Confirm
Smart Manufacturing

Confirm
Smart Manufacturing

An SRAM Optimized Approach for Constant Memory Consumption and Ultra-fast Execution of ML Classifiers on TinyML Hardware

Bharath Sudharsan

A World Leading SFI Research Centre

Science Foundation Ireland SFI For what's next

University of Limerick · Tyndall National Institute · UCC University College Cork, Ireland · NUI Maynooth · OÉ Gaillimh NUI Galway · Cork Institute of Technology · AIT · LIT

Edge Computing - Hardware View

Confirm Smart Manufacturing

ARM Cortex-M0 MCU based BLE beacon

Powerful CPU + basic GPU based SBCs (single board computers)

Edge gateway with GPUs and SSDs

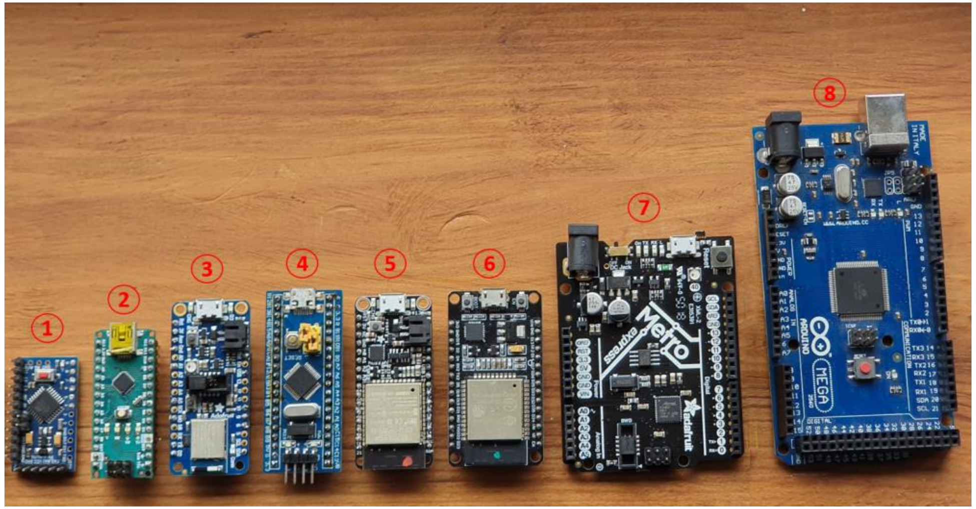
Edge computing hardware: highly resource constrained -> high resource (left to right)

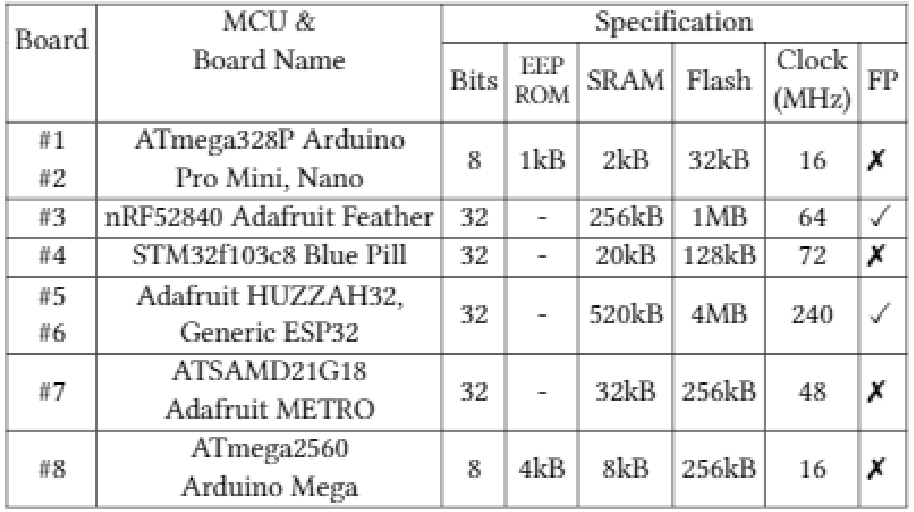
- MCUs and small CPUs: BLE beacons, smart bulbs, smart plugs, TV remotes, fitness bands
- SBCs: Raspberry Pis, BeagleBones, NVIDIA Jetsons, Latte Pandas, Intel NUCs, Google Coral
- GPU accelerated: AWS snowball, Digi gateways, Dell Edge Gateways for IoT, HPE Edgeline

