Benchmarking microcontrollers with ultra-low resolution images classification

Ali Dabbous[1[0009-0004-8978-4979]], Riccardo Berta[1[0000-0003-1937-3969]], Luca Lazzaroni[1[0000-0001-8092-5473]], Alessandro Pighetti[1[0009-0001-7166-5750]], and Francesco Bellotti[1]

[1] Department of Electrical, Electronic and Telecommunication Engineering (DITEN), University of Genoa, Via Opera Pia 11a, 16145 Genova, Italy
ali.dabbous@unige.it

Abstract. The advancement of edge devices equipped with specialized hardware accelerators, data caches, and microcontroller units (MCUs) has brought the deployment and execution of Deep Neural Network (DNN) models closer to users and real-world sensor systems. This paper explores the potential of various specialized MCUs across four real-world applications (waste classification, presence detection, miniature robot detection, and sign language interpretation). We evaluate three well-known MCU (STM32H7, Arduino, and MAX78000) comparing their inference time and power/energy consumption on four ultra-low resolution image-classification datasets with varying input and task complexities. Our findings indicate that all MCUs deliver excellent performance in terms of inference time and energy consumption for real-time applications, with the MAX78000 outperforming the others in all the metrics considered, thanks to the use of a DNN accelerator.

Keywords: Convolutional Neural Network, Hardware Accelerator, STM32, MAX78000, Energy Consumption, Tiny Machine Learning.

1 Introduction

Tiny machine learning (TinyML) techniques have been created to address the limitations of resource-constrained devices that can be deployed near sensors in the field (e.g., [1]). These techniques provide the essential advantages of edge computing, including reduced latency, enhanced data security, greater energy efficiency, and decreased bandwidth usage, all while eliminating the need for continuous connectivity (e.g.,[2–5]).

However, the diverse range of applications, the heterogeneity of constraints and embedded tiny devices, the variety of edge-to-cloud communication architectures, and the complexity of ML models complicate the work of IoT sensor and application designers. This complexity makes the design process costly and time-consuming [6], necessitating the use of appropriate methods and tools [7, 8]. Additionally, benchmarking the edge devices with TinyML models is crucial to ensure optimal performance, efficiency, and suitability for specific applications[9].

The literature documents significant achievements through quantization and the deployment of tiny models. For example, [10] presents a memory-efficient deployment of binary Neural Networks for robotic touch classification on an ARM Cortex-M7 microcontroller (MCU), achieving state-of-the-art accuracy with a notable reduction in memory footprint and latency. Another example is [11], which proposes a Single Perceptron Linear Vector Processor MCU accelerator that reduces inference time with minimal resource usage.

The ARM Cortex Microcontroller Software Interface Standard Neural Network (CMSIS-NN) open-source library is designed to maximize neural network performance on Cortex-M processors [12]. Similarly, the X-Cube-AI package facilitates the development of deep learning (DL) models on STM 32-bit (STM32) MCUs [13]. ExecuTorch, currently in its preview version, offers an end-to-end solution for enabling on-device inference across mobile and edge devices [14]. The MAX78000 AI MCU with a Neural Network Accelerator is tailored for AI applications that demand significant computational power. Other notable examples for tiny devices include Edge Learning Machine [15], FANN-on-MCU [16], and Edge Impulse [17]. These packages offer functionalities such as automation, interoperability, and deployability awareness.

In this paper, we propose tiny Convolutional Neural Network (CNN) models for image classification and investigate the effectiveness of different MCUs in executing TinyML tasks. We evaluate the performance of three well-known MCUs in the embedded domain, focusing on accuracy, inference time, and power and energy consumption. The MCUs under comparison are the STM32H7 with a data cache board, favored for its robust processing capabilities in professional applications, the Arduino Nano BLE 33, and the MAX78000 MCU with a CNN accelerator. We benchmark these MCUs using four real-world applications, namely waste classification, presence detection, miniature robot detection, and sign language interpretation, common areas of study in the TinyML domain, with different number of target classes and difficulty levels. Given that real-world applications like mobile robots and sign language interpretation are often battery-powered, optimizing energy consumption is crucial to extending operational lifespan and enhancing the practicality of micro-robots and system architecture applications in real-world scenarios. We focus on image classification due to its relevance in fields such as automotive and robotics and the significant computational challenges it poses (e.g., [13]).

