Confirm
Smart Manufacturing

Confirm
Smart Manufacturing

On-Device Learning, Optimization, Efficient Deployment and Execution of ML Algorithms on Resource-Constrained IoT Hardware

Bharath Sudharsan

A World Leading SFI Research Centre

Science Foundation Ireland sfi For what's next

University of Limerick, Ollscoil Luimnigh
Tyndall National Institute
UCC University College Cork, Ireland, Coláiste na hOllscoile Corcaigh
NUI Maynooth, Ollscoil na hÉireann Má Nuad
OÉ Gaillimh NUI Galway
CIT Cork Institute of Technology, Institiúid Teicneolaíochta Chorcaí
AIT
LIT

Overview - Edge Computing

Confirm
Smart Manufacturing

Missing Child

Icy Sidewalk

Spilled Liquid

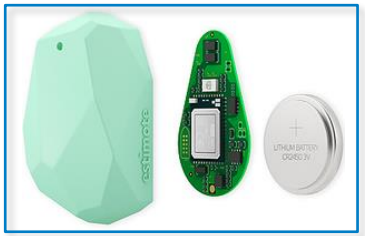
Ignored Display

Long Line

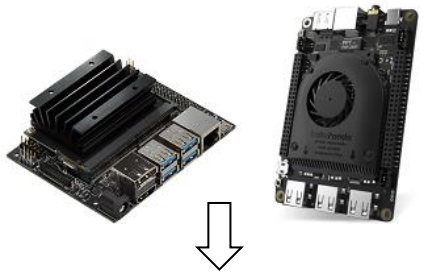
Edge computing example applications

- Push apps and computing power to the internet edge
- Manage apps outside data centers - on customer devices

Cortex-M0 MCU BLE beacon



CPU + basic GPU based SBCs

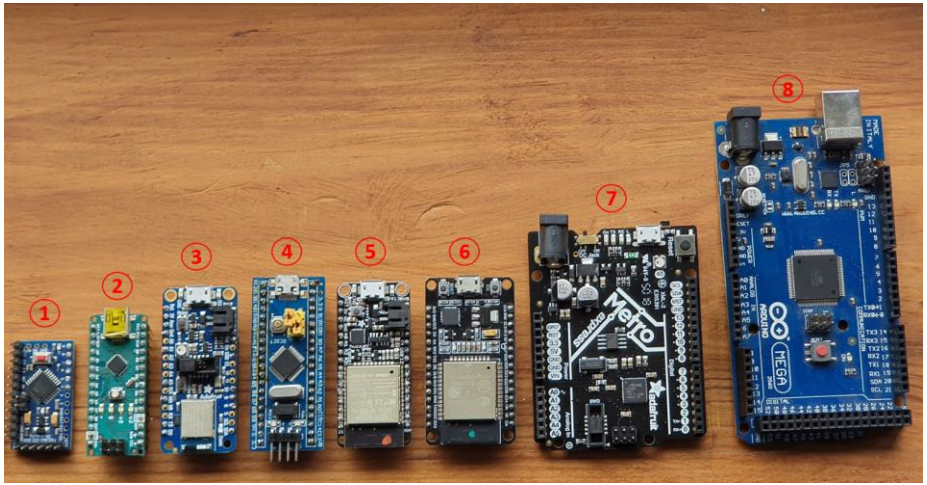


Edge gateway with GPUs and SSDs

- 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?** Roughly 50 billion chips shipped in 2020 (market estimates). GPUs & CPUs only 100 million units sold

Overview - Edge Computing Hardware

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 IoT development boards with hardware specifications

- No file system
- No parallel execution units
- Low SRAM and Flash memory
- Lack Python support. Only C, C++
- Only few hundred MHz clock speed, etc.

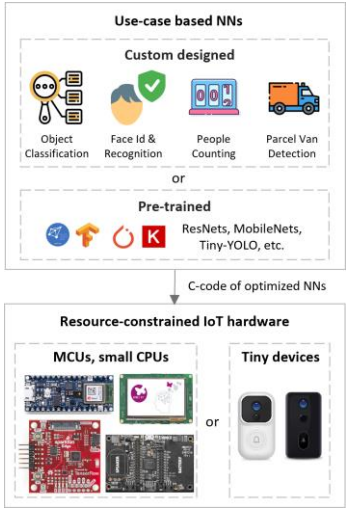
- Lack inbuilt hardware accelerators such as
  - ✓ APU (accelerated processing unit)
  - ✓ KPU (convolution operation accelerator)
  - ✓ FPU (floating-point accelerator)
  - ✓ FFT (fourier transform accelerator)

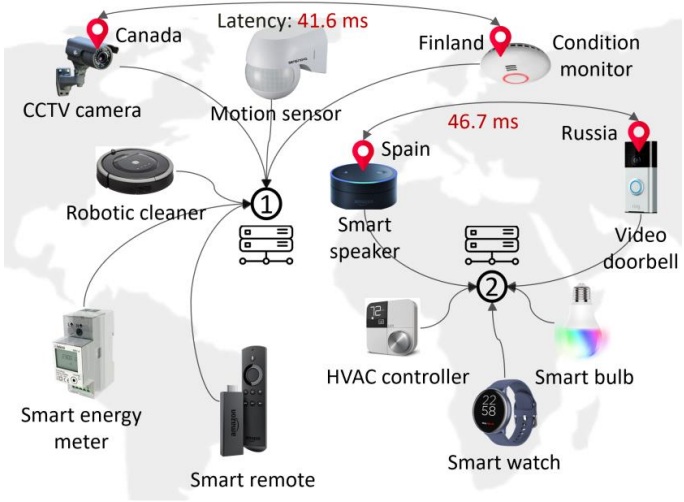
Overview - Thesis Contributions

Confirm
Smart Manufacturing

Use case: Learn to monitor vibrations on pump's crosshead, cylinder and frame. Generate alerts if anomaly patterns are detected

Use case: Learn to detect Leaks, monitor efficiency & consumptions





On-Device Learning

- Research Objective I
  - ✓ Paper I - Train++ Algorithm
  - ✓ Paper II - ML-MCU Framework
- Research Objective II
  - ✓ Paper III - Imbal-OL Plugin

Efficient ML on IoT Devices

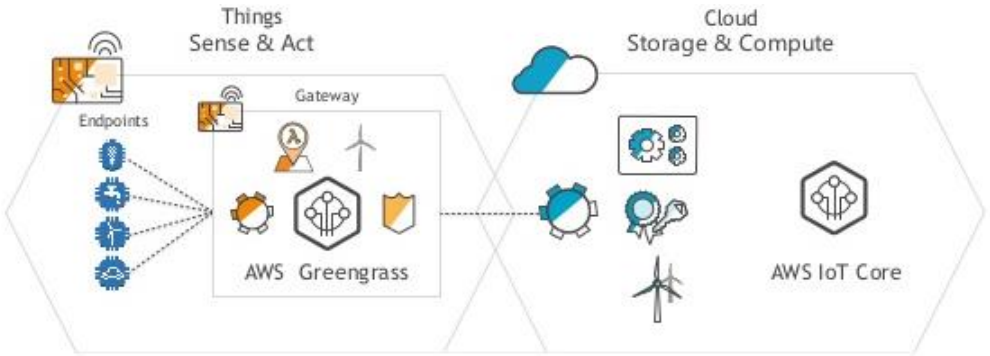
- Research Objective III
  - ✓ Paper IV - SRAM-NNs Approach
- Research Objective IV
  - ✓ Paper V - SRAM-Non-NNs Approach

Distributed ML using IoT Devices

- Research Objective V
  - ✓ Paper VI - ElastiQuant Technique
- Research Objective VI
  - ✓ Paper VII - Globe2Train Framework

On-Device Learning - Introduction

Confirm
Smart Manufacturing

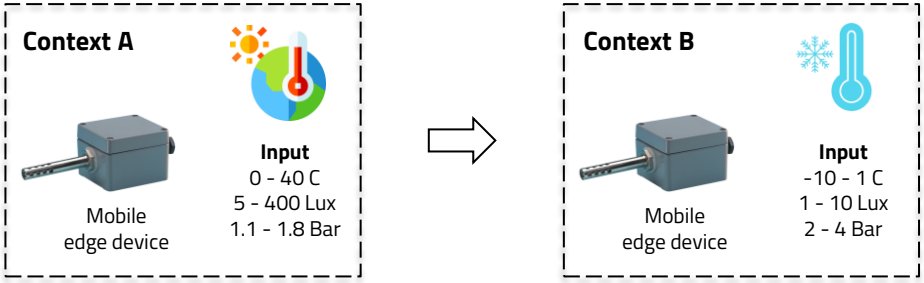


Edge devices depends on cloud for model updates

- To improve inference accuracy, edge devices log unseen data in cloud -> initial model re-trained -> sent as OTA updates
  - ✓ Devices are not self-contained
  - ✓ Hardware cost increases - wireless module addition
  - ✓ Increases cyber-security risks and power consumption
  - ✓ Latency, privacy, other concerns

On-Device Learning - Challenge I

Confirm
Smart Manufacturing



Accuracy: 92 %

Accuracy: 54 %

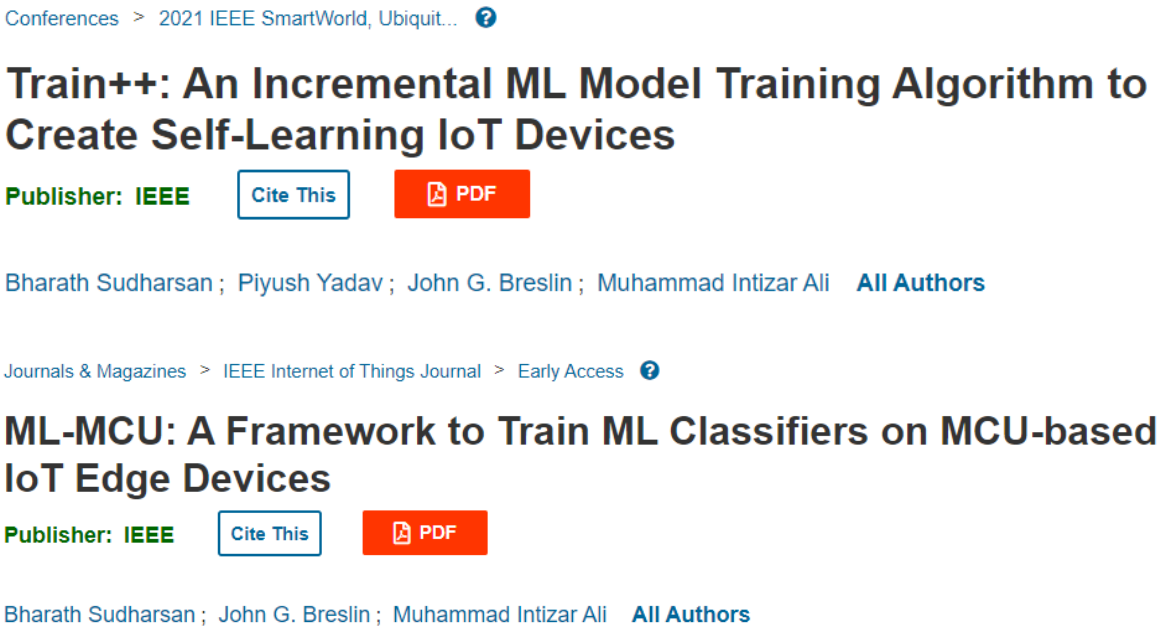
Edge devices deployed in various environments for inference

- In real-world every context can be unique by generating fresh/unseen data patterns
- Sometimes deployed models don't know how to react - can lead to false or less accurate results
- Models cannot produce expected results when trained for one context and deployed in another
- Not feasible to train multiple models for multiple contexts

