TINYMLOPS FOR REAL-TIME ULTRA-LOW POWER MCUS APPLIED TO FRAME-BASED EVENT CLASSIFICATION

Minh Tri Lê (TDK InvenSense & Inria Grenoble Rhône-Alpes)

Julyan Arbel (Inria Grenoble Rhône-Alpes)

TDK InvenSense

EuroMLSys 2023

Inria

1. Background and Motivation

Background:

- **TinyML**: Emerging field at the intersection of Machine Learning (ML) and the Internet of Things (IoT).
  - Why? Enable intelligent processing of real-time data and close to the source, offers privacy, low-cost systems, new opportunities.
  - Applications: Gesture recognition, keyword spotting, fingerprint detection, anomaly detection, health monitoring, ...
  - Challenges: High power footprint algorithm on low-power devices (microcontrollers, sensors, ...), heterogeneous hardware ecosystem, early stage of the field → **Lack of mature tools and practices**.
- **TinyMLOps**: Specialized subset of Machine Learning Operations (*MLOps*) focusing on the best practices to deploy ML models on low-power embedded systems.
- **Our use case**:
  - **Neural networks** on **ultra-low power microcontrollers** for **real-time, always-on event classification**,
  - Continuous **frame-by-frame** processing: **1 input data → 1 output decision**.

Problem:

- What are the specific challenges and solutions to design and deploy tinyML solutions ?
- How to apply it our frame-based event classification?

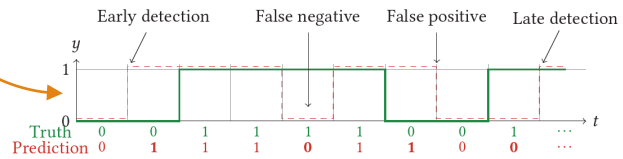


Figure 1: Example of a frame-by-frame event-based classification encoded by binary vectors. We observe four possible misclassifications.

2. TinyMLOps workflow

TinyMLOps:

- Shares common challenges and solutions with MLOps
- Differs because target hardware is **highly constrained!** $\leq 10^3$ **KB** memory, 10^2 **MHz** clock, **1 mW** scale, integer-only inference, no explicit division nor exponentials, real-time inference, always-on, battery-powered
- Early stage → Scarcity of mature and comprehensible tools for deployment of neural networks in tinyML

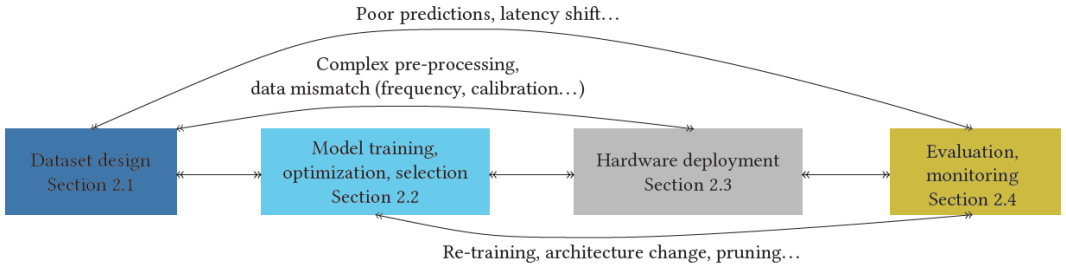


Figure 2: TinyMLOps: Steps and examples of indirect iteration causes

Problem:

- No straightforward iteration → Indirect iteration causes

2.1. Dataset design

Frame-based event classification:

- Continuous input of sensor data (e.g., accelerometer, sound frequencies, ...) → expecting a continuous output of decisions
  1 frame → 1 input or output at frequency f.

Challenges:

- Choice of frequency f (high: more precise, higher consumption and vice-versa)
- Precise labeling, start/end of an event?
- Build realistic dataset from noisy sensor data.