- Why MCU? Roughly **50 billion** MCU chips were shipped in 2020 (market estimates), which far exceeds other chips like GPUs & CPUs (only 100 million units sold)

Edge Computing - Hardware Spec

Confirm Smart Manufacturing





Board	MCU & Board Name	Specification					
		Bits	EEP ROM	SRAM	Flash	Clock (MHz)	FP
#1 #2	ATmega328P Arduino Pro Mini, Nano	8	1kB	2kB	32kB	16	✗
#3	nRF52840 Adafruit Feather	32	-	256kB	1MB	64	✓
#4	STM32f103c8 Blue Pill	32	-	20kB	128kB	72	✗
#5 #6	Adafruit HUZZAH32, Generic ESP32	32	-	520kB	4MB	240	✓
#7	ATSAMD21G18 Adafruit METRO	32	-	32kB	256kB	48	✗
#8	ATmega2560 Arduino Mega	8	4kB	8kB	256kB	16	✗

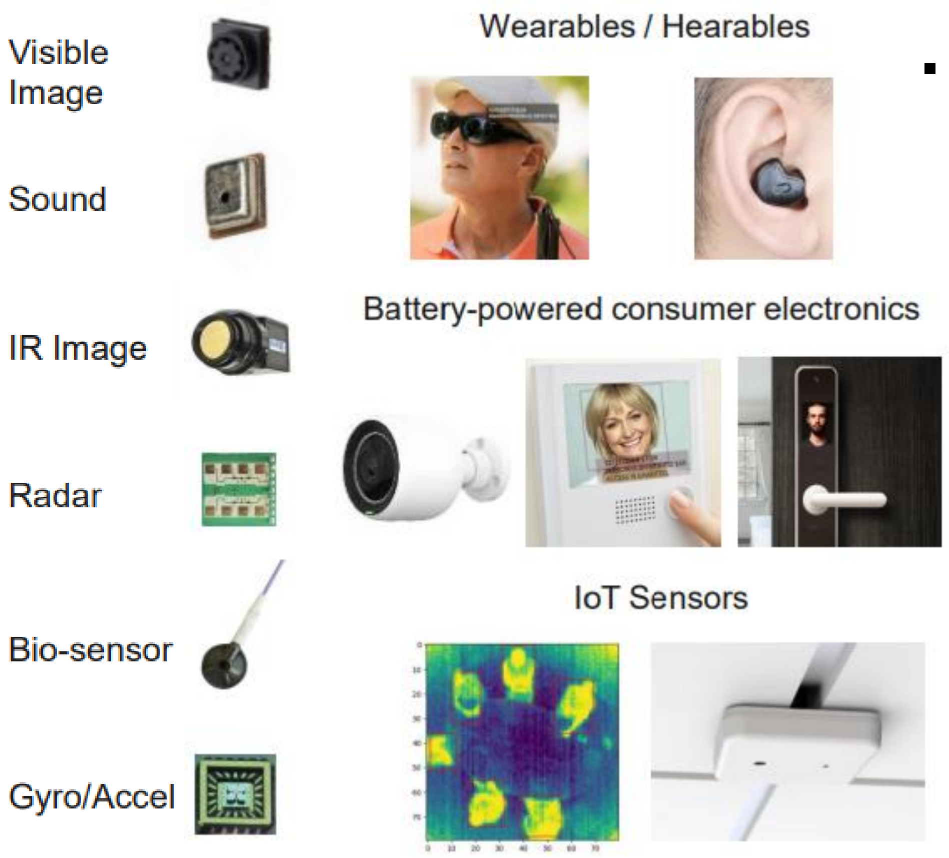
Popular open-source MCU boards with specifications