On-Device Learning - Objective I

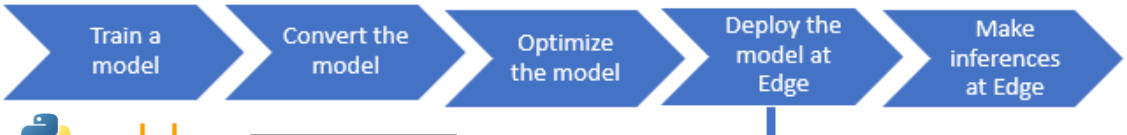
Confirm
Smart Manufacturing

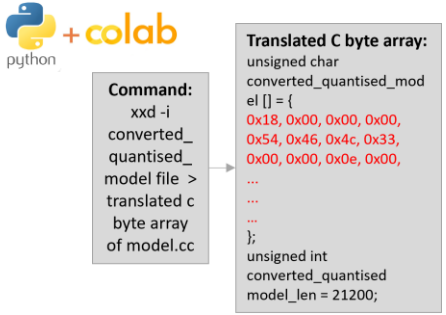
- **Research Objective I.** Design resource-friendly algorithms to enable training ML models directly on MCUs & small CPUs
- **Paper I** and **Paper II** fulfill Research Objective I

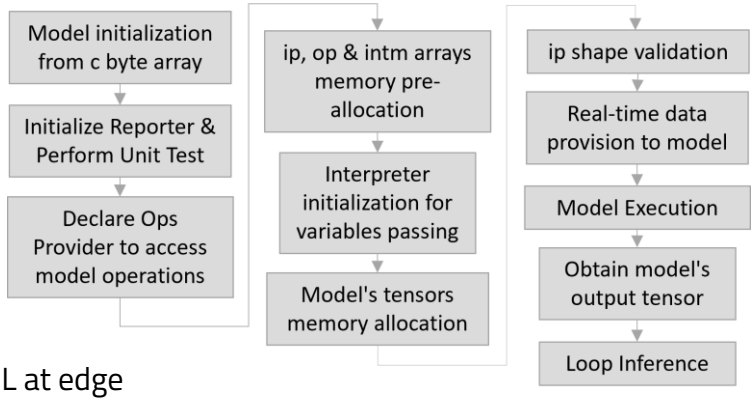


On-Device Learning - SOTA

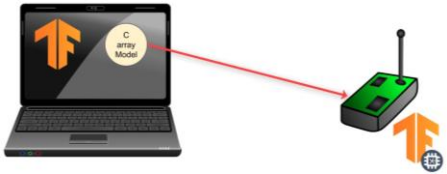
Confirm
Smart Manufacturing

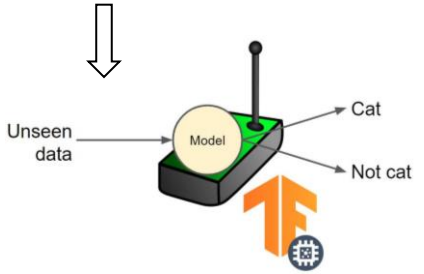






Standard workflow for ML at edge

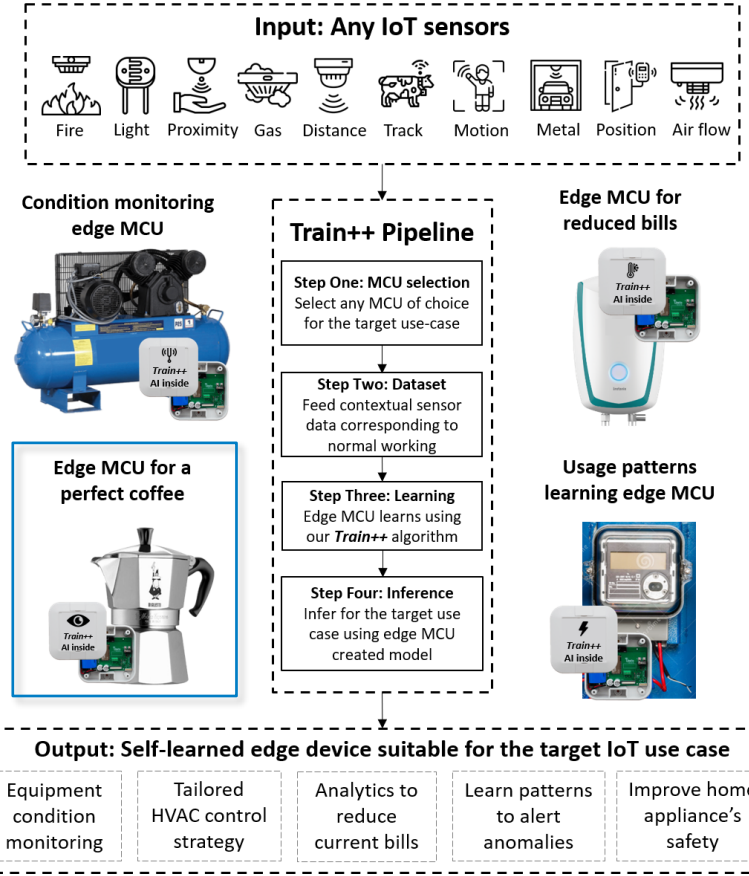




- Single Board Computers (SBCs) - well studied area
  - ✓ Numerous papers, libraries, algorithms, tools
  - ✓ TF Lite, FeatherCNN, Open-NN, Caffe, MXNet, PyTorch, Darknet, XNOR, etc.
- MCUs & small CPUs
  - Efficient Inference - well studied area
    - ✓ TF Micro, Bonsai, EMI-RNN, FastCells, Proto-NN, SeeDot, ArmNN, MCUNet, etc.
  - On-device learning - **emerging area**
    - ✓ Few papers - TinyFedTL, TinyOL, SEFR, ML in embedded sensor systems

On-Device Learning - Train++

Confirm
Smart Manufacturing



- Four-steps to create self-learning devices using **Train++ pipeline**
  - ✓ Select MCU or small CPU of choice depending on use case
  - ✓ Contextual sensor data corresponding to normal working will be collected as the local dataset and used for training
  - ✓ Devices learns/trains using our **core Train++ algorithm**
  - ✓ Resultant models can start inference over unseen data

On-Device Learning - Train++

Confirm
Smart Manufacturing

Algorithm 1 Train++: A high-performance binary classifier training algorithm for MCU-based IoT Devices.

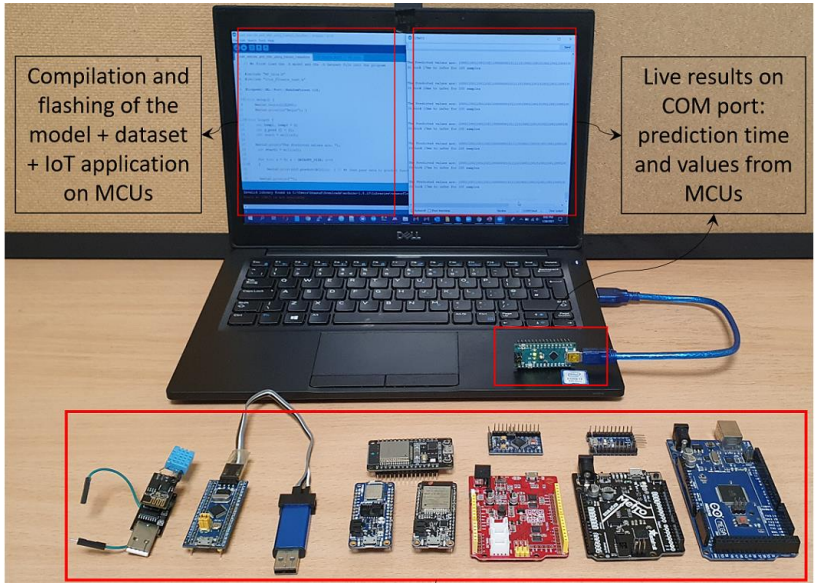
Inputs:
C: Positive parameter to control the influence of ξ.
t: The dimension of time for real-time data inputs.
x_t: Real-time sensor data inputs. Where $x_t \in \mathbb{R}^n$.
y_t: Correct labels. Where $y_i \in \{-1, +1\}$.
Output:
w_t: Incrementally learned weights.
Receive live data stream, represented as in Eqn (1).
function MCUTrain (x_t, y_t, t, C)
for t = 0 to setsize **do**
Predict $\hat{y}_t$ for every x_t. Use $sign(x_t.w_t)$.
Compute confidence score of prediction. Use $|x_t.w_t|$.
Compute margin at t. Use $y_t(x_t.w_t)$.
Show original label y_t to this algorithm.
if ($\hat{y}_t$ equals to y_t) **then**
Prediction was correct. $y_t = sign(x_t.y_t)$.
Predict again with a higher confidence score.
if ($sign(x_t.y_t)$ lesser than 1) **then**
Suffer an instantaneous loss. Use Eqn (2).
if ($sign(x_t.y_t)$ exceeds 1) **then**
Loss becomes 0.
else
Wrong prediction. Loss becomes 1 - $y_t(x_t.w_t)$.
end if
Incrementally learn w_t. Use Eqn (4).
Store the learned weights w_t.
end for

- Core Train++ algorithm implementation: https://github.com/bharathsudharsan/Train_plus_plus

  - ✓ **Dataset Size**. Incremental characteristic to train on limited SRAM while allowing to use full *n-samples* of *high-dimensional* datasets
  - ✓ **Improved Privacy.** Keeps reading live data -> incrementally learns from it -> discards data (no storing required)
  - ✓ **Light-weight.** <100 lines in C++. Does not depend on third-party libraries. Executable by 8-bit MCUs with no FPU, KPU, APU

On-Device Learning - Train++

Confirm
Smart Manufacturing

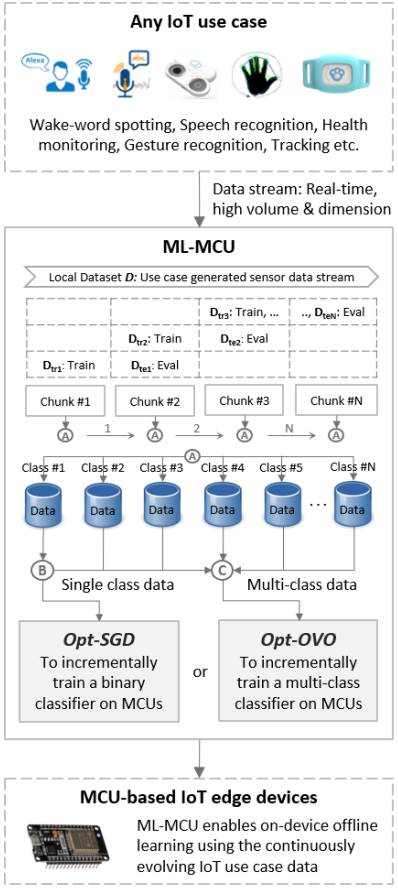


Experiment setup: ML model training and inference on MCUs using Train++

- Train++ evaluation result highlights:
  - ✓ Real-time inference on MCUs in of 1 ms
  - ✓ Reduced onboard training time by ≈10-226 sec
  - ✓ Evaluation accuracy improved by 5.15 - 7.3 %
  - ✓ Compatible with various IoT boards and datasets
  - ✓ Consumed ≈34000 - 65000x times less energy to train
  - ✓ Consumed ≈34 - 66x times less energy to infer

