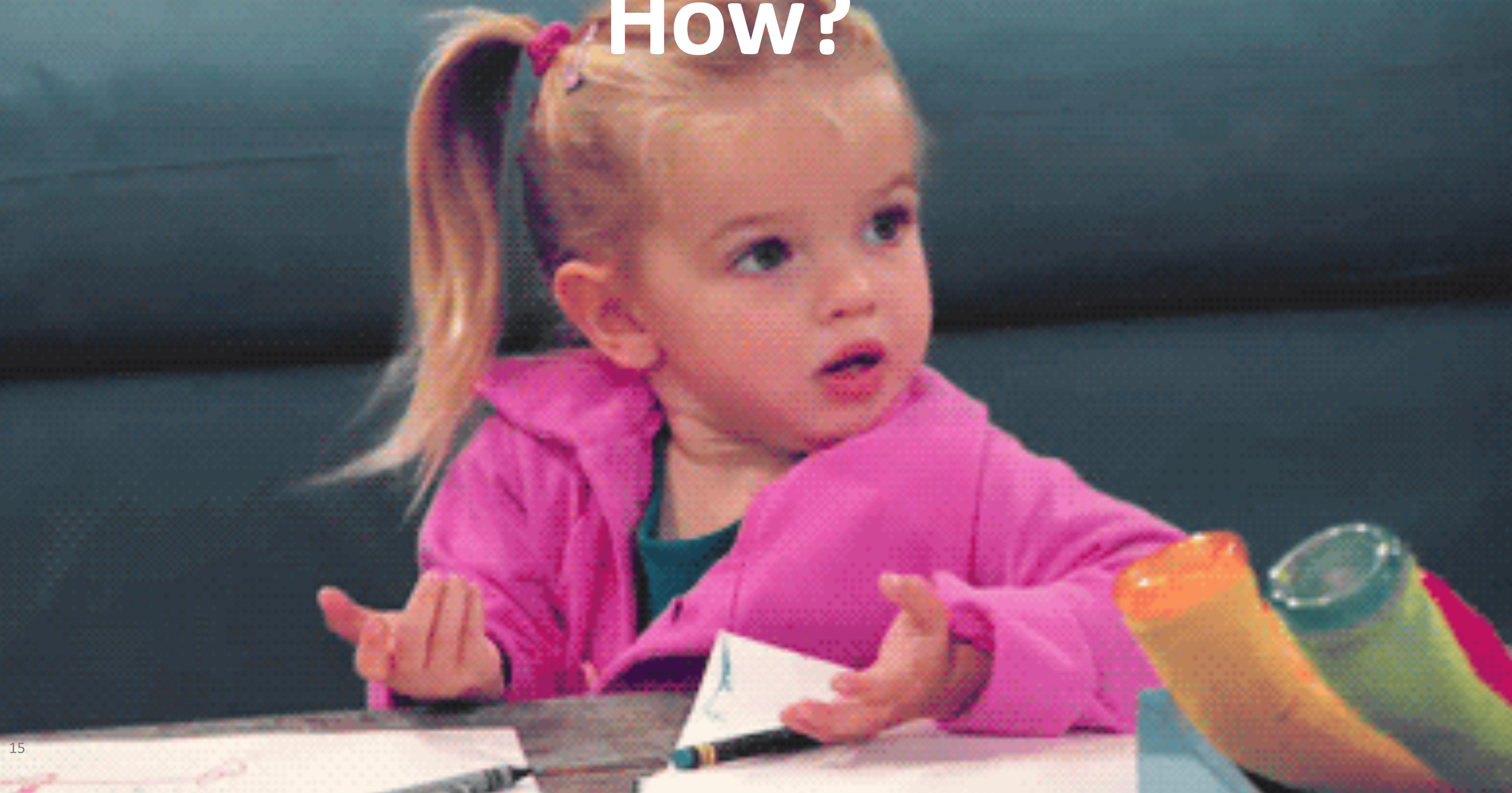
9

$$28 \times 28 = 784$$

# Multi-layer perceptron (MLP) classification



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)