- No file system. Lack Python support, only C, C++. Few hundred MHz clock speed. Low memory (SRAM, Flash). No multiple cores and parallel execution units

- Lack inbuilt hardware accelerators such as APU (Accelerated Processing Unit), KPU (convolution operation accelerator), FPU (Floating-point accelerator), and FFT (Fourier transform accelerator)

ML meets Edge Computing - TinyML

Confirm
Smart Manufacturing



- Running ML models on MCUs is ultra-low-power machine learning. Also known as TinyML
  - ✓ **Audio: always**-on ML inference for context recognition, spot keywords/wake-words/control-words
  - ✓ **Industry telemetry**: models deployed on MCUs monitor IMUs, motor bearing vibrations, or other sensors to detect anomalies and predict equipment faults
  - ✓ **Image**: object counting, text recognition, visual wake words
  - ✓ **Physiological/behavior**: activity recognition using IMU or EMG data

TinyML Practitioners

Confirm Smart Manufacturing

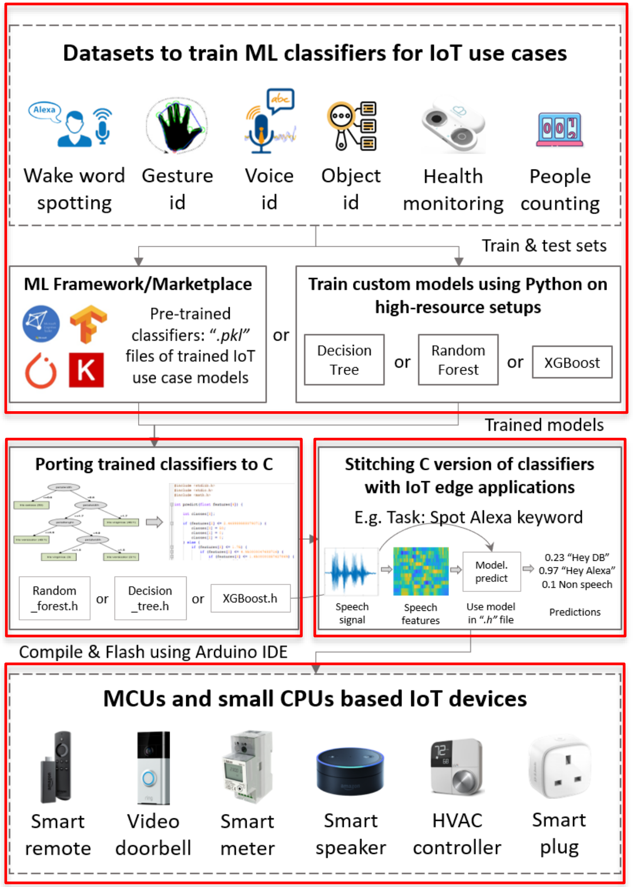
- EdgeML: https://microsoft.github.io/EdgeML/
- Model Zoo: https://github.com/ARM-software/ML-zoo

tinyMLPerf
Working Group Members
RENESAS
GREENWAVES TECHNOLOGIES
HARVARD John A. Paulson School of Engineering and Applied Sciences
OctoML
intel
ST
Southeast Missouri State University
W UNIVERSITY of WASHINGTON
cisco
Google
ambiq micro
ASU Arizona State
arm
Microsoft
Cogneefy
KU LEUVEN
Reality AI
ZOOX
Explainable AI at the Edge
cadence
SiMa ai
AONdevices AI | DSP | ASIC
Red Hat
MEDIATEK
SYNTIANT
SAMSUNG
CIRRUS LOGIC