On-Device Learning - ML-MCU

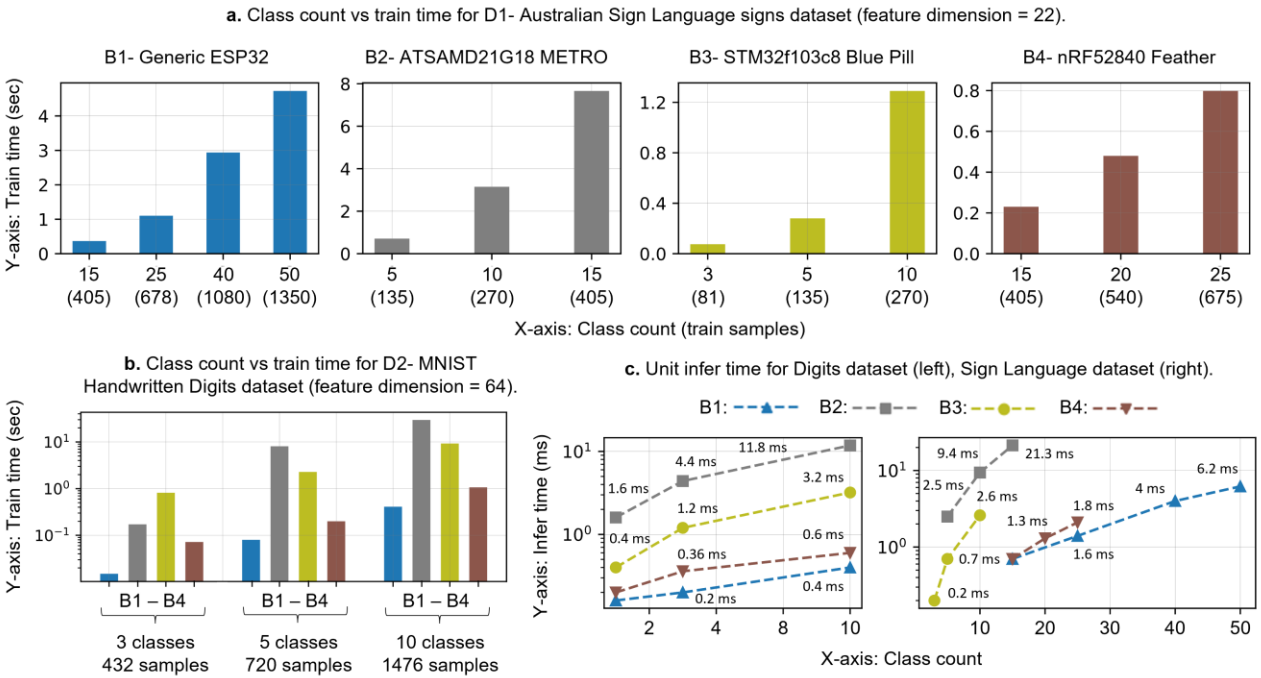
Confirm
Smart Manufacturing



- **Optimized-Stochastic Gradient Descent (Opt-SGD)**
  - ✓ For high performance, resource-friendly binary classifiers training on MCUs
  - ✓ Combines benefits from both Gradient Descent (GD) + Stochastic Gradient Descent (SGD)
  - ✓ Thus, inherits the stability of GD while retaining the work-efficiency of SGD
- **Optimized One-Versus-One (Opt-OVO)**
  - ✓ First novel algorithm to enable multi-class classifiers training on MCUs
  - ✓ Decomposes one multi-class problem into multiple binary problems (base learners)
  - ✓ Opt-OVO archives reduced computation by identifying and removing base classifiers that lack significant contributions to the overall multi-class classification result
  - ✓ For further efficiency, we use Opt-SGD inside Opt-OVO for training base learners

On-Device Learning - ML-MCU

Confirm
Smart Manufacturing



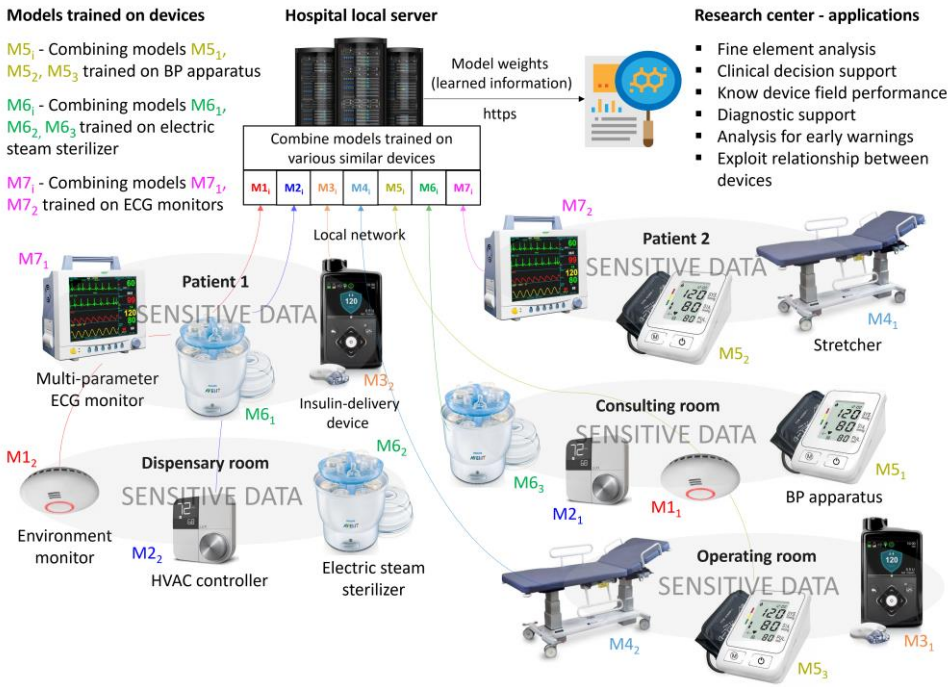
ML-MCU sets the state-of-the-art

- ✓ Unit inference in <1 sec across B1-B4
- ✓ Real-time unit inference in 11.8 ms even on the slowest B2
- ✓ B1 - Generic ESP32 is just 3$
- ✓ B1 trained in 4.7 sec for D1 (50 class)
- ✓ B1 inferred for D1 (50-class) in 6.2 ms

ML model training and inference on MCU boards using Opt-OVO
Implementation: https://github.com/bharathsudharsan/ML-MCU

On-Device Learning - Use Case

Confirm
Smart Manufacturing



Collective learning among medical devices - insulin-delivery devices, electric steam sterilizers, BP apparatus, etc.

- **Use-case: Privacy-preserving medical data collection**
  - ✓ Data required for research can be sensitive in nature
  - ✓ Medical data can revolve around private individuals
- On-device learning using Train++, ML-MCU, or others
  - ✓ Resource-constrained medical devices can train offline using sensitive data
  - ✓ Transmit models instead of the actual medical data

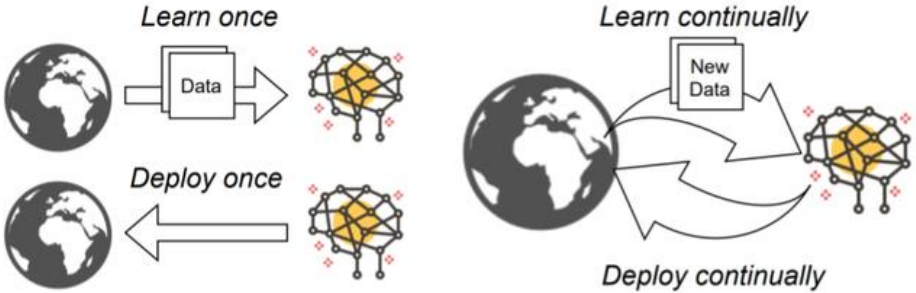
Paper. Ensemble Methods for Collective Intelligence: Combining Ubiquitous ML Models in IoT
Bharath Sudharsan and others
Published in 2022, IEEE International Conference on Big Data

On-Device Learning - Challenge II

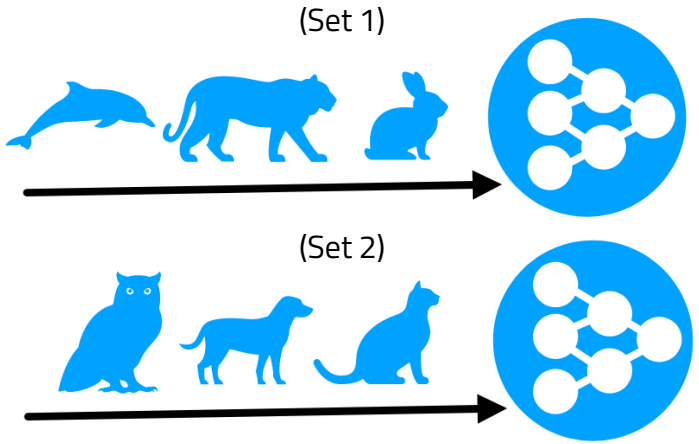
Confirm
Smart Manufacturing

Fixed C source file of ML model - flexibility impeded, no adaptation

Continuously updating last few NN layers - adapt to dynamic scenarios



On-device learning (OL) using NNs



Catastrophic forgetting example

- NNs are known to catastrophically forget the information that are not frequently or recently seen
  - ✓ Train a NN to recognize dolphin, rabbit and lion (Set 1). Then train it with bird, cat, dog classes (Set 2)
  - ✓ After some time, the NN can start getting poor results in recognizing any class in Set 1
  - ✓ **Why?** Because the weights are biased to recognize the classes in Set 2

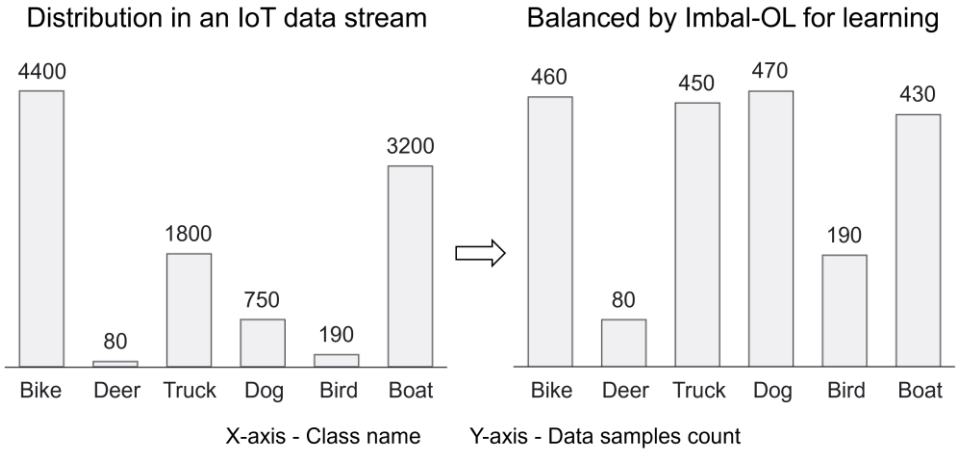
On-Device Learning - Objective II

Confirm
Smart Manufacturing

- **Research Objective II.** To perform OL in non-ideal real-world settings, an OL plugin need to be designed, that understands the supplied data stream and balances the class size before sending it for learning
- **Paper III** fulfill Research Objective II

On-Device Learning - Imbal-OL

Confirm
Smart Manufacturing



ImbalOL plugin illustration

- Without discarding samples from significantly underrepresented deer and bird classes, Imbal-OL balances remaining classes
- Using Imbal-OL processed stream for OL, models can adapt faster to changes in stream while parallelly preventing catastrophic forgetting

- Sensor data streams in real-world IoT
  - ✓ Severely imbalanced + temporally correlated
  - ✓ Non-stationary data distribution
  - ✓ Incomplete
- Imbal-OL plugin characteristics:
  - ✓ **Underrepresented Classes.** Identifies and stores all data representing minority classes
  - ✓ **Memory Utilization.** To avoid inefficiency, it fills entire memory, then balances classes
  - ✓ **Weighted Replay.** Replays samples considering their class rather than uniform (similar prob.)

On-Device Learning - Imbal-OL

Confirm
Smart Manufacturing

TABLE I
ACCURACY (%) COMPARISON OF MODELS AFTER OL USING CIFAR-10.

Scheme	m = 100		m = 500	
	Uniform	Weighted	Uniform	Weighted
GSS-greedy	63.6	65.2	62.2	66.8
Imbal-OL	69.4	70.7	70.2	72.6

TABLE II
TIME, MEMORY CONSUMED BY SCHEMES TO MAKE STREAMS OL READY.