2 Methodology

2.1 Microcontrollers

We conducted an experimental analysis using three MCU boards, as detailed in Table 1. The first board represents the main industrial series from ST, specifically the high-performance H series. The Arduino Nano 33 BLE Sense Rev2 features a compact form factor, various environmental sensors, and the ability to run Tiny ML and TensorFlow Lite. Lastly, the MAX78000 is an AI MCU designed for running neural networks with ultra-low power consumption on small-scale devices. It includes two

primary processing modules: a dual-core MCU and a CNN accelerator. The MCU consists of an Arm Cortex-M4 processor with a floating-point unit (FPU) operating at up to 100 MHz, and a 32-bit RISC-V co-processor running at up to 60 MHz. The hardware-based CNN accelerator contains 64 parallel physical channel processors, each with a pooling unit and a convolutional engine with dedicated weight memory. The CNN engine offers 442KB of weight storage memory and supports weights of 1, 2, 4, and 8 bits.

Table 1. Microcontrollers specifications

Board	**RAM [kiB]**	**ROM [kiB]**	**Processor speed [MHz]**
STM32H7B3I	1,400	2,000	480
Arduino Nano 33 BLE Sense Rev2	256	1,000	64
MAX78000	512	128	100

2.2 Dataset

This paper evaluates the performance of the proposed MCU benchmarking for image classification through four case studies, each representing a range of low-resolution imaging applications suitable for edge deployment. The tasks include: (i) detecting miniature mobile robots using an ultra-low resolution time-of-flight sensor, (ii) classifying visible-light images of electronic components, (iii) detecting presence in thermal images, and (iv) recognizing sign language digits in thermal images.

The first application “time-of-flight (TOF)” dataset , [20], contains 4,150 unique time-of-flight images that either include or exclude a miniature mobile robot. These images were captured using the ST VL53L5CX ultra-low resolution time-of-flight sensor, which can measure distances up to 4 meters at a frequency of 60 Hz. This sensor captures depth images with a resolution of up to 8×8 8-bit pixels, where each pixel represents a distance ranging from 0 to 50 cm. A comprehensive description of the dataset can be found in [20], which also introduces an MCU-deployed classifier based on a custom CNN architecture that achieves an accuracy of 91.8%.

The second dataset “Electronic Parts (EPD)” focuses on classifying used electronic components for an e-waste application [21]. It comprises 1,734 grayscale images, each with a resolution of 30×30 pixels, categorized into three classes: capacitors, potentiometers, and voltage regulator integrated circuits. Experiments are conducted using both shallow ML classifiers and CNNs, with the CNNs achieving the highest accuracy of 98.4%.

The third dataset “Infrared Thermal (ITD)” [22] consists of low-resolution infrared thermal images of people and objects collected in indoor environments. Using the MLX90640 far-infrared thermal sensor array, which has a resolution of 32×24 pixels, a total of 1,770 images were captured at approximately 15°C. The dataset identifies four distinct classes: 'one person', 'two people', 'objects', and 'people and objects'. A CNN-based approach for human detection achieves an accuracy of 90.3% in a simplified binary classification task, identifying the presence or absence of at least one person in the space. No classification experiments are conducted on the four classes.

The fourth dataset “Thermal Sign Language Digits (SFD)” [23] comprises 3,200 thermal images, with equal representation of each sign language digit across 10 classes. The images were captured using a low-resolution microelectromechanical systems (MEMS) thermal camera, the Omron D6T, which has a resolution of 32×32 pixels. Using this dataset, [24] introduces an end-to-end edge hand gesture classification system that leverages a lightweight deep residual learning model on a Raspberry Pi, achieving a test set accuracy of 99.52%.