- Professional Certificate in TinyML
- TinyMLPerf

Who are practicing TinyML - executing ML models on their MCU-based products

Proposed Approach



- Generic End-to-end design flow:
  - ✓ **Step1:** Users can take a pre-trained model or train their own custom classifier model
  - ✓ **Step2:** Port the given classifiers to its MCU executable C version using the SRAM optimized approach
  - ✓ **Step3:** Stitch generated C classifier with the IoT use-case application using SRAM optimized approach
  - ✓ **Step4:** Efficiently execute models on MCUs, small CPUs of IoT devices using SRAM optimized approach

Porting classifier models to plain C

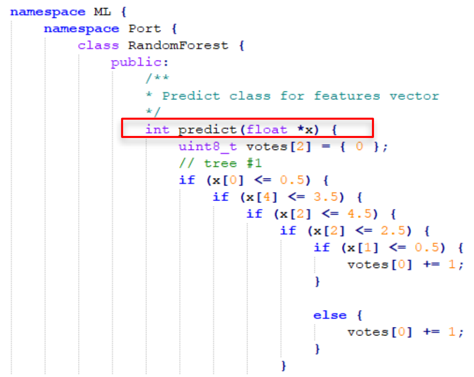
Confirm
Smart Manufacturing

- Proposed porting method
  - ✓ Produces C version DTs which does not depend on the SRAM during execution
  - ✓ **How?** by trading with the larger flash memory
  - ✓ In other words, we propose to sacrifice flash memory in favor of the limited SRAM since it is the scarcest resource in the majority MCUs
- Since our method does not allocate any variables, 0 bytes of SRAM will be consumed to execute the C version of the ported classifier to produce inference results

Porting classifier models to plain C

Confirm Smart Manufacturing

- **Step1**: We train a Random Forest (RF) classifier model using Iris Flowers dataset

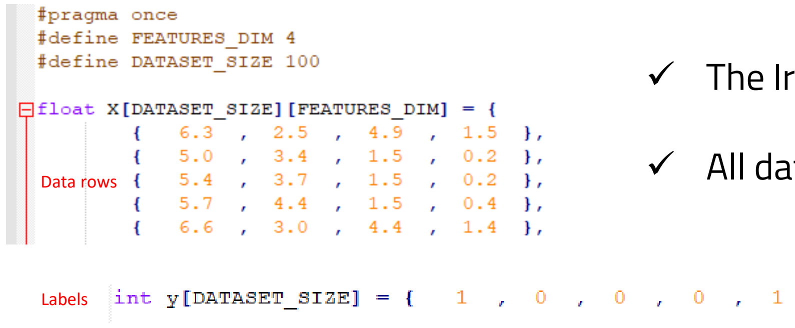


```
namespace ML {
    namespace Port {
        class RandomForest {
        public:
            /**
            * Predict class for features vector
            */
            int predict(float *x) {
                uint8_t votes[2] = { 0 };
                // tree #1
                if (x[0] <= 0.5) {
                    if (x[4] <= 3.5) {
                        if (x[2] <= 4.5) {
                            if (x[2] <= 2.5) {
                                if (x[1] <= 0.5) {
                                    votes[0] += 1;
                                }

                                else {
                                    votes[0] += 1;
                                }
                            }
```

✓ **Step2:** The shown .h RF model was exported using our SRAM optimized approach

✓ It contains a function named *predict* (red box) that links RF model with the main IoT application

- We create a .h dataset file that supplies data during the onboard inference



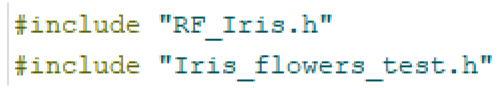
```
#pragma once
#define FEATURES_DIM 4
#define DATASET_SIZE 100

float X[DATASET_SIZE][FEATURES_DIM] = {
          {   6.3  ,  2.5  ,  4.9  ,  1.5  },
          {   5.0  ,  3.4  ,  1.5  ,  0.2  },
          {   5.4  ,  3.7  ,  1.5  ,  0.2  },
          {   5.7  ,  4.4  ,  1.5  ,  0.4  },
          {   6.6  ,  3.0  ,  4.4  ,  1.4  },

int y[DATASET_SIZE] = {   1  ,  0  ,  0  ,  0  ,  1
```