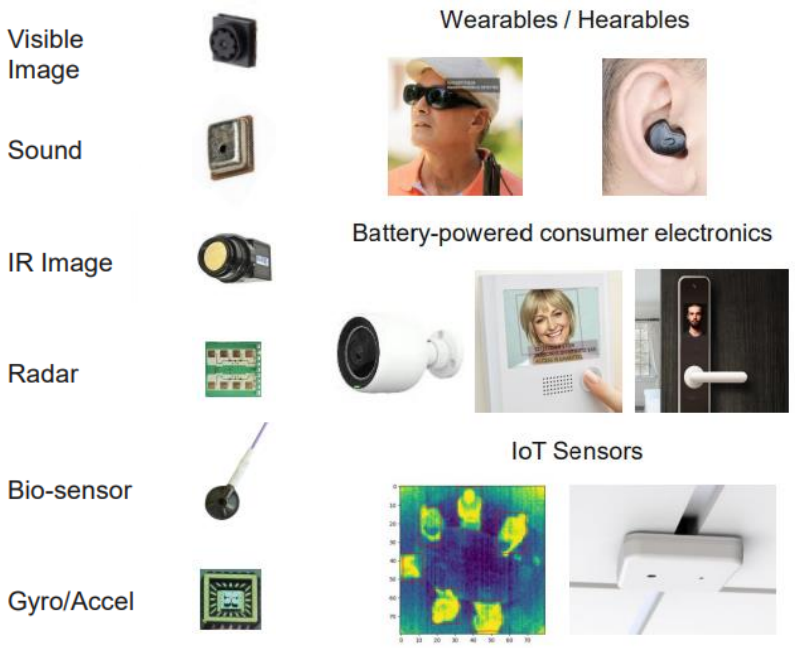
Scheme	Time (sec)		Memory (%)	
	CIFAR-10	CIFAR-100	CIFAR-10	CIFAR-100
Random-replace	4.17	4.9	3.7	3.4
GSS-greedy	24.3	23.6	31.8	32.6
Imbal-OL	8.18	9.06	4.6	5.1

- Uniform replay
  - ✓ Higher performance gap
  - ✓ For GSS-greedy, models get **influenced** by imbalances
  - ✓ So models show top performance only for few classes
- Weighted replay
  - ✓ Partially **masks** effect of imbalanced streams
  - ✓ Oversampling minority classes

- Time
  - ✓ Random-replace, Imbal-OL use roughly same time
  - ✓ GSS-greedy non-suitable for tiny devices
- Memory Consumption
  - ✓ To speed up the storage process after sampling
  - ✓ Imbal-OL store the **memory address** of the classes
  - ✓ So uses more memory than Random-replace

Efficient ML on IoT Devices - Introduction

Confirm
Smart Manufacturing



tinyMLPerf **Working Group Members**

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

TinyML practitioners - Executing ML on their MCU-based products

- Popular applications
  - ✓ Audio - always-on inference
  - ✓ Industry telemetry - detect bearing vibration anomalies
  - ✓ Image - object counting, visual wake words
  - ✓ Behavior - activity recognition using IMU or EMG

- Popular implementations
  - ✓ NanoEdge AI: https://cartesiam.ai/
  - ✓ EdgeML: https://microsoft.github.io/EdgeML/
  - ✓ **Model Zoo:** https://github.com/ARM-software/ML-zoo
  - ✓ TinyMLPerf: https://github.com/mlcommons/tiny

Efficient ML on IoT Devices - Challenge III

Confirm
Smart Manufacturing

```
Sketch uses 1395454 bytes (106%) of program storage space. Maximum is 1310720 bytes.
text section exceeds available space in board
Global variables use 64484 bytes (19%) of dynamic memory, leaving 263196 bytes for l
Sketch too big; see http://www.arduino.cc/en/Guide/Troubleshooting#size for tips on
Error compiling for board ESP32 Dev Module.
```

Flash overflow example

```
data section exceeds available space in boardSketch uses 28960 bytes (89%) of program storage space. Maximum is 32256 bytes.

Global variables use 2566 bytes (125%) of dynamic memory, leaving -518 bytes for local variables. Maximum is 2048 bytes.
Not enough memory; see http://www.arduino.cc/en/Guide/Troubleshooting#size for tips on reducing your footprint.
Error compiling for board Arduino Uno.
```

SRAM overflow example

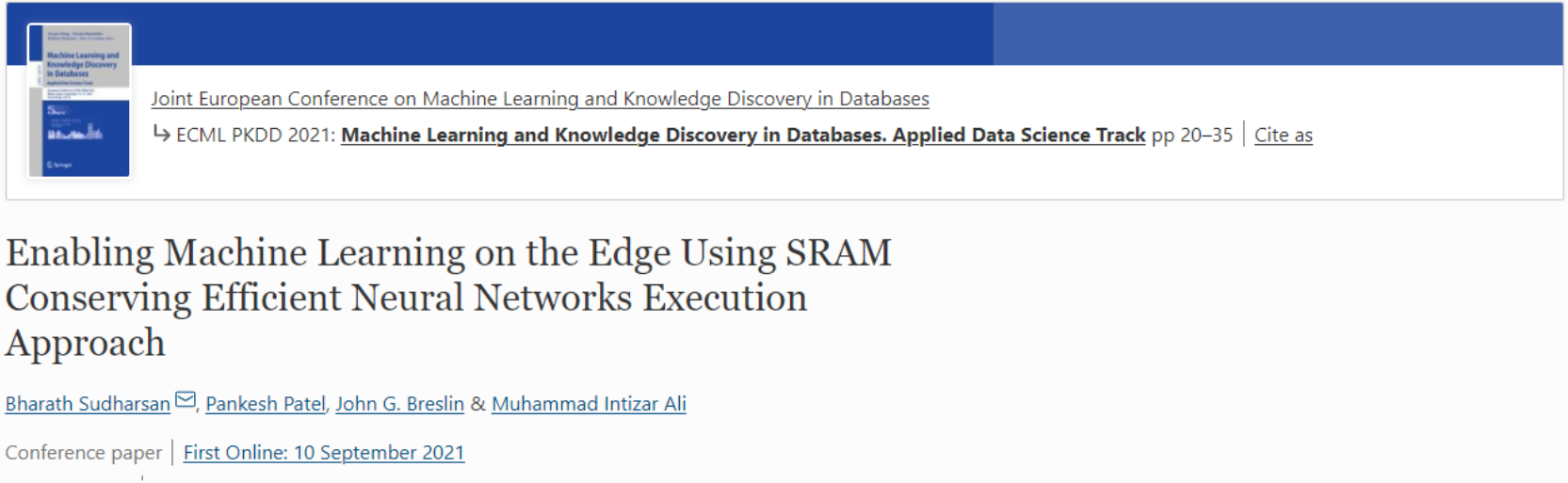
- Deploying maximum deep compressed NNs
  - ✓ NNs can exceed memory by just few bytes
  - ✓ Already fully compressed
  - ✓ No optimization compatibility
- Require NAS and re-training or
- Hardware upgrade to accommodate model

- After deep optimization
- If running at peak memory
  - ✓ Overheating
  - ✓ Fast battery wear
  - ✓ Run-time stalling
  - ✓ Low service life

Efficient ML on IoT Devices - Objective III

Confirm
Smart Manufacturing

- **Research Objective III.** How to reduce maximum optimized NNs peak SRAM so to fit and comfortably execute on tiny devices?
- Paper IV fulfills Research Objective III



Joint European Conference on Machine Learning and Knowledge Discovery in Databases

↳ ECML PKDD 2021: **Machine Learning and Knowledge Discovery in Databases. Applied Data Science Track** pp 20–35 | Cite as

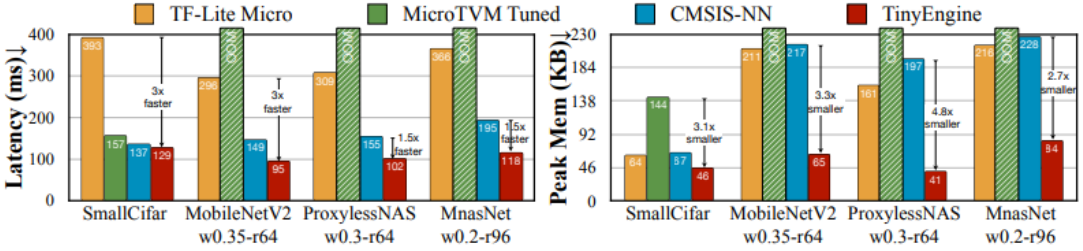
Enabling Machine Learning on the Edge Using SRAM Conserving Efficient Neural Networks Execution Approach

Bharath Sudharsan, Pankesh Patel, John G. Breslin & Muhammad Intizar Ali

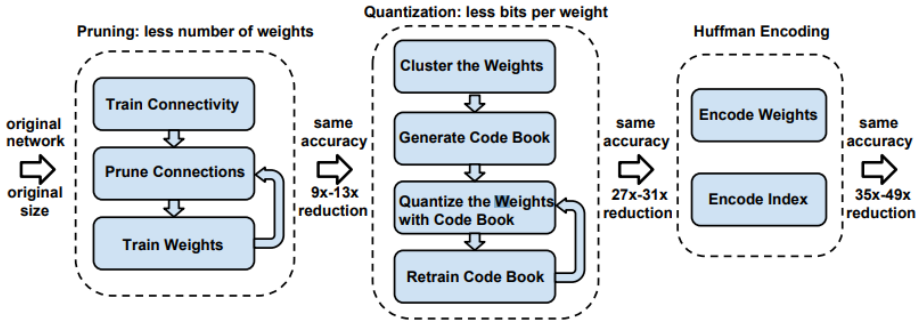
Conference paper | First Online: 10 September 2021

Efficient ML on IoT Devices - SOTA

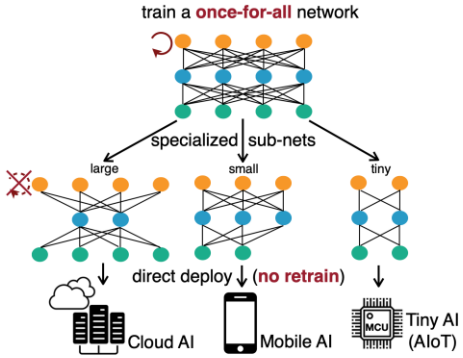
Confirm
Smart Manufacturing

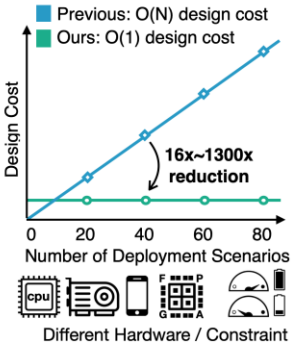


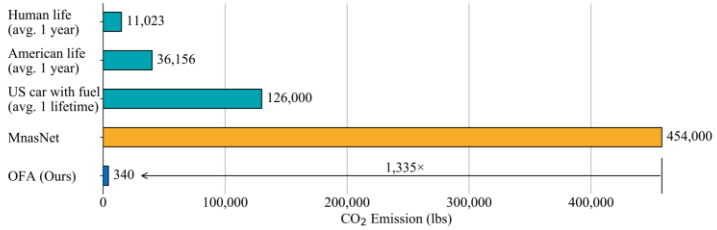
- ✓ TinyEngine 3× faster than TF-Lite Micro (Google),1.6× than CMSIS-NN (ARM)
- ✓ By reducing memory, it enlarges design space for TinyNAS for MCUs
- ✓ Specialize each optimization, adopt in-place depth-wise convolution