2.3 2D Classifier Convolutional Networks Architecture

In this paper, we propose tiny CNN architectures with one to two convolutional layers for image classification.. The first step in the challenging training process involves fine-tuning the hyperparameters of the classifier architecture through cross-validation on the training set. A grid of potential candidates was explored, emphasizing solutions with minimal parameters and high computational efficiency. Table 2 shows the results of a hyperparameter study conducted on models featuring 2-D convolutional layers, for each dataset.

Table 2. CNN Configuration.

Dataset	**Kernels**	**Kernel size**	**Dropout**
TOF	[32,32]	3 × 3	0.3
EPD	[64, 32]	3 × 3	0.3
ITD	[64, 32]	3 × 3	0.3
SFD	[32]	5 × 5	0.5

As an activation function, we employ the ReLU. We employed the Adam optimizer for all datasets with learning rate equal to 0.0001. In all the datasets the network is trained over 200 epochs to keep consistency. Since the MAX78000 does not support on-chip training, models must be trained and compiled in a desktop environment before being deployed on the chip for inference using the model binary. MAX78000 supports PyTorch for model development, offering custom layer implementations for quantized and fused operations, as well as pipelines for quantization-aware training or post-training quantization. For each dataset in this study, we developed the same model with two different implementations: one carried out using the TensorFlow framework and its convenient wrapper, Keras, which is supported by STM32 and Arduino MCUs, and the other using the PyTorch version supported by the MAX78000. Given the MCU target, we employed a Quantization-Aware Training (QAT) approach for model training , quantizing all the NN weights to 8 bits. QAT is a method that enables the model to adapt to the quantization process during its training phase.

For STM32 MCUs deployment, the STM32Cube.AI tool offers a utility function to import a .tflite file and convert it into a C byte array within a .h header file. This file can then be seamlessly integrated into an STM32 project, allowing inference on the

MCU using the X-Cube AI tool. Additionally, a .tflite file can be converted into a C byte array using the "xxd" command and then uploaded to the Arduino IDE. The TensorFlow Lite Micro interpreter is then used to run the model on an Arduino device. For the MAX78000, after training, specific tools convert PyTorch checkpoints or TensorFlow-exported ONNX files into C code, which is then compiled and executed on the MAX78000 through a process called synthesis. During synthesis, a YAML file defining the model architecture and processor IDs is created, along with a dataset sample file. Following training and synthesis, the deployment process embeds the converted C code on the MAX78000 for execution. The device supports various accelerated operations, including 1D convolution with kernel sizes from 1 to 9, 2D convolution with 1×1 or 3×3 kernel sizes, linear (fully connected) layers, max pooling, average pooling, ReLU and Abs activations, and batch normalization. Other operations can be executed on the Arm or RISC-V cores, albeit with increased latency. During model execution, the MCU transfers configuration data, input data, and weights to the CNN accelerator, which processes each layer sequentially. Kernels are loaded from memory for each layer, and input channels are assigned to convolutional processors, achieving acceleration through parallel processing across multiple processors.

3 Results

Table 3 reports the inference time, power consumption, and energy consumption for the proposed models across the mentioned edge devices. We employed a USB multimeter connected between the power supply and the device to track power variations. Energy is then computed by multiplying power per inference time. Reported values are averages over 100 runs.

Results are also compared with the CNN-based model presented in literature. In all cases, our model outperforms the state-of-the-art baseline in terms of accuracy, in which our proposed model achieved 98.07% (TOF), 99.63% (EPD), 95.99% (ITD), and 100% (SFD). For the EPD and ITD datasets, literature doesn’t provide any embedded computation analysis, while for TOF our model deployed outperforms the state of the art solution in terms of all computation metrics. Considering the three platforms under test, the MAX78000 outperforms the other MCUs in terms of energy consumption and inference time. In particular, the energy consumption and inference time for MAX78000 are much lower than the STM32H7 and Arduino MCUs for all dataset, which makes this MCU the most suitable choice in scenarios with limited or sporadic power availability such as the miniature mobile robot application. However, MAX78000 has very high learning curve and training on it is not as easy as other platforms, specially it does not support TensorFlow Lite models. Nevertheless, infe ence times across all boards remain always within acceptable ranges, making them all reasonably suitable for real-time execution.

