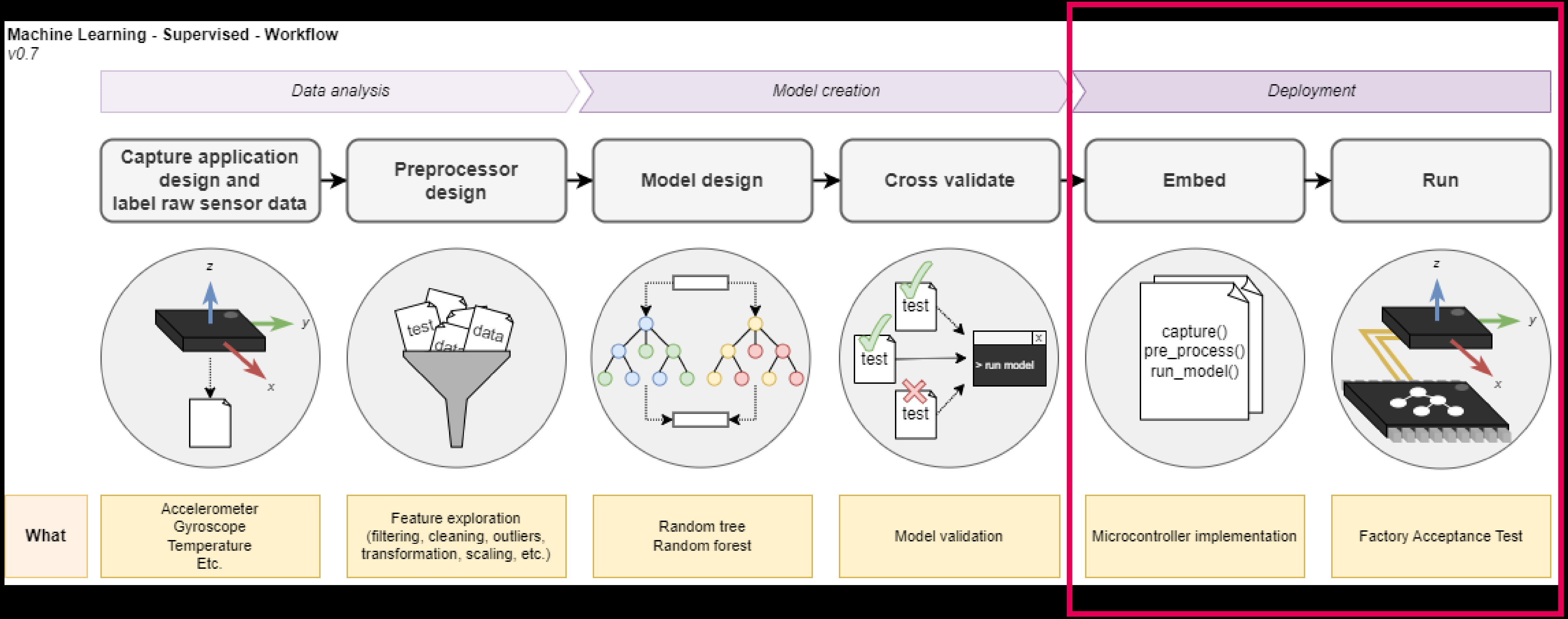
S6-ESE-AI

DEPLOYMENT

JEROEN VEEN
HUGO ARENDS

HAN_UNIVERSITY OF APPLIED SCIENCES

WORKFLOW



HAN_UNIVERSITY OF APPLIED SCIENCES

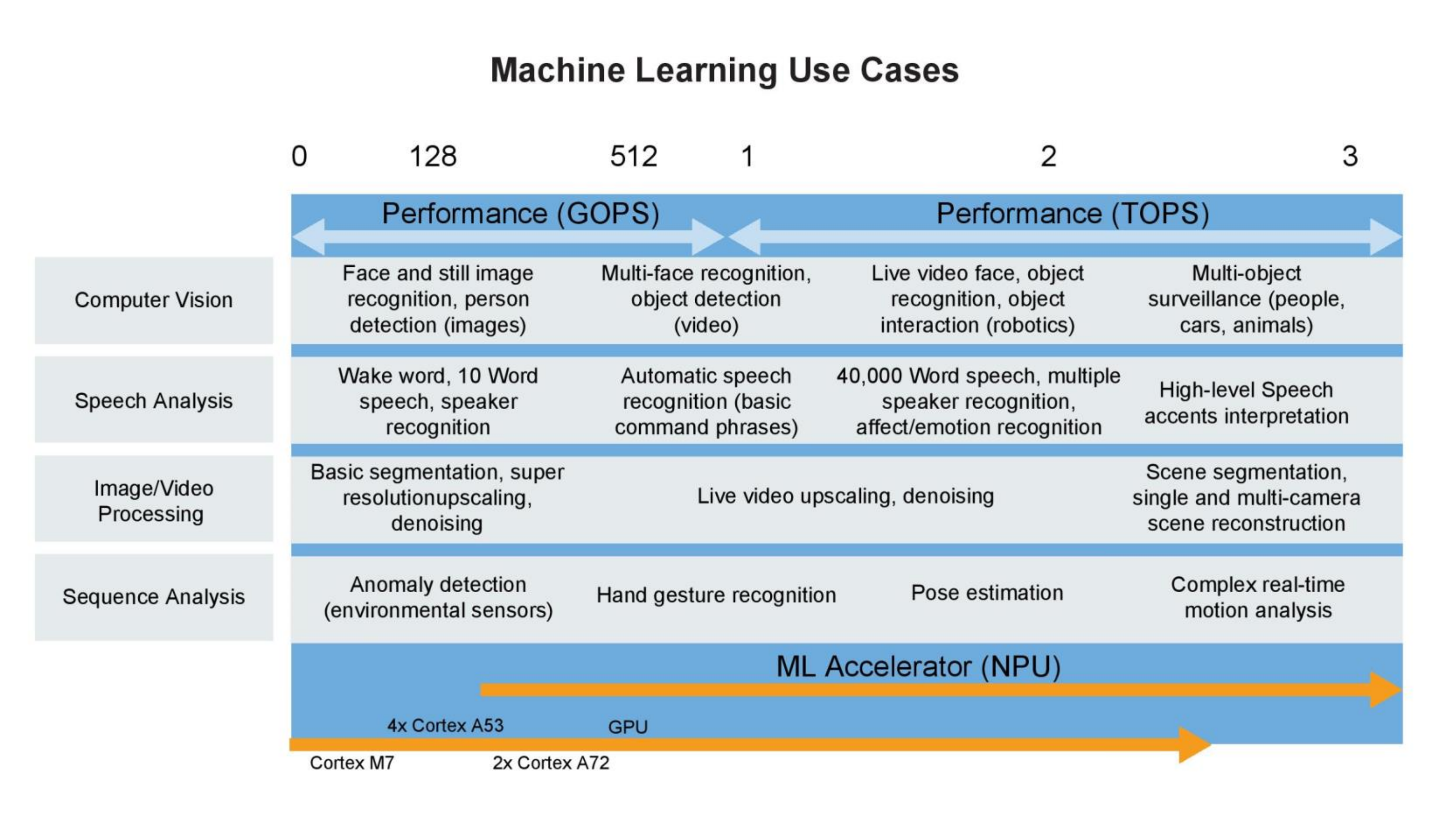
AGENDA

- Code generation
- Reducing computational complexity
- Accelerators
- Platforms
- User Acceptance Testing

REDUCING COMPUTATIONAL COMPLEXITY

- Quantization
- Pruning of trees
- Separable convolutions

EDGE ML USE CASES



Source: NXP

HW ACCELERATORS

- Get started with the Dev Board Micro | Coral
- AI Accelerator Hailo-8 M.2 AI Module | Superior Edge Performance
- Jetson TX2 for next-level performance | NVIDIA