✓ The Iris Flowers dataset is stored in this .h file

✓ All dataset rows with its labels are written in this file

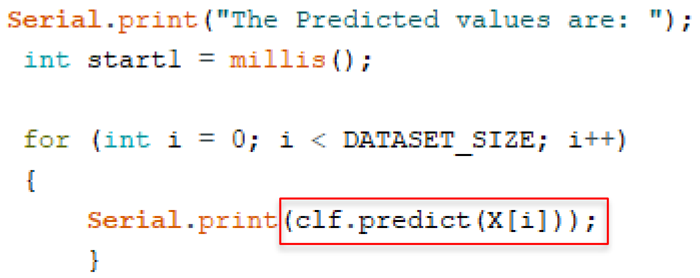
Executing ported classifiers on MCUs

- **Step3**: We first load .h model and .h dataset file into the IoT use-case application



```
#include "RF_Iris.h"
#include "Iris_flowers_test.h"
```

- **Step4**: We then pass data to *predict* function inside the .h model file to obtain inference results

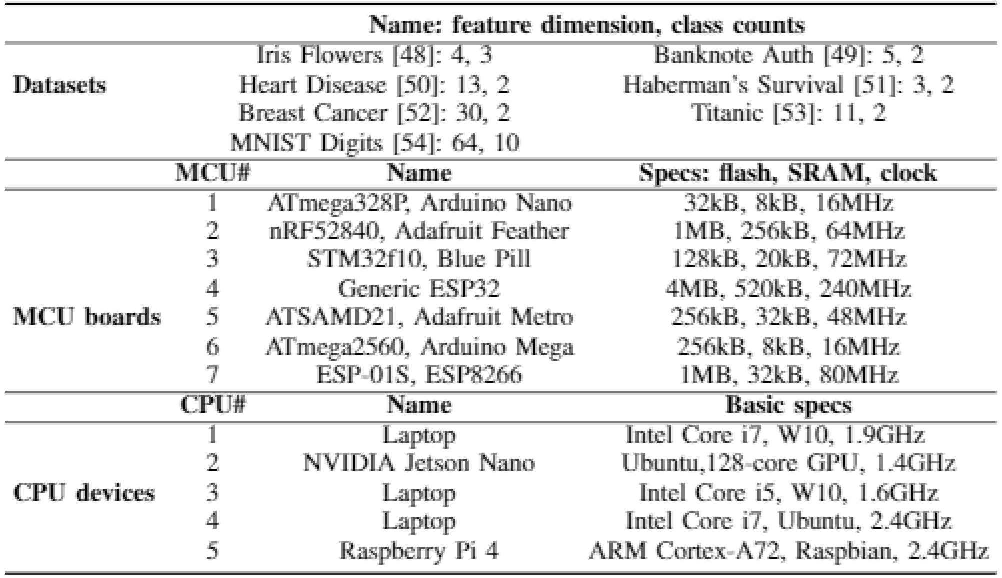


```
Serial.print("The Predicted values are: ");
 int start1 = millis();

 for (int i = 0; i < DATASET_SIZE; i++)
 {
     Serial.print(clf.predict(X[i]));
     }
```