Solutions:

- Define early/late accept of frames (softer labels)
- Data augmentation (e.g., background noise, shifts, scaling)

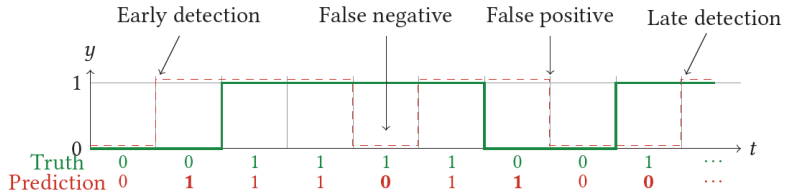


Figure 1: Example of a frame-by-frame event-based classification encoded by binary vectors. We observe four possible misclassifications.

2.2. Model training, optimization, and selection

Challenges:

- What model architectures for ultra-low power microcontrollers?
- High power footprint in **computation and size.**
- Integer-only inference, no explicit division ($\neq 2^n$)
- Model has poor performance → Tune hyper-parameters or go back to dataset design.

Solutions:

- Convolutional 1D GRU are polyvalent with good size-performance tradeoff for sequence classification [1],
- Model compression [2]: Pruning, knowledge distillation, low-rank matrix decomposition, weight sharing,
- Quantization [2]: Reduce parameter precision from 32-bits floating point to lower-bit integer (e.g., 8-bits), **mandatory step**.

[1] Roberto Cahuantzi, Xinye Chen, and Stefan Güttel. 2021. A Comparison of LSTM and GRU Networks for Learning Symbolic Sequences. arXiv:2107.02248 [cs] http://arxiv.org/abs/2107.02248
[2] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. 2020. A Survey of Model Compression and Acceleration for Deep Neural Networks. (2020). arXiv:1710.09282 [cs] http://arxiv.org/abs/1710.09282

2.3. Hardware deployment and code generation

Challenges:

- Scarcity of suitable library for model conversion/inference on low-power embedded hardware.
- Heterogeneous hardware landscape: from vendor to vendor or even series of products.
- If model contains unsupported operations (pre-processing or model inference stage) → Need to go back to previous steps.

Solutions:

- CMSIS-NN: Low-level library for efficient neural network inference on microcontrollers
- Frameworks to convert, quantize and generate deep learning models compatible with microcontrollers:
  - TensorFlow Lite Micro (TFLM) [3]: Interpreter-based → wide hardware support, open-source, good performance, TensorFlow ecosystem, missing some operations (e.g., GRU, some activations, ...), no target-specific optimizations, difficult to customize and debug
  - NNoM [4]: C code generation, easy to interpret, debug, wide hardware support, open-source, support CMSIS-NN for optimized inference, smaller community and adoption, may slow future development and bugfix, but we found unstable performance results.

Our solution:

- **Create our own tinyMLOps framework to train and deploy efficient neural networks on ultra-low power microcontrollers, added missing supported operations or options (GRU, Conv1D), ...**
- **C code generation for wide support, CMSIS-NN for fast inference, minimal code, lightweight deployment.**

[3] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. 2021. TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems. (2021). arXiv:2010.08678 [cs] http://arxiv.org/abs/2010.08678
[4] Jianjia Ma. 2020. A higher-level Neural Network library on Microcontrollers (NNoM). https://doi.org/10.5281/zenodo.4158710
[5] Abbas Ataya. 2022. Tiny ML for Tiny Sensors: Waking smarter for less. https://cms.tinyml.org/wp-content/uploads/summit2022/Abbas-Ataya.pdf

2.4. Evaluation and monitoring

Challenges:

- Evaluate and verify neural network performance and support on embedded systems
- Bad results → Need to go back to previous steps.

