EFFICIENT AI ON EMBEDDED SYSTEMS

Diederik M. Roijers
HU: Microsystems technology
VUB: AI laboratory

HOGESCHOOL UTRECHT

CO-AUTHORS

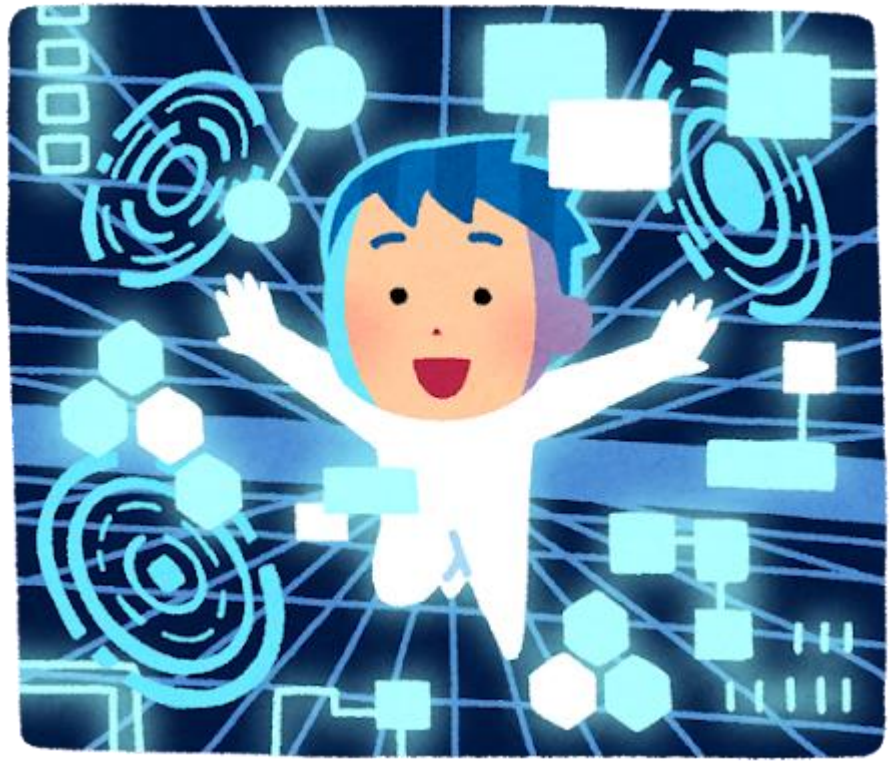
Jeff van de Kamer