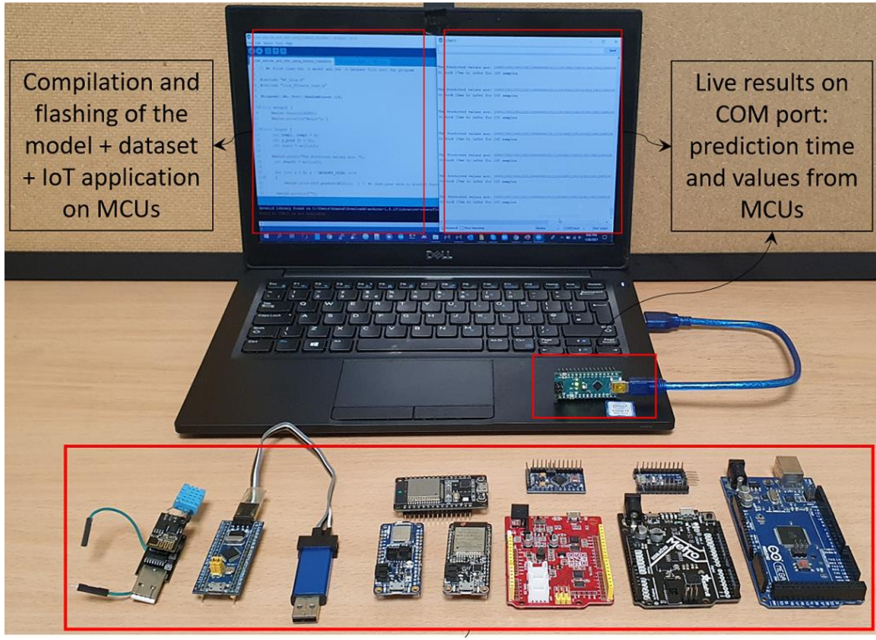
Experiment Setup

- 7 popular MCU boards and 5 CPU devices. 7 datasets of varying feature dimensions and class counts

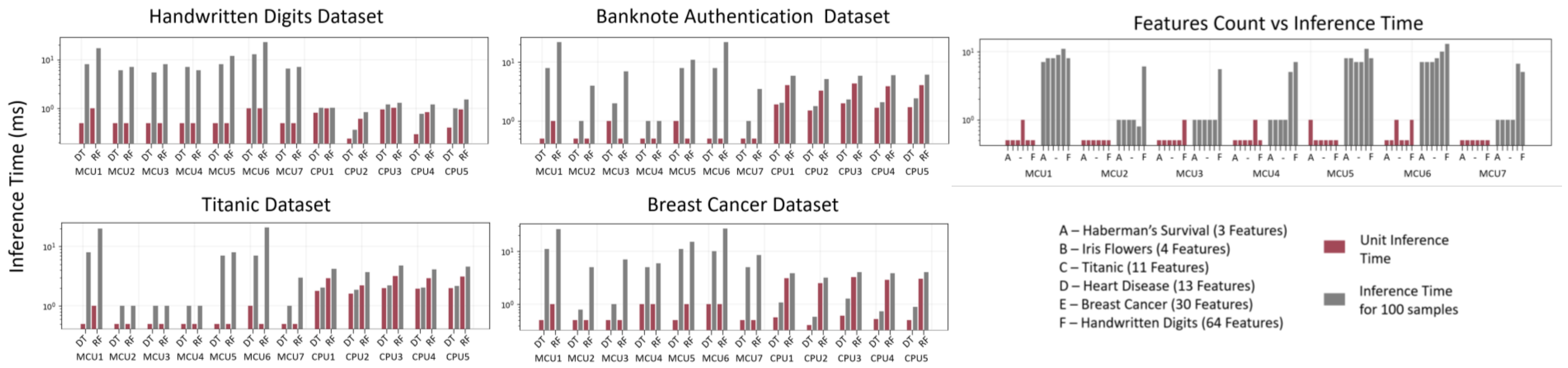
TABLE I
DATASETS, HARDWARE CHOSEN TO EVALUATE THE SRAM OPTIMIZED APPROACH.