$$1 \text{ byte} * (784 + (128 * 784) + 128) = 98.891 \text{ kB}$$

# 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

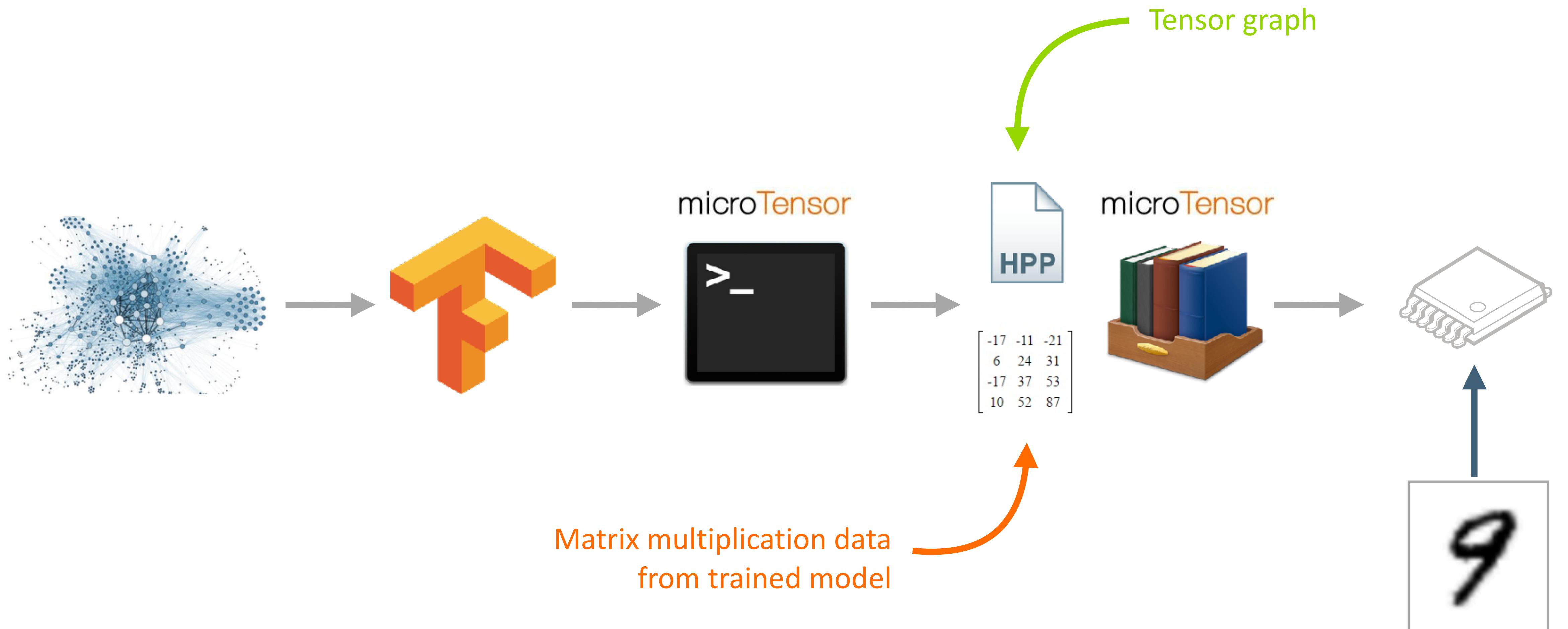
Flash tensor

Sparse tensor

Networked tensor

Tensors can be paged to fit larger networks

# Workflow



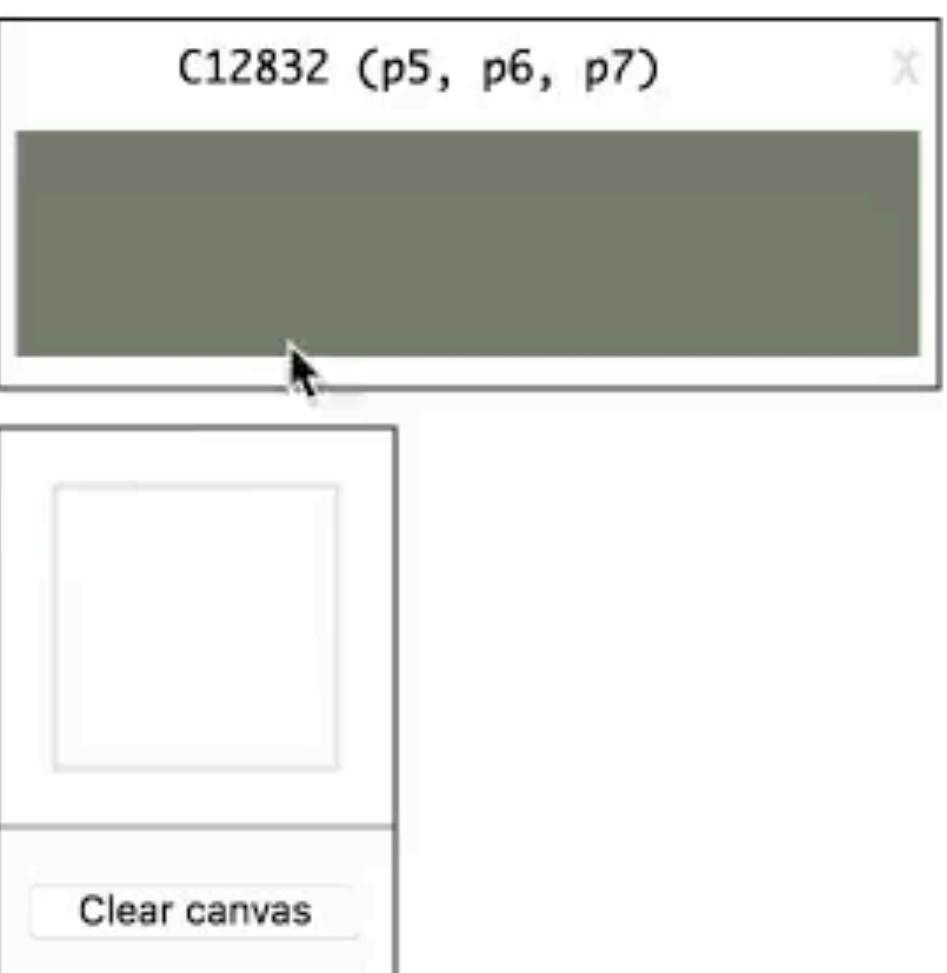
# Developing using the simulator

## Arm Mbed OS simulator

[How to debug](#) | [GitHub project](#)

```
uTensor          Load demo      Run      + Add component
```

```
1  /**
2   * This is a demo application for uTensor - an AI inference library for
3   * deep learning on small microcontrollers.
4   * It's trained to recognize handwritten digits via the MNIST data set.
5   *
6   * See https://github.com/utensor/utensor
7   */
8
9 #include "mbed.h"
10 #include "tensor.hpp"
11 #include "deep_mnist_mlp.hpp"
12 #include "emscripten.h"
13 #include "C12832.h"
14
15 C12832 lcd(SPI_MOSI, SPI_SCK, SPI_MISO, p8, p11);
16
17 EventQueue queue;
18 InterruptIn btn(BUTTON1);
19
20 void run_mlp() {
21     EM_ASM{
22         // this writes the content of the canvas (in the simulator) to /fs/tmp.idx
23         window.dumpCanvasToTmpFile();
24     });
25
26     // invoke the MLP algorithm against the temp file (just saved from canvas)
27     int prediction = runMLP("/fs/tmp.idx");
28     lcd.cls();
29     lcd.locate(3, 13);
30     lcd.printf("Predicted: %d", prediction);
31 }
```



## Serial output

Deep MLP on Mbed (Trained with Tensorflow)

Draw a number (0-9) on the canvas, then hit the button on the board to run MLP algorithm

Please draw the image as large as possible \*in the gray box\* for best results

# 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