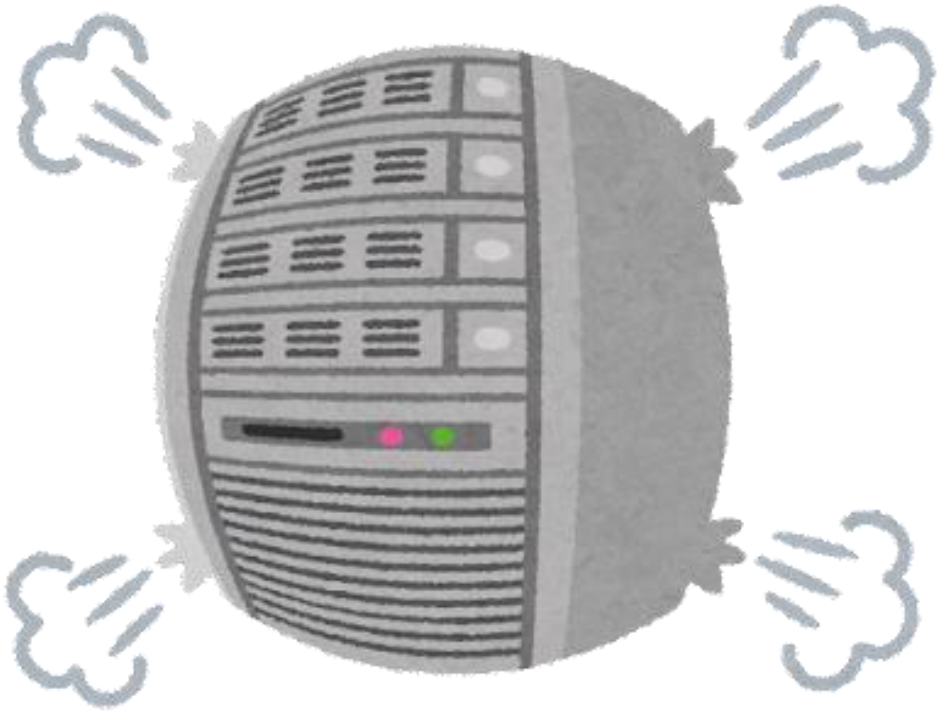
Maaike Hovenkamp

Erik Puik

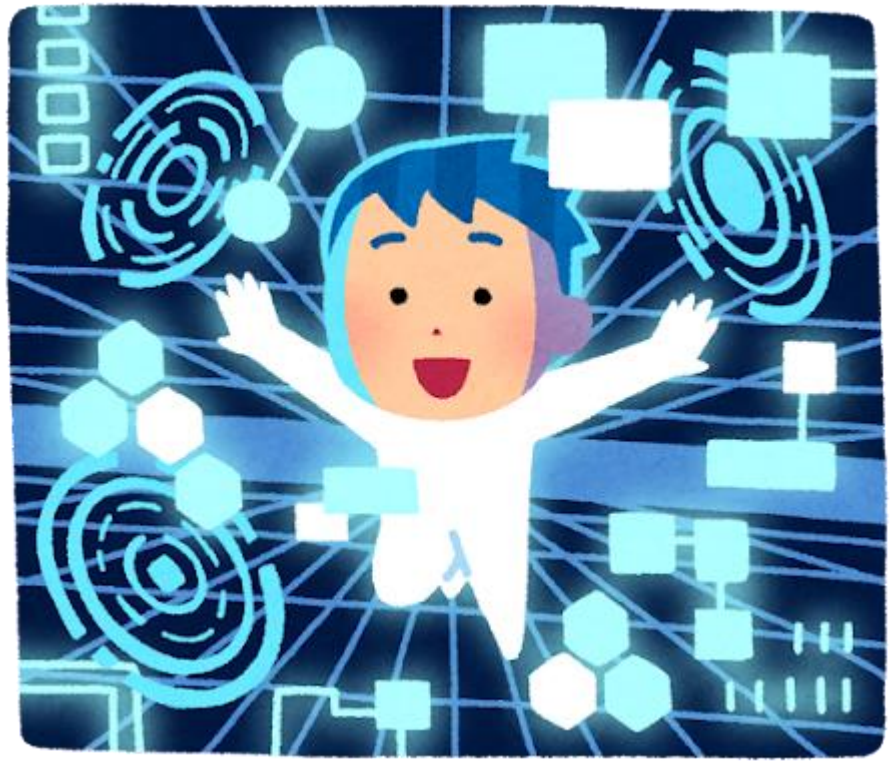
HOGESCHOOL UTRECHT

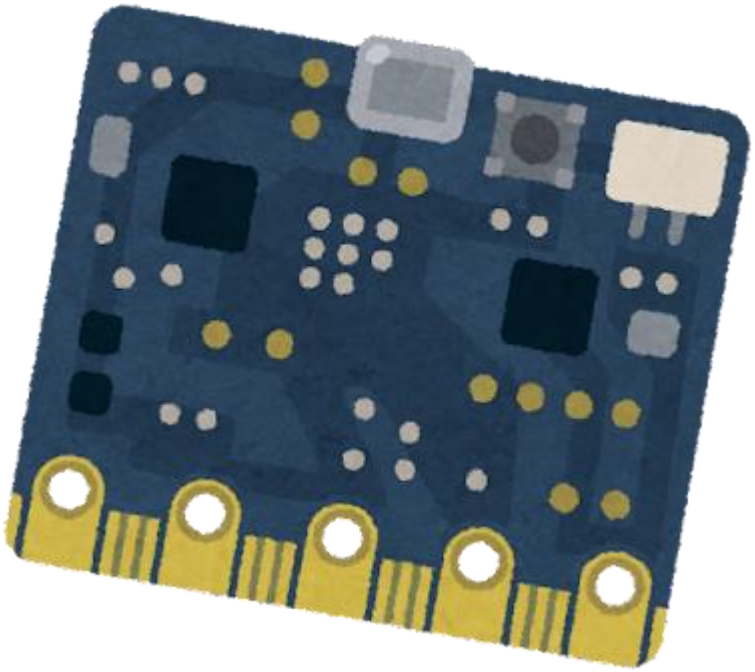
MACHINE LEARNING?