		Name: feature dimension, class counts	
Datasets		Iris Flowers [48]: 4, 3	Banknote Auth [49]: 5, 2
		Heart Disease [50]: 13, 2	Haberman's Survival [51]: 3, 2
		Breast Cancer [52]: 30, 2	Titanic [53]: 11, 2
		MNIST Digits [54]: 64, 10	
	MCU#	**Name**	**Specs: flash, SRAM, clock**
MCU boards	1	ATmega328P, Arduino Nano	32kB, 8kB, 16MHz
	2	nRF52840, Adafruit Feather	1MB, 256kB, 64MHz
	3	STM32f10, Blue Pill	128kB, 20kB, 72MHz
	4	Generic ESP32	4MB, 520kB, 240MHz
	5	ATSAMD21, Adafruit Metro	256kB, 32kB, 48MHz
	6	ATmega2560, Arduino Mega	256kB, 8kB, 16MHz
	7	ESP-01S, ESP8266	1MB, 32kB, 80MHz
	CPU#	**Name**	**Basic specs**
CPU devices	1	Laptop	Intel Core i7, W10, 1.9GHz
	2	NVIDIA Jetson Nano	Ubuntu,128-core GPU, 1.4GHz
	3	Laptop	Intel Core i5, W10, 1.6GHz
	4	Laptop	Intel Core i7, Ubuntu, 2.4GHz
	5	Raspberry Pi 4	ARM Cortex-A72, Raspbian, 2.4GHz

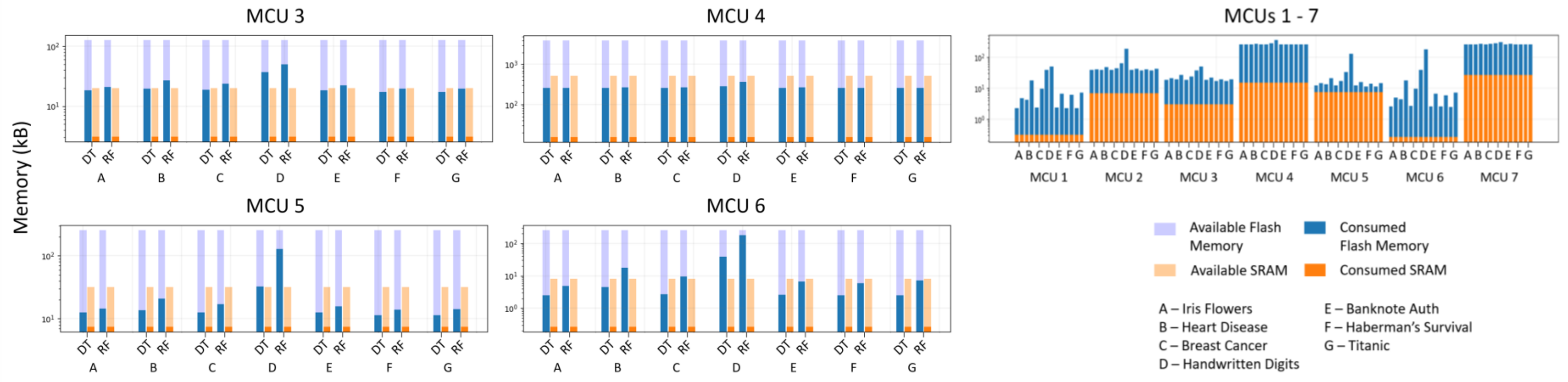

Compilation and flashing of the model + dataset + IoT application on MCUs
Live results on COM port: prediction time and values from MCUs

Results - Inference Performance



- Time taken by each MCU to perform unit inference and inference for 100 samples for each of the 14 models
  - ✓ All the MCUs performed unit inference in < 1 ms
  - ✓ 8-bit Atmega328P MCU1 performed faster unit inference than Jetson Nano and Raspberry Pi 4
  - ✓ ESP32 (MCU 4) inferred for 100 samples in 7 ms, only 5 ms slower than 200 times more costly CPUs 3 & 4
  - ✓ MCUs 1,5 & 6 show logarithmic growth. Fast MCUs 2, 3, 4 & 7 show time growth only for Cancer and Digits