- ✓ Compression pipeline paper - Pruning + Quantization + Huffman coding



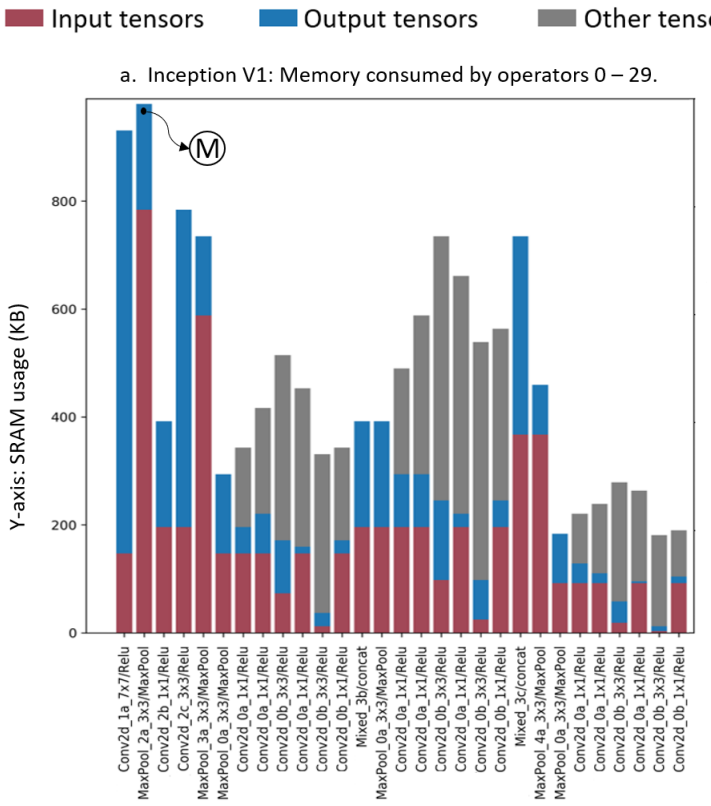




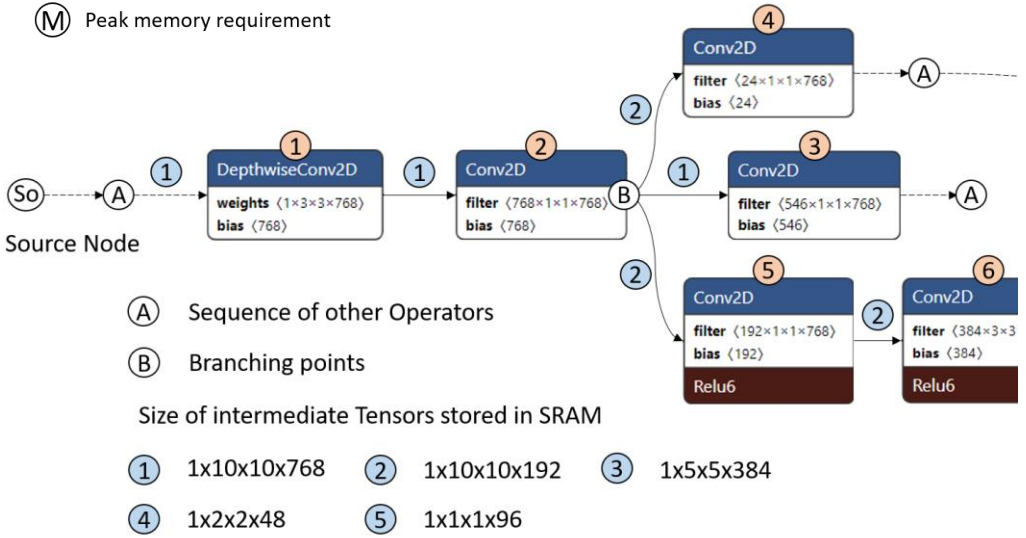
- ✓ Single network trained to support versatile devices
- ✓ Given deployment scenario, subnetwork is selected
- ✓ Cost of deep learning deployment from O(N) to O(1)

Efficient ML on IoT Devices - SRAM-NNs

Confirm
Smart Manufacturing



TMM for Inception V1 from TFHub

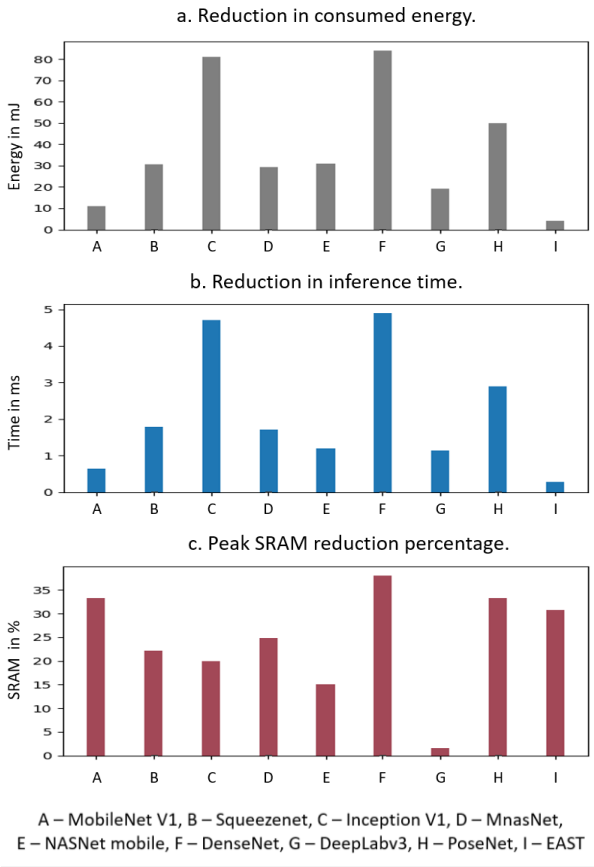


A part of the COCO SSD MobileNet graph with its branched operators

- **Proposed Approach.** Core algorithm implemented for experiments
  - ✓ Tensor Memory Mapping (TMM) method
  - ✓ Load fewer tensors and tensors re-usage
  - ✓ Cheapest NN graph execution sequence

Efficient ML on IoT Devices - SRAM-NNs

Confirm
Smart Manufacturing



- Comparing default sequences with optimized sequences
  - Peak SRAM reduction
    - ✓ Max of 38.06% for DenseNet (F)
    - ✓ Least of 1.61% for DeepLabv3 (G)
  - Inference time reduction
    - ✓ Max of 4.9 ms for DenseNet (F)
    - ✓ Least of 0.28 ms reduction for EAST (I)
  - Energy conservation
    - ✓ 4 - 84 mJ less energy for unit inference

Efficient ML on IoT Devices - Challenge IV

Confirm
Smart Manufacturing

- Increasing training samples and classes increases model size and inference complexity
- For DTs and RFs to keep a low memory footprint
  - ✓ Node parameters in the DTs are shared using a directed acyclic graph, Pruning, etc.
  - ✓ Multi-stage MCU-aware optimization (tailored) - ARM microTVM
  - ✓ Design sparse and shallow tree learners that fit in few kBs
- Such methods of learning shallow trees or aggressive pruning often led to degradation in accuracy
- Due to approximation of non-linear and complex decision boundaries using small number of axis-aligned hyperplanes

Efficient ML on IoT Devices - Objective IV

Confirm
Smart Manufacturing

- **Research Objective IV.** An SRAM optimized approach is needed to perform ML classifier porting, stitching, and efficient execution
- **Paper V** fulfills Research Objective IV

Conferences > 2021 IEEE International Conference on Services Computing (SCC)

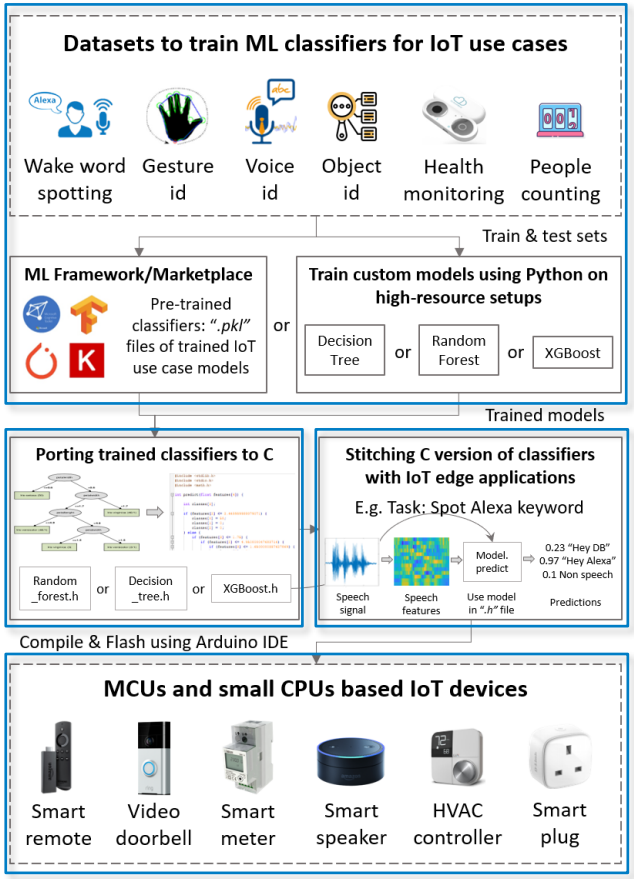
An SRAM Optimized Approach for Constant Memory Consumption and Ultra-fast Execution of ML Classifiers on TinyML Hardware

Publisher: IEEE Cite This PDF

Bharath Sudharsan ; Piyush Yadav ; John G. Breslin ; Muhammad Intizar Ali **All Authors**

Efficient ML on IoT Devices - SRAM-Non-NNs

Confirm Smart Manufacturing



- Generic 4-step end-to-end design flow
- Novelty - SRAM optimized porting approach
  - ✓ Produce C version DTs - do not depend (0 bytes) on SRAM during execution
  - ✓ **How?** by trading with the larger flash memory
  - ✓ Sacrifice flash memory in favor of SRAM since it is scarcest

- Exported RF **.h** model

```
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;
                                    }
                                }
```

- Load files into IoT app

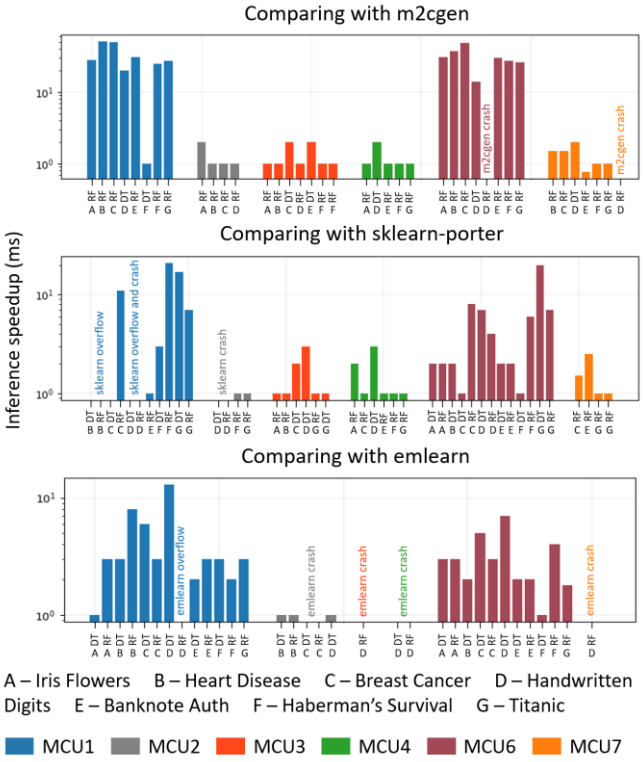
```
#include "RF_Iris.h"
#include "Iris_flowers_test.h"
```