ML FRAMEWORKS

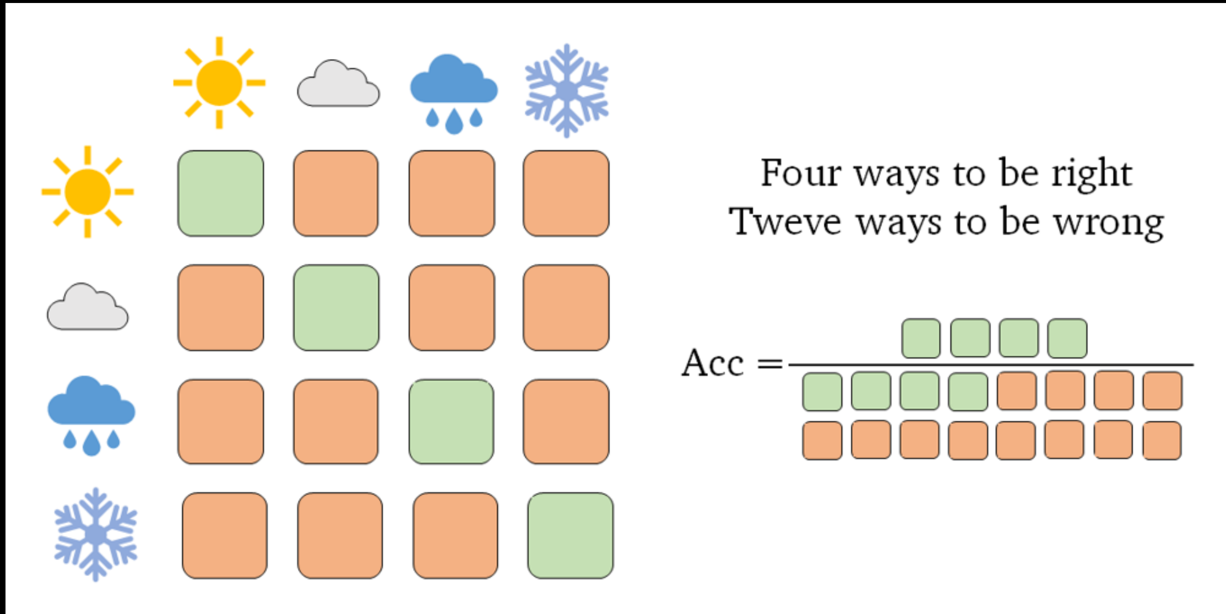
- TinyML
  Focuses on ultra-low power and small footprint ML, ideal for microcontroller-based systems
- Tensorflow Lite
  Well-suited for mobile and edge devices, with a broad range of model support, e.g. TensorFlow.
- PyTorch Mobile
  Good for deploying PyTorch models on mobile devices, with a growing ecosystem and support.
- OpenVINO
  Optimized for deploying on Intel hardware, including CPUs, GPUs, and VPUs.
- NVIDIA Jetson
  Targeted at edge computing, leveraging NVIDIA GPUs for high-performance inference.
- Caffe2
  Known for its speed and efficiency, but with a smaller community

CRITERIA FOR COMPARING ML FRAMEWORKS

- Performance
- Speed
- Resource efficiency
- Ease of use and integration
- Platform and hardware compatibility
- Community and ecosystem
- Security and privacy
- Compatibility and support
- Update frequency and maintenance

USER ACCEPTANCE TESTING

- Determine confusion matrix
- Compute classification report



Confusion matrix and accuracy for a four-classes problem

```
>>> from sklearn.metrics import classification_report
>>> y_true = [0, 1, 2, 2, 2]
>>> y_pred = [0, 0, 2, 2, 1]
>>> target_names = ['class 0', 'class 1', 'class 2']
>>> print(classification_report(y_true, y_pred, target_names=target_names))
              precision    recall  f1-score   support

     class 0       0.50      1.00      0.67         1
     class 1       0.00      0.00      0.00         1
     class 2       1.00      0.67      0.80         3

    accuracy                           0.60         5
   macro avg       0.50      0.56      0.49         5
weighted avg       0.70      0.60      0.61         5

>>> y_pred = [1, 1, 0]
>>> y_true = [1, 1, 1]
>>> print(classification_report(y_true, y_pred, labels=[1, 2, 3]))
              precision    recall  f1-score   support

           1       1.00      0.67      0.80         3
           2       0.00      0.00      0.00         0
           3       0.00      0.00      0.00         0

   micro avg       1.00      0.67      0.80         3
   macro avg       0.33      0.22      0.27         3
weighted avg       1.00      0.67      0.80         3
```

https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html

GENERALIZATION

- Test the model with new, unseen data to assess its generalization capability.
- Use relevant metrics (like accuracy, precision, recall) to evaluate performance.