Solutions:

- **Our tinyMLOps framework [5]:** Based on plain C code, easy to debug, and lightweight, good performance

Table 1: Comparison of a model quantized to int8 deployed on an Arm Cortex-M4 MCU using tf.lite [8], NNoM [22] and our tinyMLOps solution on an activity recognition dataset

Metric	tf.lite	NNoM	Our
Accuracy (%)	85.5	68.24	86.95
Model size (KB)	6.72	0.29	1.41
Stack size (KB)	12	6.1	5.5
Code memory (KB)	303	16.12	5.5

[5] Abbas Ataya. 2022. Tiny ML for Tiny Sensors: Waking smarter for less. https://cms.tinyml.org/wp-content/uploads/summit2022/Abbas-Ataya.pdf

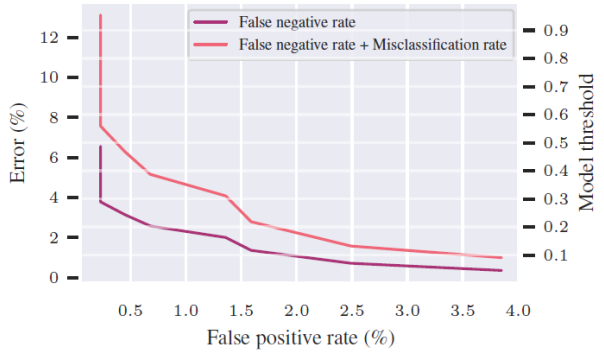
2.4. Evaluation and monitoring

Challenges:

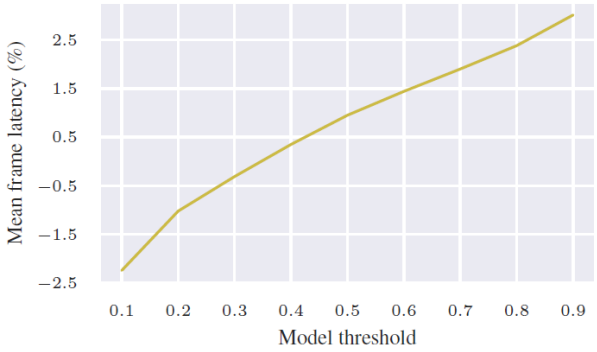
- Finding and measuring meaningful metrics that reflects on-device user experience before release
- Ambiguous errors in frame-based events: early/late detection

Solutions:

- Create custom metrics and tune the optimal model output threshold (softmax/sigmoid input) and plot the results.
- Use user-define early/late acceptance margin of frame, based on the application (Responsive but lower quality inference vs slower with higher quality inference?)



(a) False positive rate versus error rate for different error types (See Appendix A.1 listing errors in a confusion matrix): False negative rate and misclassification rate between positive classes are measured by varying the model output threshold.



(b) Model threshold versus average number of frames latency. Early accepted frames are defined as those captured 10 prior to the actual event, and late accepted frames are set to 20 after the actual event. Frame rate is at 20Hz.

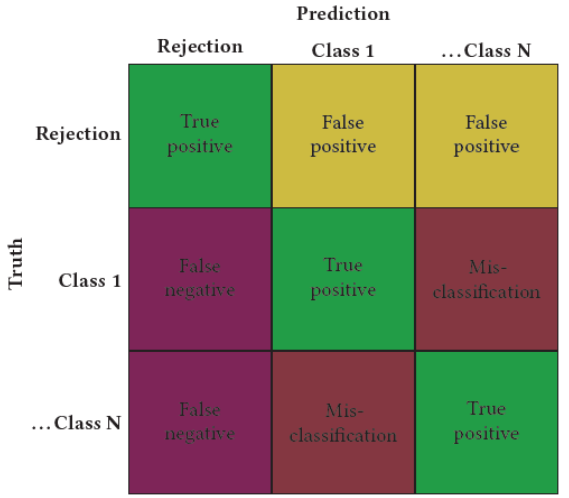


Figure 4: Error types of a generalized confusion matrix for multiclass event classifications with a rejection class (i.e., non-event). There is an additional error type compared to binary event classification: misclassification between positive classes.

Conclusion

- TinyMLOps: unique set of challenges and solutions, non-linear process and nascent field.
- Our tinyMLOps solution: Competitive results with existing solutions, but is more stable and lightweight, while keeping performance
- Frame-based event classification: careful consideration on datasets and metrics for real-time inference on ultra-low power microcontrollers.
- TinyML: high potential, needs innovation/adoption of efficient tools and methods by researchers and industrials to deploy neural networks at the edge.