Results - SRAM and Flash usage



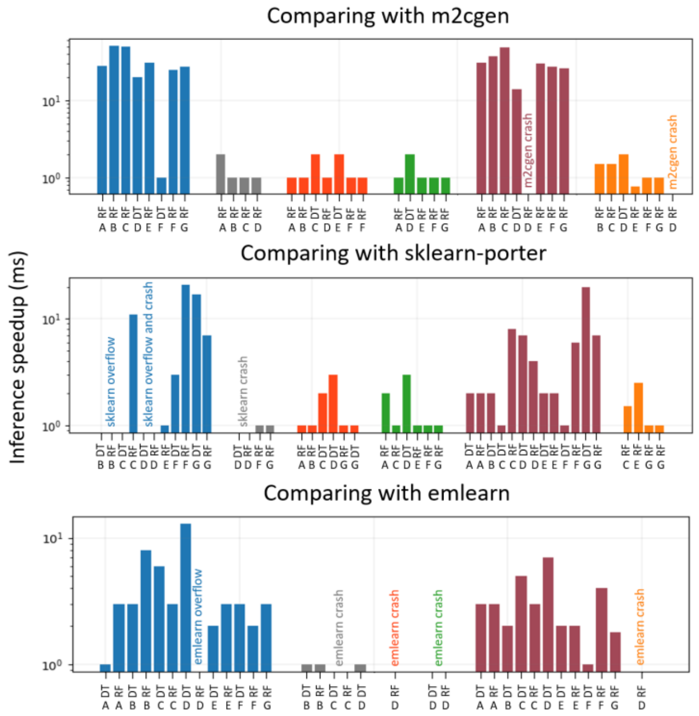
- Memory consumed by MCUs to execute various datasets trained DT and RF classifiers

  ✓ MCU 7 has 31 times more Flash than available 8 kB SRAM. Our approach do not store model-related parameters in variables, so the ported models do not consume SRAM. Instead, it sacrifices larger Flash

  ✓ RF classifier trained using Digits dataset (64 features and 10 classes) has the largest model size after porting (so it occupies more Flash memory). But it consumes the same amount of SRAM as other classifiers produced by training using smaller datasets like Iris Flowers (4 features and 3 classes)

Results Comparison

Confirm
Smart Manufacturing



A – Iris Flowers B – Heart Disease C – Breast Cancer D – Handwritten Digits E – Banknote Auth F – Haberman's Survival G – Titanic

🟦 MCU1 ⬜ MCU2 🟥 MCU3 🟩 MCU4 🟥 MCU6 🟧 MCU7

- Inference speedups achieved in comparison with popular m2cgen, sklearn-porter, emlearn libraries
  - ✓ 1-4x times faster inference
  - ✓ More resource-constrained MCUs 1 & 7 achieved higher speedups than others
  - ✓ No SRAM overflows and crashes
  - ✓ Eliminated debugging and code fine tuning steps
  - ✓ No additional files (eml_trees.h) dependency

Summary

Confirm Smart Manufacturing

- SRAM optimized approach: End-to-end **porting, stitching and execution** of ML classifier models on MCU based resource-constrained IoT devices
  - ✓ **Resource-friendly**: Consumes 0 bytes of SRAM during onboard classifier execution
  - ✓ **Compatibility:** Applicable to variety of DT and RF classifiers, MCUs and small CPUs
  - ✓ **Ultra-fast classifications:** 1-4 x times faster than state-of-the-art libraries
  - ✓ **Performance guarantee**: Same support, macro avg, F1 score, etc. as original models before porting

Confirm
Smart Manufacturing

Confirm
Smart Manufacturing

Contact: Bharath Sudharsan
Email: bharath.sudharsan@insight-centre.org

www.confirm.ie

Science Foundation Ireland For what's next