MACHINE LEARNING FOR EMBEDDED SYSTEMS!





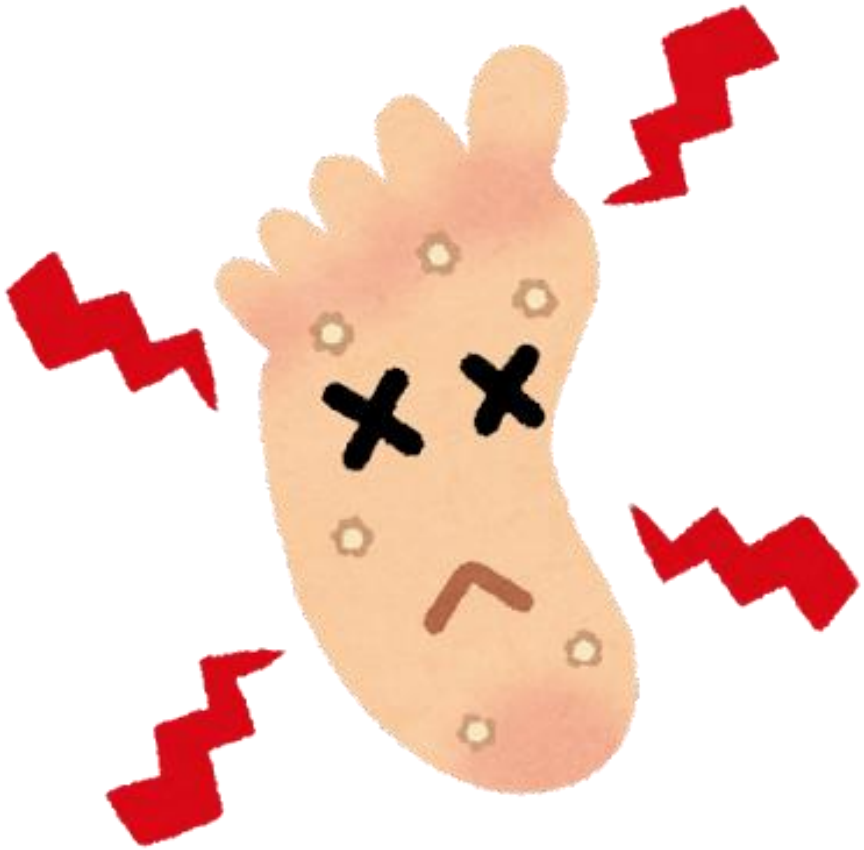
DIABETIC FEET

Ulcers

Compliance

Daily activities

- walking
- walking stairs

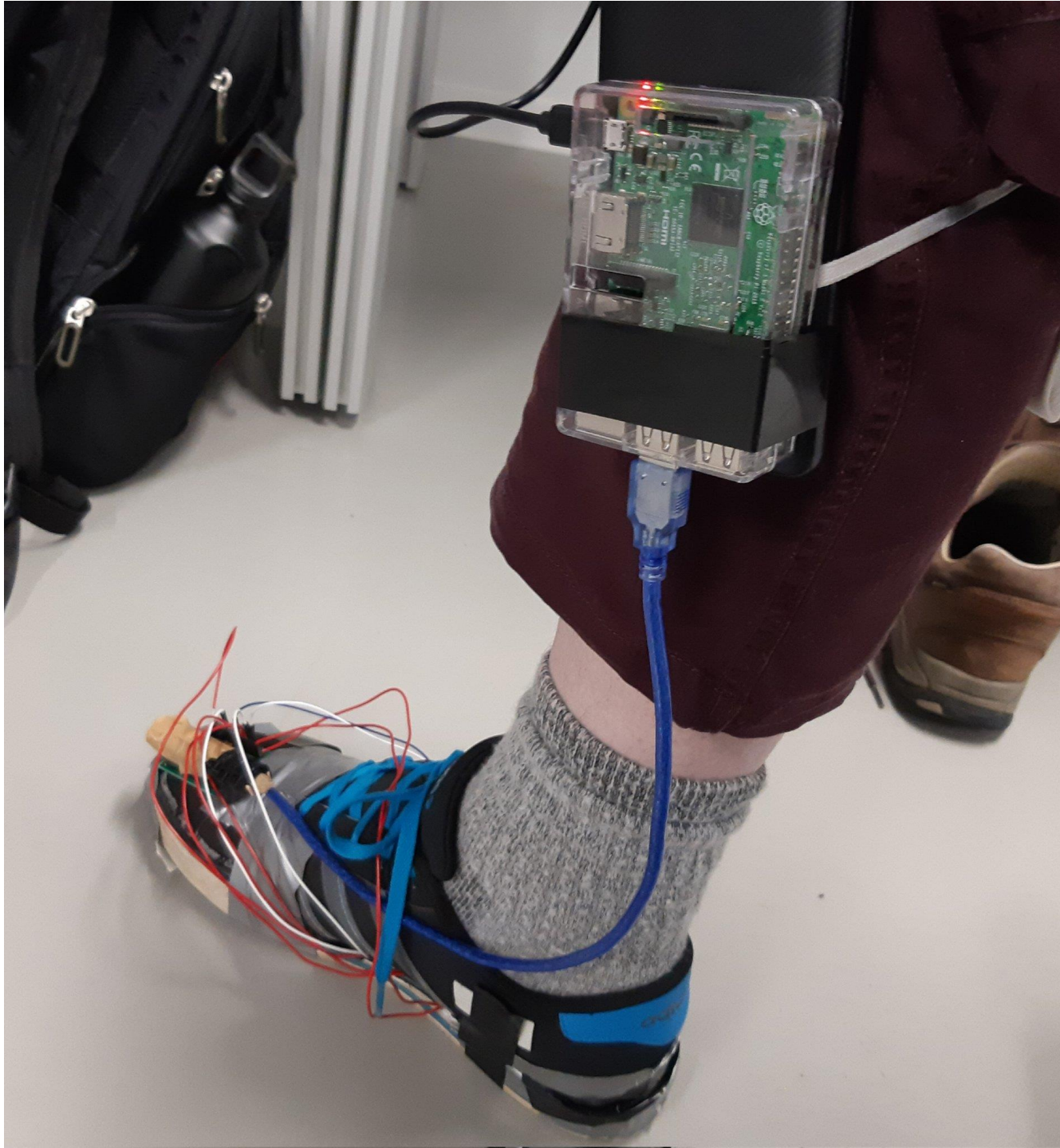


HOGESCHOOL UTRECHT

SMART INSOLES

Smart insoles

- 3D-printed insole
- pressure re-distribution
- equipped with sensors
- Arduino Nano 33 BLE Sense (Cortex-m4)
- Raspberry pi for data gathering only