Table 3. MCUs Performance.

Dataset	Platform	Power [mW]	Energy [µJ]	Inference time [ms]
TOF	STM32 H7	223.72	132	0.590
	Arduino	1.43	10.29	7.2
	MAX78000	**12.54**	**0.66**	**0.053**
	Raspberry Pi [20]	108.35	3,684.10	34.00
EPD	STM32 H7	227.86	6,892.79	30.25
	Arduino	1.43	751.36	522.15
	MAX78000	**9**	**13.05**	**1.45**
ITD	STM32 H7	227.85	4,996.80	21.93
	Arduino	1.43	526.72	366.04
	MAX78000	**48**	**51.36**	**1.07**
SFD	STM32 H7	230.43	3,470.40	15.06
	Arduino	1.43	345.62	240.04
	MAX78000	**26**	**20.02**	**0.77**
	Raspberry Pi [24]	--	--	30

4 Conclusion

Executing Tiny ML models on MCUs has demonstrated significant performance improvements in real-world applications. Each MCU has its own specifications, including data cache, memory, flash size, and accelerators. In experiments conducted across four datasets from different application domains, the MAX78000, utilizing a CNN architecture, outperformed other mainstream and well-known embedded devices (such as STM32H7 and Arduino) in terms of inference time and energy consumption. However, training on MAX78000 is not straight forward as other MCUs, and it has limitation in RAM size, such as the maximum number of features that cannot be more than 1024, while STM32 boards have a range of RAM size up to 1.6 MB. Future work will focus on designing a toolkit for creating Tiny ML models, studying hyperparameters, and developing APIs for model conversion and deployment on various MCUs. This will enable real-world applications with just one click and reduced time investment.

Acknowledgements

This work has been supported by the GEMINI "Green Machine Learning for the IoT" national research project, funded by the MUR under the PRIN 2022 program (Contract 20223M4HZ4).