- Pass data to *predict* fn of RF **.h** model

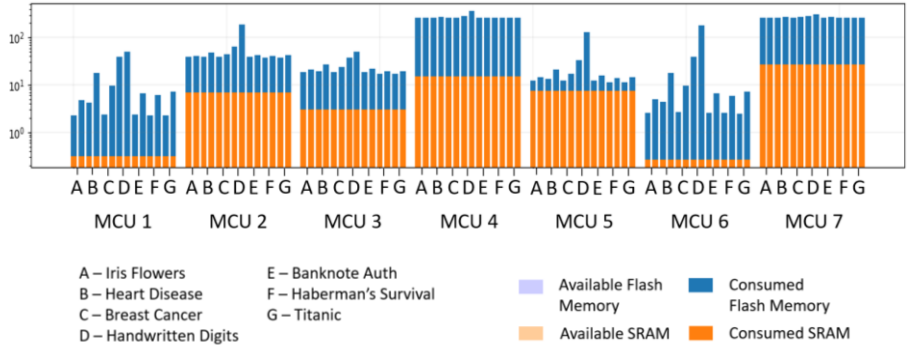
```
Serial.print("The Predicted values are: ");
 int startl = millis();

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

Efficient ML on IoT Devices - SRAM-Non-NNs

Confirm
Smart Manufacturing



Inference performance comparison

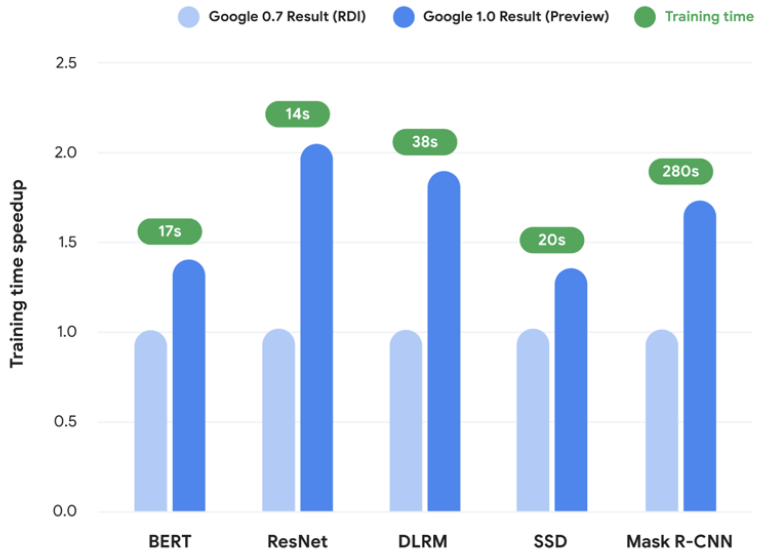


Porting and executing classifiers on MCUs using proposed approach

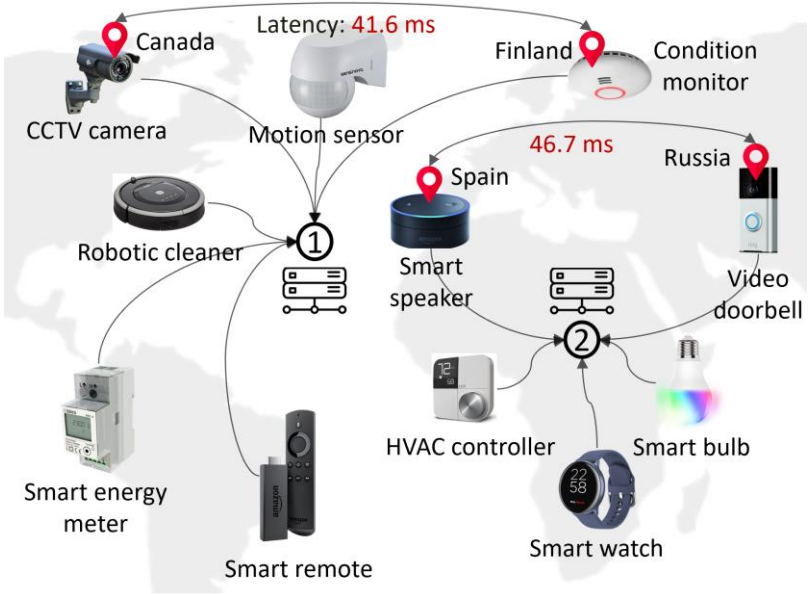
- Comparison with SOTA libraries
  - ✓ Higher speedups on most tiny MCUs
  - ✓ Eliminate debugging & code tuning
  - ✓ No files (eml_trees.h) dependency
  - ✓ No SRAM overflows and crashes
  - ✓ 1-4x times faster inference
- Memory usage
  - ✓ Digits (64 f x10 c) RF model
  - ✓ Largest Flash size after porting
  - ✓ But it consumes same SRAM
  - ✓ As of Iris Flowers (4 f x 3 c)

Distributed ML using IoT Devices - Introduction

Confirm
Smart Manufacturing



Training ML models on Google TPU Pod v4 - MLPerf



Training ML models on IoT devices - Globe2Train

- **Distributed ML in Data Centers/TPU Pods/GPU Clusters** - Train SOTA models faster, at greater scale, and at lower cost
- **Distributed ML in IoT** - On-device training with privacy preservation. Distribute neural workloads on 1000s of idle IoT devices

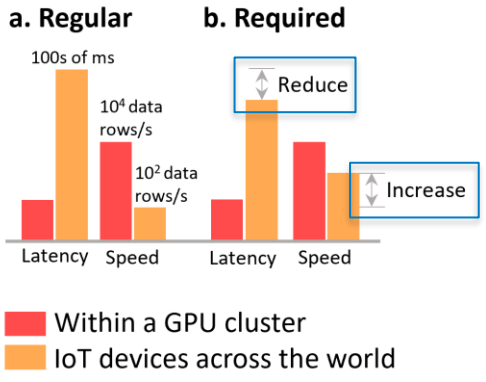
Distributed ML using IoT Devices - Challenge V

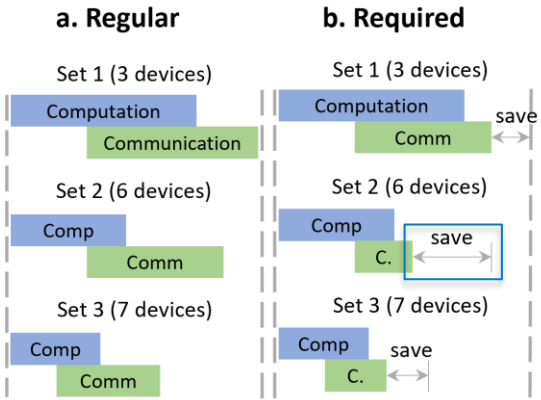
Confirm
Smart Manufacturing

Paper. Toward Distributed, Global, Deep Learning Using IoT Devices
Bharath Sudharsan and others
IEEE Internet Computing Journal

- **Staleness Effects** - produce stale parameters
  - ✓ Parameters arrive late
  - ✓ Slow convergence, degrade model perf
- **Slow Exchange of Model Gradients**
  - ✓ High latency & scalability issues
- Other challenges and bottlenecks
  - ✓ Expensive dataset loading I/O
  - ✓ High FLOPs consumption
  - ✓ Variable training and convergence speed, etc.

Network conditions across different continents



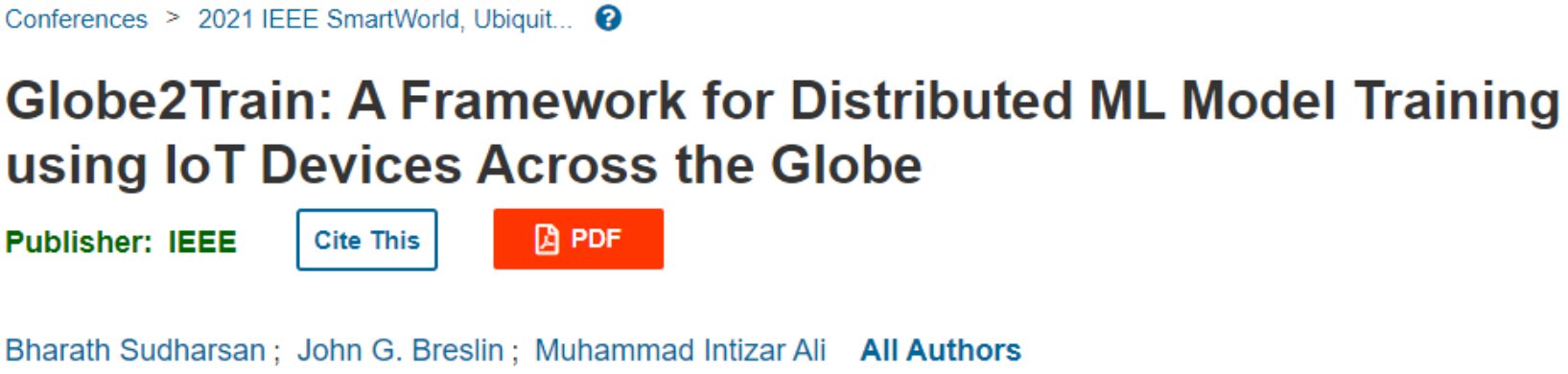


Aims. Gain ability to tolerate latency. Improve scalability by reducing communication-to-computation ratio

Distributed ML using IoT Devices - Objective V

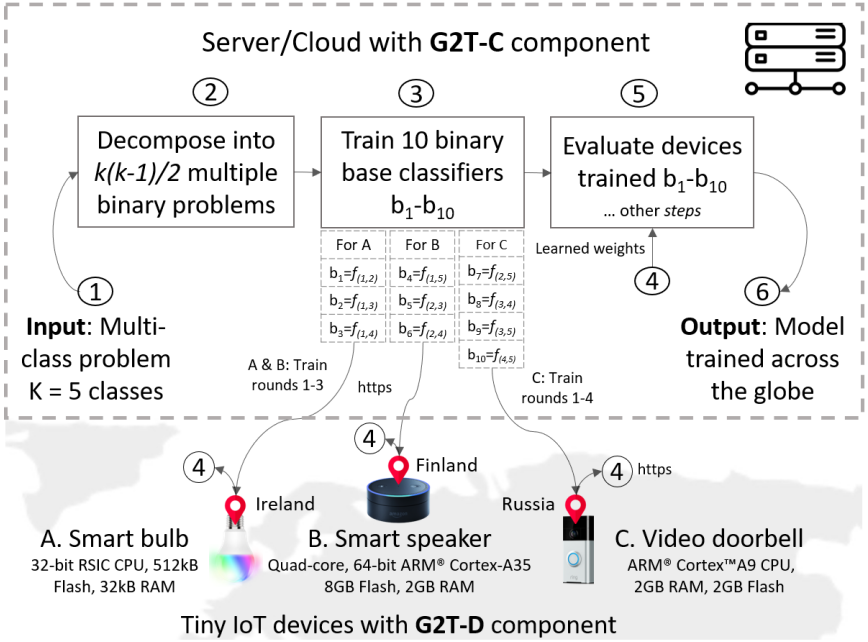
Confirm
Smart Manufacturing

- **Research Objective V.** Design a cloud-level and a device-level framework component that can improve distributed training scalability, speed while reducing communication frequency and tolerating network latency
- **Paper VI** fulfill Research Objective V



Distributed ML using IoT Devices - Globe2Train

Confirm
Smart Manufacturing



Globe2Train Cloud (G2T-C) component architecture

- Decompose 1 multi-class problem into k(k-1)/2 binary
- Other G2T-C steps are from our ML-MCU work
  - ✓ Create correlation matrix, probability table, etc.
  - ✓ Till producing prediction ready models
- G2T-C training contribution
  - ✓ Designed not to face staleness
  - ✓ Tasks executes in multiple rounds on IoT devices
  - ✓ Split tasks are independent of each other

Distributed ML using IoT Devices - Globe2Train

Confirm
Smart Manufacturing

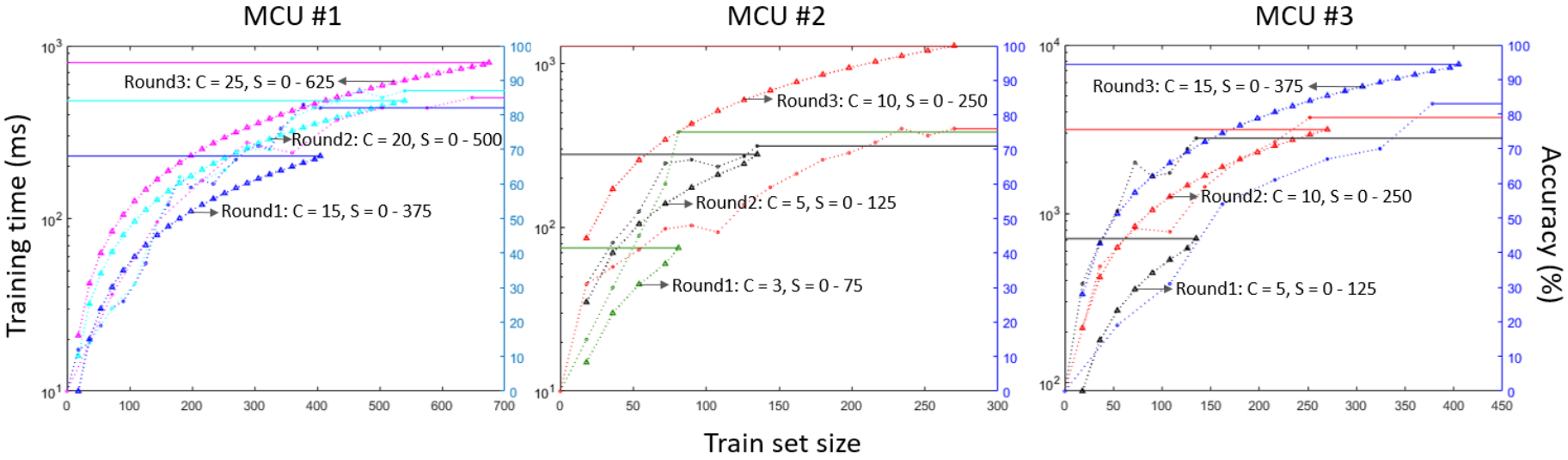
Algorithm 1 G2T-D to efficiently transmit the IoT device trained weights even under poor network conditions.

Input: Available weights $w_0, \ldots, w_n$ of IoT device trained base binary classifiers $b_1, \ldots, b_n$
Output: Efficiently transmit shrunken weights $w_{s0}, \ldots, w_{sn}$ to a central server with G2T-C component
Set weights threshold $W_t \leftarrow 1/\propto$ network quality
if n in $w_n > W_t$
    for i = 0, 1, 2, …, n **do**
        $w_{si} \leftarrow$ shrink the weight w_i. Use *encode* (w_i)
    end for
Transmit $w_{s0}, \ldots, w_{sn}$. Use HTTPS
else
    Locally accumulate weights till W_t

- Sets W_t high for devices with poor internet
  - ✓ Reduces frequent transmission of weights
  - ✓ So, conserving network bandwidth
  - ✓ i.e., weights are locally accumulated till it reaches W_t
- IoT hardware friendliness - implemented in few lines code
- G2T-D training contribution
  - ✓ Congestion and latency toleration
  - ✓ Improved scalability - reduced communication-to-computation ratio

Distributed ML using IoT Devices - Globe2Train

Confirm
Smart Manufacturing



Class count (C), Train time (T), Accuracy (A), and Train set size (S).

C = 3: T, A. C = 5: T, A. C = 10: T, A. C = 15: T, A. C = 20: T, A. C = 25: T, A

- We initiate distributed training from G2T-C component
  - ✓ Decomposes given 95 classes problem into 4465 binary problems - i.e., K = 95 in K(K-1)/2
  - ✓ b_0 to b_{4465} binary classifiers need to be trained by MCUs 1-3 in numerous rounds and in each round report back weights

Distributed ML using IoT Devices - Objective VI

Confirm
Smart Manufacturing

- **Research Objective VI.** Design a strategy to improve communication efficiency during distributed learning in IoT that can: Improve solution quality of resultant models by achieving lower loss and better accuracy; Reduce quantization induced variance, etc.
- **Paper VII** fulfill Research Objective VI

RESEARCH-ARTICLE OPEN ACCESS

ElastiQuant: elastic quantization strategy for communication efficient distributed machine learning in IoT