ACTIVITY RECOGNITION

Sensor data

- pressure sensors
- gyroscope
- accelerometer

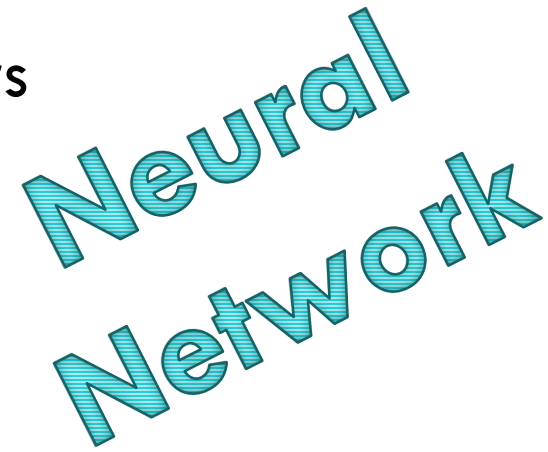
Activity monitoring

- sitting (compliance)
- not worn
- standing
- walking
- walking stairs

ACTIVITY RECOGNITION

Sensor data

- pressure sensors
- gyroscope
- accelerometer



Activity monitoring

- sitting (compliance)
- not worn
- standing
- walking
- walking stairs

REQUIREMENTS

Accurate

Real-time

Energy-efficient

EFFICIENT FAST NEURAL NETWORK

Fewer parameters

- Smaller network
- Max pooling

Smaller parameters

- Full integer quantisation
- 3~4 times smaller

Tensorflow Lite

```
model = Sequential(name "Lightwegeight_model")


# (strong) resolution reduction model.add(MaxPooling2D((6,1),input_shape =
inputshape)) model.add(Conv2D(3,(3,1),activation = "relu",padding =
    "same",name = "conv1b")) model.add(Conv2D(3,(3,11),activation =
"relu",padding =
    "same",name = "conv1"))


# (light) Dropout
model.add(Dropout(0.2))


# flattening and resolution reduction model.add(MaxPooling2D(2,1))
model.add(Flatten())


# output layer
model.add(Dense(5,activation = "softmax",name = "Output"))

# Adam optimizer (with reduced learning rate) is used, as this optimizer
has shown the greatest results in tests done previous to this one.
model.compile(optimizer = tf.keras.optimizers.Adam(0.00001),
    loss='categorical_crossentropy',metrics=['accuracy'])

return model
```

HOGESCHOOL UTRECHT

PIPELINE

Gather data on device

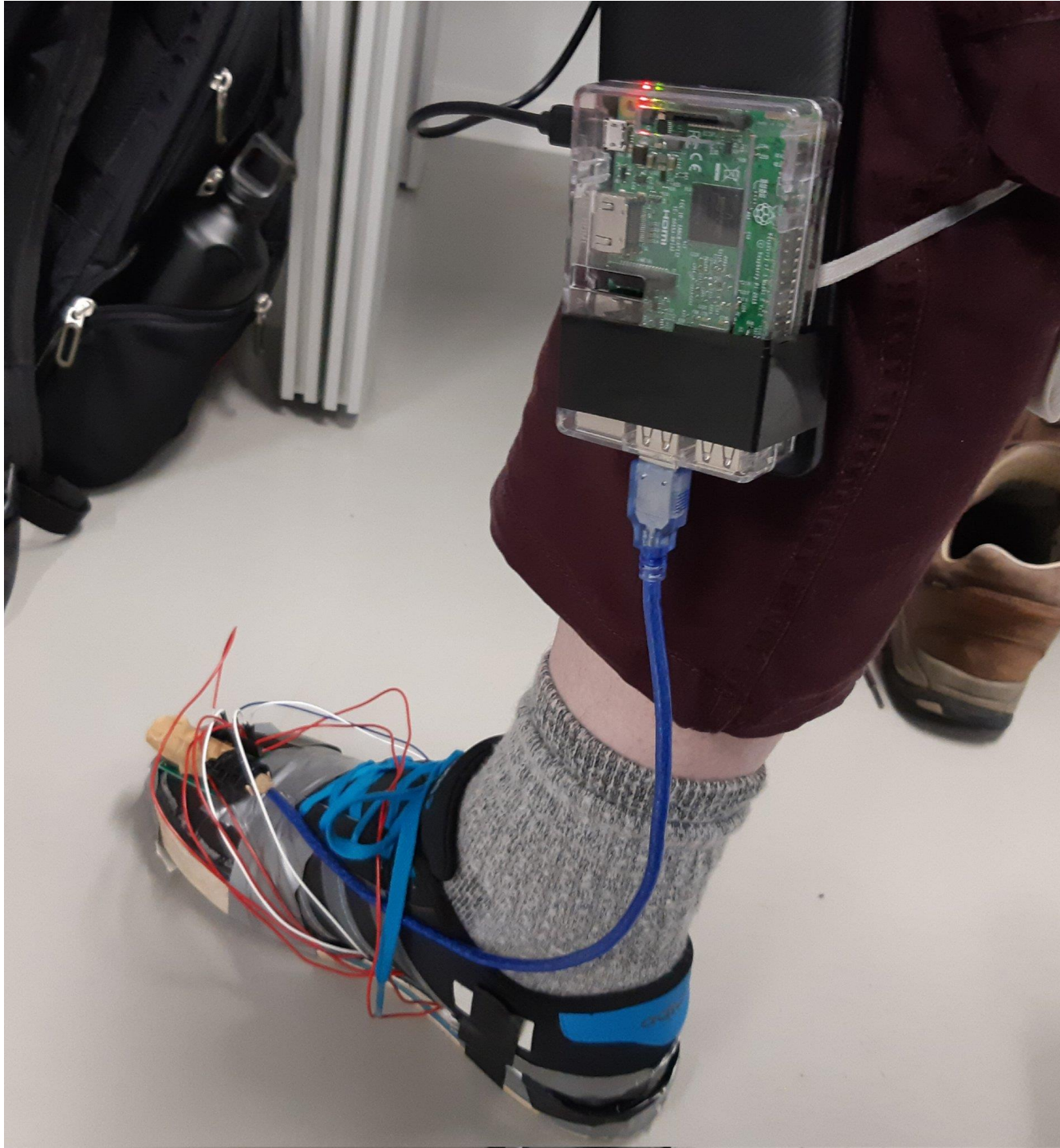
Train NN on PC using data

Quantise the model (integer weights)