EMBED

Input	Process	Output
./data/model/dtc_model.gz	./model_embedding/ code_generator_dtc2c.py	*./data/model_embedding/ dtc_model.c*
./data/model/dtc_train_bunch.csv		
		./data/model/.png*

EMBED

./data/model_embedding/dtc_model.c

```c
typedef enum
{
    left_right = 0,
    stationary = 1,
    up_down = 2,
}dtc_t;
```

Enumerated type containing each label

EMBED

./data/model_embedding/dtc_model.c

```
/*
 * \brief Decision tree classifier
 *
 * Decision tree classifier based on
 * the following input characteristics:
 *   BLOCK_SIZE: 100
 *   BLOCK_TYPE: BLOCK
 *
 * \return dtc_t
 *   0  left_right
 *   1  stationary
 *   2  up_down
 */
```

Documented function

EMBED

./data/model_embedding/dtc_model.c

```c
dtc_t dtc(const float y_out_fir_rescale_variance,
    const float x_out_fir_rescale_variance)
{
    dtc_t ret;

    if(y_out_fir_rescale_variance <= 0.000847f)
    {
         ret = stationary;
    }
    else // y_out_fir_rescale_variance > 0.000847f
    {
        if(x_out_fir_rescale_variance <= 0.033878f)
        {
            ret = up_down;
        }
```

Function implementation

EMBED

main.c – BLOCK example

```c
// Handle the data as soon as new data is available
// mma8451 accelerometer Output Data Rate (ODR) is set to 100 Hz
if(mma8451_ready_flag)
{
    // Set initial timestamp
    ms1 = ms;

    // Clear the flag
    mma8451_ready_flag = false;

    // Reads the data in three global variables: x_out_mg, y_out_mg and
    // z_out_mg
    mma8451_read();
```

EMBED

main.c – BLOCK example

```
    // TODO Implement filter function as required by the application.

    // Filter accelerometer data
    x_out_mg = fir(x_out_mg, fir_coefs, fir_x, N_FIR);
    y_out_mg = fir(y_out_mg, fir_coefs, fir_y, N_FIR);
    z_out_mg = fir(z_out_mg, fir_coefs, fir_z, N_FIR);
```

EMBED

main.c – BLOCK example

```c
    // TODO Implement normalization function as required by the
    //      application.

    // Scale accelerometer data
    const float from[2] = {-1000.0f, 1000.0f};
    const float to[2] = {-1.0f, 1.0f};

    x_out_mg = rescale(x_out_mg, from, to);
    y_out_mg = rescale(y_out_mg, from, to);
    z_out_mg = rescale(z_out_mg, from, to);
```

EMBED

main.c – BLOCK example

```
    // TODO Finish this example by designing an ML model and implement
    //       the generated C code.

    // Add accelerometer data to the buffer
    buffer_x_out[n] = x_out_mg;
    buffer_y_out[n] = y_out_mg;
    buffer_z_out[n] = z_out_mg;

    n++;
```

EMBED

main.c – BLOCK example

```c
    // Buffer full?
    if(n >= N_BUFFER)
    {
        // Reset buffer counter for next block
        n = 0;

        // Calculate features by using feature functions
        float x_out_var = variance(buffer_x_out, N_BUFFER);
        float y_out_var = variance(buffer_y_out, N_BUFFER);

        // Calculate label by using the generated Decision Tree
        // Classifier
        dtc_t label = dtc(x_out_var, y_out_var);
```

EMBED

main.c – BLOCK example

```
        char *label_str = "";

        // Use the calculated label for further processing
        if(label == stationary)
        {
                label_str = "stationary";
        }
        else if(label == up_down)
        {
                label_str = "up_down";
        }
        else if(label == left_right)
        {
                label_str = "left_right";
        }
```

EMBED

main.c – BLOCK example

```
        // Set final timestamp
        ms2 = ms;

        // Print duration and label
        printf("%d,%d,%s\n",
            ms1,
            ms2,
            label_str);
    }
}
```