Authors: Bharath Sudharsan, John G. Breslin, Muhammad Intizar Ali, Peter Corcoran, Rajiv Ranjan Authors Info & Claims

SAC '22: Proceedings of the 37th ACM/SIGAPP Symposium on Applied Computing • April 2022 • Pages 246–254 • https://doi.org/10.1145/3477314.3507135

Distributed ML using IoT Devices - SOTA

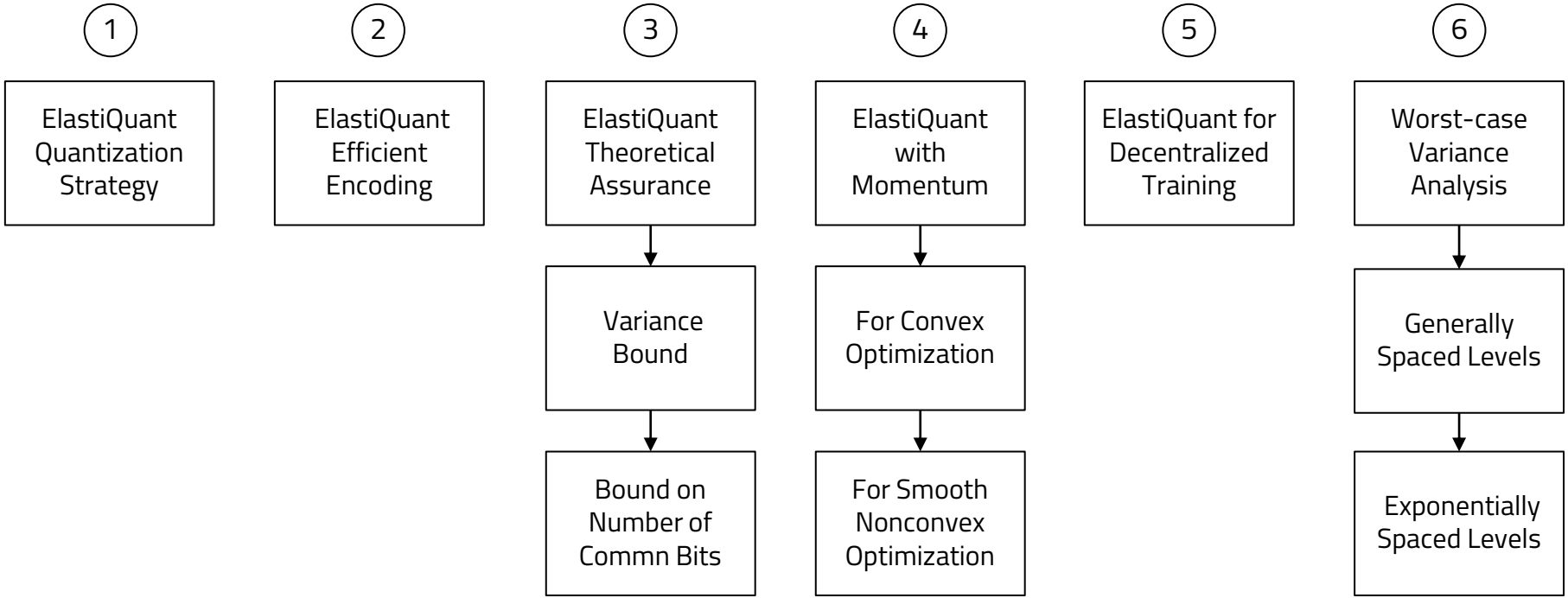
Confirm
Smart Manufacturing

Table 1: ElastiQuant comparison with related papers. Not Applicable (NA), Not Investigated (NI), No Guarantee/assurance (NG).

Paper	Test Setup	Gradients; Momentum	Variance	Com bits	Scalability	Worst-case	Highlight
ATOMO [30]	AWS EC2 cloud	Atomic sparsification; NI	NG	Bounds	Within cluster nodes	NI	Sparsification to minimize variance
Terngrad [32]	128-node each with 4 Nvidia P100	Quantize to ternary levels; Yes	NI	NI	Within cluster nodes	NI	Need only three levels to aggressively reduce communication time
Globe2Train [20]	MCUs, CPUs	NA	NI	NI	Global IoT devices	NI	Latency, congestion tolerance
AD-PSGD [13]	32-node each with 4 Nvidia P100	NI	Bounds	NI	Within cluster nodes	NI	Robust to heterogeneity
QSGD [1]	AWS EC2 cloud	Lossy compression; NI	Bounds	Bounds	Within cluster nodes	NI	Good practical performance
NUQSGD [15]	8 NVIDIA 2080 Ti GPUs	Nonuniform quantization; Yes	Assurance, bounds	Bounds	Within cluster nodes	Yes	Stronger guarantees, higher empirical performance
D2 [27]	16 workers	NI	Assurance, bounds	NI	Within cluster nodes	NI	Much improve convergence rate, robust to data variance
EF-SignSGD [11]	Multiple workers	Error-feedback; Yes	NI	NI	Within cluster nodes	NI	Simply add EF to recover performance
DGC [14]	64-node each with 4 Nvidia Titan XP	Deep compression; Yes	NI	NI	Within cluster nodes, mobiles	NI	270 x - 600 x gradient compression ratio without losing accuracy
PowerSGD [29]	8-node each with 2 Nvidia Titan X	Low-rank compression; Yes	NI	NI	Within cluster nodes	NI	Consistent wall-clock speedups, test performance on par with SGD
ElastiQuant	18 IoT boards, edge GPUs	Elastic quantization; Yes	Assurance, bounds	Bounds	IoT boards, mobiles, edge GPUs	Yes	Higher solution quality, scalability - assurance with results

Distributed ML using IoT Devices - ElastiQuant

Confirm
Smart Manufacturing



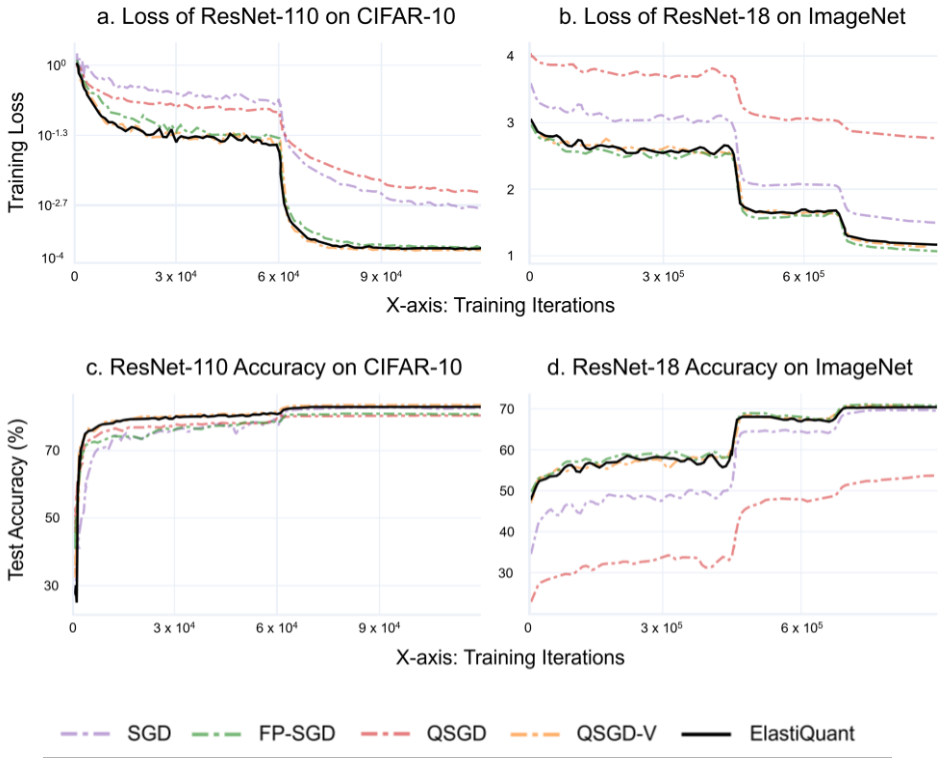
Distributed ML using IoT Devices - ElastiQuant

Confirm
Smart Manufacturing

- **Setup.** 18 development boards wirelessly set up to replicate real-world heterogeneous IoT:
  - ✓ 7 Jetson Xaviers
  - ✓ 4 Jetson Nanos inserted on a Jetson Mate carrier board
  - ✓ 3 accelerated Google Coral boards
  - ✓ 4 Intel Movidius NCS accelerated Raspberry Pi
- **Data.** Portions of ImageNet, CIFAR-10, CIFAR-100 datasets are supplied to these boards for distributed training
- **Network.** ResNet family - ResNet-18, ResNet-20, ResNet-34, ResNet-50, Resnet-110 are trained
- **Implementation.** ElastiQuant in TensorFlow

Distributed ML using IoT Devices - ElastiQuant

Confirm
Smart Manufacturing



- **Training Loss** - Distributed training on 18 GPUs
  - ✓ CIFAR10 - Significant gap in training loss which grows as training progresses
  - ✓ ImageNet - ElastiQuant, QSGD-V has lower loss compared to QSGD
- **Test Accuracy**
  - ✓ Unlike ElastiQuant, QSGD does not achieve the accuracy of FP-SGD
  - ✓ For both datasets, ElastiQuant and QSGD-V outperform QSGD
  - ✓ The gap between ElastiQuant and QSGD is significant on ImageNet
  - ✓ ElastiQuant and QSGD-V show lower variance in comparison to QSGD

Distributed ML using IoT Devices - ElastiQuant

Confirm
Smart Manufacturing

Table 3: Evaluating scalability performance of ElastiQuant by distributed training on 2 to 7 devices, and comparison with SGD: Calculating speedup (*Sp*) and total time (*T*) per epoch in minutes - sum of computation (*Cp*), encoding (*En*), transmission (*Tx*).

Network, Dataset	Scheme	2 Edge GPUs				4 Edge GPUs					7 Edge GPUs				
		Cp	*En*	*Tx*	*T*	*Cp*	*En*	*Tx*	*T*	*Sp*	*Cp*	*En*	*Tx*	*T*	*Sp*
ResNet-34, ImageNet	SGD	16.06	NA	17.93	33.99	10.47	NA	20.62	31.09	-2.9 ↑	15.34	NA	26.84	42.18	11.09 ↓
	8-bit-ElastiQuant	14.61	4.15	16.16	34.92	7.98	4.04	11.92	23.94	-10.98 ↑	10.05	4.04	10.26	25.35	1.41 ↓
	4-bit-ElastiQuant	15.75	0.93	15.55	32.23	11.19	1.25	9.32	21.76	-10.47 ↑	9.64	1.34	7.47	18.45	-3.31 ↑
	E-ElastiQuant	14.72	1.34	15.24	31.3	10.57	1.55	8.29	20.41	-10.89 ↑	8.91	1.35	6.01	16.27	-4.14 ↑
ResNet-50, ImageNet	SGD	177.23	NA	222.53	399.76	167.9	NA	265.17	433.07	33.31 ↓	85.28	NA	465.06	550.34	117.27 ↓
	8-bit-ElastiQuant	179.21	38.65	190.55	409.09	145.25	39.97	145.25	330.49	-78.62 ↑	99.94	38.64	183.89	322.47	-8.02 ↑
	4-bit-ElastiQuant	179.89	9.33	183.89	373.11	142.58	8.12	118.59	269.17	-103.94 ↑	110.6	6.66	126.59	243.85	-25.32 ↑
	E-ElastiQuant	169.23	21.32	171.9	362.45	126.59	18.66	103.93	249.18	-113.27 ↑	119.93	19.99	78.62	218.54	-30.64 ↑

- SGD **negative** scalability - T increases from 31.09 to 42.18 min for ResNet-34, and 433.07 to 550.34 min for ResNet-50
- 4-bit-ElastiQuant attains **positive** scalability - 1.48 times speedup for ResNet-34 when GPUs scaled from 2 to 4
- 8-bit-ElastiQuant faces a scalability **stall** when GPUs scaled from 4 to 7 - due to elevated encoding & communication costs
- E-ElastiQuant shows **top-of-the-class** scalability and communication compression