Compile NN model together with C++ code for microprocessor

Deploy

Classify activities



DATA FRAME

180 lines

~1.5 sec

COM10

```
10:54:38.373 -> -0.05,0.01,0.24,-0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.373 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.373 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.373 -> -0.05,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.373 -> -0.05,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.419 -> -0.06,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.419 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.16,0.07,0.00
10:54:38.419 -> -0.06,0.01,0.24,-0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.419 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.419 -> -0.06,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.14,0.06,0.00
10:54:38.464 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.464 -> -0.05,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.464 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.464 -> -0.05,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.464 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.517 -> -0.05,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.517 -> -0.05,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.16,0.06,0.00
10:54:38.517 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.517 -> -0.06,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.517 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.562 -> -0.06,0.01,0.24,-0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.562 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.562 -> -0.05,0.01,0.24,-0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.562 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.562 -> -0.05,0.01,0.24,-0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.603 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.603 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.603 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.603 -> -0.05,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.603 -> -0.06,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.16,0.06,0.00
10:54:38.651 -> -0.05,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.651 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.651 -> -0.06,0.01,0.24,-0.00,0.00,0.00,0.00,0.00,0.16,0.06,0.00
10:54:38.651 -> -0.05,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.651 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.698 -> -0.05,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.698 -> -0.06,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.698 -> -0.05,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.698 -> -0.05,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.698 -> -0.06,0.01,0.24,0.00,0.00,-0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.745 -> -0.06,0.01,0.24,-0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.745 -> -0.05,0.01,0.24,-0.00,0.00,-0.00,0.00,0.00,0.15,0.06,0.00
10:54:38.745 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.16,0.06,0.00
10:54:38.745 -> -0.06,0.01,0.24,0.00,0.00,0.00,0.00,0.00,0.15,0.07,0.00
10:54:38.745 -> -0.06,0.01,0.24,0.00,0.00,0.0
```

HOGESCHOOL UTRECHT

CODE PYTHON

```
model = Sequential(name "Lightwegeight_model")


# (strong) resolution reduction
model.add(MaxPooling2D((6,1),input_shape = inputshape))
model.add(Conv2D(3,(3,1),activation = "relu",padding = "same",name = "conv1b"))
model.add(Conv2D(3,(3,11),activation = "relu",padding = "same",name = "conv1"))


# (light) Dropout
model.add(Dropout(0.2))


# flattening and resolution reduction
model.add(MaxPooling2D(2,1)) model.add(Flatten())


# output layer
model.add(Dense(5,activation = "softmax",name = "Output"))

# Adam optimizer (with reduced learning rate) is used, as this optimizer has shown the
greatest results in tests done previous to this one.
model.compile(optimizer = tf.keras.optimizers.Adam(0.00001),
    loss='categorical_crossentropy',metrics=['accuracy'])

return model
```

HOGESCHOOL UTRECHT

Arduino_AI_BLE | Arduino 1.8.15

Bestand Bewerken Schets Hulpmiddelen Help

Arduino_AI_BLE § | Architecture7-quantized.h | Sensorbuffer.h | sensorhandler.h

```
234    while(central.connected()){
235        // read sensors
236        std::array<float,11> measurements = getMeasurements();
237        //update buffer
238        framebuffer.addMeasurements(measurements);
239
240        if(updatecount % kInferenceInterval == 0){
241          // insert timeframe into neural network
242          insertTensor(model_input,framebuffer.getFrame());
243          updatecount = 0;
244
245          // Run inference & measure duration
246          unsigned long t1 = millis();
247          TfLiteStatus invoke_status = interpreter->Invoke();
248          unsigned long t2 = millis();
249
250          if (invoke_status != kTfLiteOk) {
251              error_reporter->Report("Invoke failed on input");
252              return;
253          }
254
255          //calculate duration
256          unsigned long inferencetime = t2-t1;
257
258          // obtain output
259          float class1 = model_output->data.f[0]; // not being worn
260          float class2 = model_output->data.f[1]; // standing
261          float class3 = model_output->data.f[2]; // sitting
262          float class4 = model_output->data.f[3]; // walking
263          float class5 = model_output->data.f[4]; // climbing / descending stairs
264
265
266          // handle output
267          updateServices(inferencetime,class1,class2,class3,class4,class5); // output data over BLE
```