EMBED

main.c – SLIDING example

```c
// Handle the data as soon as new data is available
// mma8451 accelerometer Output Data Rate (ODR) is set to 100 Hz
if(mma8451_ready_flag)
{
    // Set initial timestamp
    ms1 = ms;

    // Clear the flag
    mma8451_ready_flag = false;

    // Reads the data in three global variables: x_out_mg, y_out_mg and
    // z_out_mg
    mma8451_read();
```

EMBED

main.c – SLIDING example

```c
    // TODO Implement filter function as required by the application.

    // Filter accelerometer data
    x_out_mg = fir(x_out_mg, fir_coefs, fir_x, N_FIR);
    y_out_mg = fir(y_out_mg, fir_coefs, fir_y, N_FIR);
    z_out_mg = fir(z_out_mg, fir_coefs, fir_z, N_FIR);
```

EMBED

main.c – SLIDING example

```
    // TODO Implement normalization function as required by the
    //      application.

    // Scale accelerometer data
    const float from[2] = {-1000.0f, 1000.0f};
    const float to[2] = {-1.0f, 1.0f};

    x_out_mg = rescale(x_out_mg, from, to);
    y_out_mg = rescale(y_out_mg, from, to);
    z_out_mg = rescale(z_out_mg, from, to);
```

EMBED

main.c – SLIDING example

```c
    // TODO Finish this example by designing an ML model and implement
    //      the generated C code.

    // Buffer full?
    if(n >= (N_BUFFER-1))
    {
        // Remove first data item in the buffer and move all others one
        // position.
        memmove(&buffer_x_out[0], &buffer_x_out[1], sizeof(float) * (N_BUFFER-1));
        memmove(&buffer_y_out[0], &buffer_y_out[1], sizeof(float) * (N_BUFFER-1));
        memmove(&buffer_z_out[0], &buffer_z_out[1], sizeof(float) * (N_BUFFER-1));
    }
```

EMBED

main.c – SLIDING example

```c
    // Add accelerometer data to the buffer
    buffer_x_out[n] = x_out_mg;
    buffer_y_out[n] = y_out_mg;
    buffer_z_out[n] = z_out_mg;

    n++;
```

EMBED

main.c – SLIDING example

```c
    // Buffer full?
    if(n >= N_BUFFER)
    {
        // Set counter at the end of the buffer
        n = N_BUFFER-1;

        // Calculate features by using feature functions
        float y_out_var = variance(buffer_y_out, N_BUFFER);

        // Calculate label by using the generated Decision Tree
        // Classifier
        dtc_t label = dtc(y_out_var);
```

EMBED

main.c – SLIDING example

```c
        char *label_str = "";

        // Use the calculated label for further processing
        if(label == stationary)
        {
                label_str = "stationary";
        }
        else if(label == up_down)
        {
                label_str = "up_down";
        }
        else if(label == left_right)
        {
                label_str = "left_right";
        }
```

EMBED

main.c – SLIDING example

```
        // Set final timestamp
        ms2 = ms;

        // Print duration and label
        printf("%d,%d,%s\n",
            ms1,
            ms2,
            label_str);
    }
}
```

RUN

Input	Process	Output
./../lib/features.c *./../lib/features.h* *./../lib/filters.c* *./../lib/filters.h* *./../lib/normalizations.c* *./../lib/normalizations.h* *./data/model_embedding/dtc_model.c* *main.c*	Source file in microcontroller IDE project	*Factory acceptance test*

RUN

Input	Process	Output
Microcontroller development board	./capturing/data_logger.py	*Terminal window showing printed output*

Microcontroller development board → UART → ./capturing/data_logger.py → *Terminal window showing printed output*

RUN

Terminal

```
Press CTRL+C to quit
Opened COM13 @ 115200bps
139577,139579,stationary
140553,140555,stationary
141529,141531,up_down
142505,142507,up_down
143481,143483,left_right
144457,144459,left_right
145433,145435,left_right
147385,147387,stationary
148361,148363,stationary
Stop app
Closed COM13
```

RUN

Input	Process	Output
Microcontroller development board	./capturing/timestamp_inspector.py	*Terminal window showing timestamp inspected output*

UART

RUN

Terminal

```
Press CTRL+C to quit
Opened COM13 @ 115200bps
[      62] average:   2.0ms | min:   1.0ms | max:   2.0ms
```