Distributed ML using IoT Devices - ElastiQuant

Confirm
Smart Manufacturing

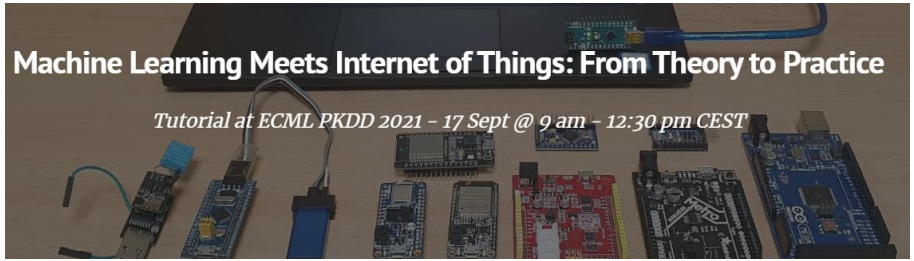
Table 4: Accuracy comparison of ElastiQuant trained ResNets with models trained using DGC, compression ratio (CR) tuned DGC, Atomo, TernGrad, others. FP-SGD is baseline.

Network, Dataset	Scheme	Tune	Edge GPUs	Test Accuracy (%)
ResNet-20, CIFAR-10	Atomo	Default	2	87.6
	TernGrad			No convergence
	FP-SGD			90.5
	4-bit-ElastiQuant			86.3
ResNet-18, CIFAR-10	DGC	0.02 CR	2	91.25
			6	88.87
		0.12 CR	2	90.08
			6	87.36
	FP-SGD	Default	6	92.72
	4-bit-ElastiQuant		6	91.96
ResNet-18, CIFAR-100	DGC	0.02 CR	2	74.41
			6	72.69
	FP-SGD	Default	6	74.33
	4-bit-ElastiQuant			73.63
ResNet-110, CIFAR-10	SGD	Default	6	89.76
	QSGD			89.22
	QSGD-V			90.10
	FP-SGD			92.03
	TernGrad			91.33
	4-bit-ElastiQuant			90.80

- ElastiQuant vs Deep Gradient Compression (DGC)
  - ✓ ResNet-18 on CIFAR-10, CIFAR-100 - unlike ElastiQuant, DGC **accuracy degrades** when GPUs scaled from 2 to 4
  - ✓ So even if ElastiQuant could save lesser commn bandwidth than DGC, ElastiQuant is **practical** due to its better scalability
- ElastiQuant vs ATOMO and TernGrad
  - ✓ For ResNet-20, although ATOMO shows slightly higher accuracy than ElastiQuant, ATOMO has **high train time** plus GPU strain
  - ✓ TernGrad convergence was **under par** for standard parameters - tuning to bring performance close to ATOMO & ElastiQuant
  - ✓ For ResNet-110, TernGrad shows the closest performance to FPSGD and slightly **outperforms** 4-bit-ElastiQuant

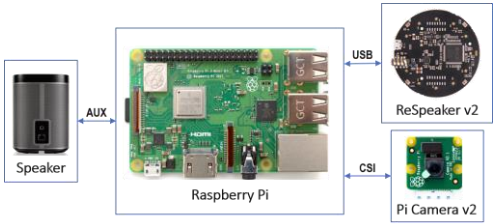
Research Outreach

Confirm
Smart Manufacturing

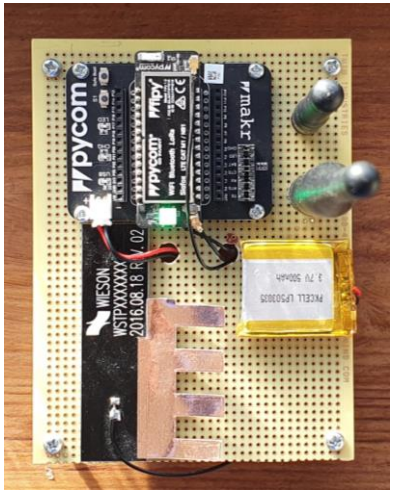


Website: https://sites.google.com/view/ml-meets-iot-ecml-tutorial/

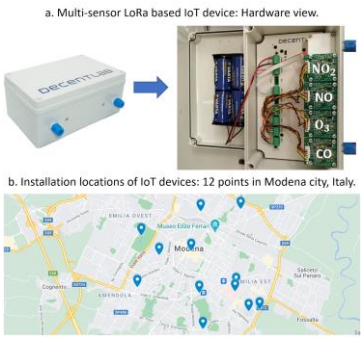
Avoid Touching Your Face: COVID-away Dataset and Models



Advanced Alexa prototypes @ AICS 2019



RIS-IoT @ ICCPS 2022



Air analytics @ UbiComp 2022

OWSNet @ WF-IoT 2021

Precision Agri @ IPSN 2022

10+ short papers - Demos and Posters

Publication List

Confirm
Smart Manufacturing

Only 7 papers are used for the thesis and covered in this talk. Others include:

Magazine Articles

- ✓ [B] First author. OTA-TinyML: Over the Air Deployment of TinyML Models and Execution on IoT Devices @ IEEE Internet Computing
- ✓ [B] First author. Towards Distributed, Global, Deep Learning using IoT Devices @ IEEE Internet Computing

Demonstration at Conferences

- ✓ [A*] Co-author. TinyFedTL: Federated Transfer Learning on Ubiquitous Tiny IoT Devices @ IEEE PerCom
- ✓ [A] Co-author. GNOSIS- Query-Driven Multimodal Event Processing for Unstructured Data Streams @ ACM Middleware
- ✓ First author. SRAM Optimized Porting and Execution of Machine Learning Classifiers on MCU-based IoT Devices @ ACM/IEEE ICCPS
- ✓ First author. TinyML Benchmark: Executing Fully Connected Neural Networks on Commodity Microcontrollers @ IEEE WF-IoT

Posters at Conferences

- ✓ [A*] Co-author. Embedded ML Pipeline for Precision Agriculture @ ACM/IEEE IPSN
- ✓ [A*] First author. Training up to 50 Class ML Models on 3$ IoT Hardware via Optimizing One-vs-One Algorithm @ AAAI
- ✓ [A*] First author. Approach for Remote, On-Demand Loading and Execution of ML Models on Arduino IoT Boards @ ACM/IEEE IPSN
- ✓ [A*] First author. ElastiCL: Elastic Quantization for Communication Efficient Collaborative Learning in IoT @ ACM SenSys
- ✓ [A*] Air Quality Sensor Network Data Acquisition, Cleaning, Visualization, and Analytics @ ACM UbiComp-ISWC
- ✓ First author. RIS-IoT: Towards Resilient, Interoperable, Scalable IoT @ ACM/IEEE ICCPS

Applied Research Papers at Conferences

- ✓ First author. Ensemble Methods for Collaborative Intelligence: Combining Ubiquitous ML Models in IoT @ IEEE BigData
- ✓ First author. Edge2Guard: Botnet Attacks Detecting Offline Models for Resource-Constrained IoT Devices @ IEEE PerCom Workshop
- ✓ First author. OWSNet: Towards Real-time Offensive Words Spotting Network for Consumer IoT Devices @ IEEE WF-IoT
- ✓ First author. RCE-NN: A Five-stages Pipeline to Execute Neural Networks (CNNs) on Resource-constrained IoT Edge Devices @ ACM IoT
- ✓ First author. Edge2Train: A Framework to Train Machine Learning Models (SVMs) on Resource Constrained IoT Edge Devices @ ACM IoT
- ✓ First author. Adaptive strategy to improve quality of communication for IoT edge devices @ IEEE WF-IoT

Conclusion and Future Work

Confirm
Smart Manufacturing

- **On-Device Learning**
  - ✓ ML-MCU sets SOTA - ML training on 3$ hardware using 50 class data and unit inference in super real-time
  - ✓ Investigate Train++, ML-MCU for split-learning, federated learning, centralized learning, distributed ensemble learning
- **Efficient ML on IoT Devices**
  - ✓ Consume 1.61-38.06 % less SRAM to produce inference results when using popular NNs
  - ✓ Port and execute ML classifiers 1-4x times faster than SOTA libraries
  - ✓ Extend work to support other classifiers SVMs, kNN, AdaBoost, etc.
- **Distributed ML using IoT Devices**
  - ✓ Improved solution quality, reduced quantization induced variance, higher training scalability and speedup
  - ✓ Investigate Globe2Train and ElastiQuant for decentralized and collaborative learning applications