Arduino_AI_BLE - Architecture7-quantized.h | Arduino 1.8.15

Bestand Bewerken Schets Hulpmiddelen Help

Arduino_AI_BLE § | Architecture7-quantized.h | Sensorbuffer.h | sensorhandler.h

```
1  #ifndef ARCHITECTURE_7_H
2  #define ARCHITECTURE_7_H
3
4  unsigned int Architecture7_quantized_len = 8896;
5
6  unsigned char Architecture7_quantized[] = {
7    0x1c, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x12, 0x00,
8    0x1c, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x10, 0x00, 0x14, 0x00,
9    0x00, 0x00, 0x18, 0x00, 0x12, 0x00, 0x00, 0x00, 0x03, 0x00, 0x00, 0x00,
10   0x0c, 0x22, 0x00, 0x00, 0x44, 0x14, 0x00, 0x00, 0x2c, 0x14, 0x00, 0x00,
11   0x3c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x01, 0x00, 0x00, 0x00,
12   0x0c, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00, 0x04, 0x00, 0x08, 0x00,
13   0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
14   0x13, 0x00, 0x00, 0x00, 0x6d, 0x69, 0x6e, 0x5f, 0x72, 0x75, 0x6e, 0x74,
15   0x69, 0x6d, 0x65, 0x5f, 0x76, 0x65, 0x72, 0x73, 0x69, 0x6f, 0x6e, 0x00,
16   0x11, 0x00, 0x00, 0x00, 0xdc, 0x13, 0x00, 0x00, 0xc8, 0x13, 0x00, 0x00,
17   0x9c, 0x13, 0x00, 0x00, 0x78, 0x13, 0x00, 0x00, 0x54, 0x13, 0x00, 0x00,
18   0x30, 0x13, 0x00, 0x00, 0x1c, 0x02, 0x00, 0x00, 0xf8, 0x01, 0x00, 0x00,
19   0xb4, 0x00, 0x00, 0x00, 0xa8, 0x00, 0x00, 0x00, 0x94, 0x00, 0x00, 0x00,
20   0x80, 0x00, 0x00, 0x00, 0x6c, 0x00, 0x00, 0x00, 0x58, 0x00, 0x00, 0x00,
21   0x44, 0x00, 0x00, 0x00, 0x30, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
22   0x9a, 0xeb, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x10, 0x00, 0x00, 0x00,
23   0x31, 0x2e, 0x35, 0x2e, 0x30, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
24   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
25   0x00, 0x00, 0x00, 0x00, 0x74, 0xe1, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
26   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x84, 0xe1, 0xff, 0xff,
27   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
28   0x94, 0xe1, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
29   0x00, 0x00, 0x00, 0x00, 0xa4, 0xe1, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
30   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0xb4, 0xe1, 0xff, 0xff,
31   0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
32   0xc4, 0xe1, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00,
33   0x00, 0x00, 0x00, 0x00, 0xd4, 0xe1, 0xff, 0xff, 0x00, 0x00, 0x00, 0x00,
34   0x2a, 0xec, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x29, 0x01, 0x00, 0x00,
35   0xf4, 0x77, 0x26, 0xf1, 0xde, 0xcd, 0x4b, 0x28, 0x1f, 0xc1, 0x09, 0x12,
36   0x63, 0xfe, 0x18, 0xf1, 0xf2, 0x10, 0xdf, 0xfe, 0xb7, 0xe1, 0xd2, 0xf5,
```

HOGESCHOOL UTRECHT

HOGESCHOOL UTRECHT

RESULTS

Accuracy: 86% (max 99.5%, but 1.5s per data frame)

Energy usage: 0.10W (23mA*4.51V; to further optimise)

Prediction speed: 50~70ms per data frame