References

1. Sakr, F., Berta, R., Doyle, J., Capello, A., Dabbous, A., Lazzaroni, L., Bellotti, F.: CBin-NN: An Inference Engine for Binarized Neural Networks. Electronics. 13, 1624 (2024). https://doi.org/10.3390/electronics13091624.
2. Wu, Q., Zhao, Y., Fan, Q., Fan, P., Wang, J., Zhang, C.: Mobility-Aware Cooperative Caching in Vehicular Edge Computing Based on Asynchronous Federated and Deep Reinforcement Learning. IEEE Journal of Selected Topics in Signal Processing. 17, 66–81 (2023). https://doi.org/10.1109/JSTSP.2022.3221271.
3. Fresta, M., Dabbous, A., Bellotti, F., Capello, A., Lazzaroni, L., Pighetti, A., Berta, R.: Low-Cost, Edge-Cloud, End-to-End System Architecture for Human Activity Data Collection. In: Bellotti, F., Grammatikakis, M.D., Mansour, A., Ruo Roch, M., Seepold, R., Solanas, A., and Berta, R. (eds.) Applications in Electronics Pervading Industry, Environment and Society. pp. 444–449. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-48121-5_64.
4. Fresta, M., Bellotti, F., Capello, A., Dabbous, A., Lazzaroni, L., Ansovini, F., Berta, R.: End-to-End Dataset Collection System for Sport Activities. Electronics. 13, 1286 (2024). https://doi.org/10.3390/electronics13071286.
5. Dabbous, A., Fresta, M., Bellotti, F., Berta, R.: Neural Architecture for Tennis Shot Classification on Embedded System. In: Bellotti, F., Grammatikakis, M.D., Mansour, A., Ruo Roch, M., Seepold, R., Solanas, A., and Berta, R. (eds.) Applications in Electronics Pervading Industry, Environment and Society. pp. 97–102. Springer Nature Switzerland, Cham (2024). https://doi.org/10.1007/978-3-031-48121-5_14.
6. Jha, D.N., Alwasel, K., Alshoshan, A., Huang, X., Naha, R.K., Battula, S.K., Garg, S., Puthal, D., James, P., Zomaya, A., Dustdar, S., Ranjan, R.: IoTSim-Edge: A simulation framework for modeling the behavior of Internet of Things and edge computing environments. Software: Practice and Experience. 50, 844–867 (2020). https://doi.org/10.1002/spe.2787.
7. Dwivedi, Y.S., Singh, R., Sharma, A.K., Sharma, A.K.: Enhancing the Performance of Photonic Sensor Using Machine-Learning Approach. IEEE Sensors Journal. 23, 2320–2327 (2023). https://doi.org/10.1109/JSEN.2022.3225858.
8. Dabbous, A., Berta, R., Fresta, M., Ballout, H., Lazzaroni, L., Bellotti, F.: Bringing Intelligence to the Edge for Structural Health Monitoring. The Case Study of the Z24 Bridge. IEEE Open Journal of the Industrial Electronics Society. 1–15 (2024). https://doi.org/10.1109/OJIES.2024.3434341.
9. Berta, R., Dabbous, A., Lazzaroni, L., Pau, D., Bellotti, F.: Developing a TinyML Image Classifier in a Hour. IEEE Open Journal of the Industrial Electronics Society. 1–15 (2024). https://doi.org/10.1109/OJIES.2024.3451959.

10. Sakr, F., Berta, R., Doyle, J., Younes, H., De Gloria, A., Bellotti, F.: Memory Efficient Binary Convolutional Neural Networks on Microcontrollers. In: 2022 IEEE International Conference on Edge Computing and Communications (EDGE). pp. 169–177 (2022). https://doi.org/10.1109/EDGE55608.2022.00032.
11. An 8-bit Single Perceptron Processing Unit for Tiny Machine Learning Applications | IEEE Journals & Magazine | IEEE Xplore, https://ieeexplore.ieee.org/abstract/document/10295439, last accessed 2024/07/03.
12. Ltd, A.: CMSIS, https://www.arm.com/technologies/cmsis, last accessed 2023/10/07.
13. X-CUBE-AI - AI expansion pack for STM32CubeMX - STMicroelectronics, https://www.st.com/en/embedded-software/x-cube-ai.html, last accessed 2023/10/07.
14. Welcome to the ExecuTorch Documentation — ExecuTorch documentation, https://pytorch.org/executorch/stable/index.html, last accessed 2023/10/18.
15. Sakr, F., Bellotti, F., Berta, R., De Gloria, A.: Machine Learning on Mainstream Microcontrollers. Sensors. 20, 2638 (2020). https://doi.org/10.3390/s20092638.
16. Wang, X., Magno, M., Cavigelli, L., Benini, L.: FANN-on-MCU: An Open-Source Toolkit for Energy-Efficient Neural Network Inference at the Edge of the Internet of Things. IEEE Internet of Things Journal. 7, 4403–4417 (2020). https://doi.org/10.1109/JIOT.2020.2976702.
17. Edge Impulse, https://edgeimpulse.com/, last accessed 2023/10/10.
18. Sensing and Machine Learning for Automotive Perception: A Review, https://ieeexplore.ieee.org/abstract/document/10089400, last accessed 2024/07/03.
19. Zou, M., Yu, J., Lv, Y., Lu, B., Chi, W., Sun, L.: A Novel Day-to-Night Obstacle Detection Method for Excavators Based on Image Enhancement and Multisensor Fusion. IEEE Sensors Journal. 23, 10825–10835 (2023). https://doi.org/10.1109/JSEN.2023.3254588.
20. Pleterski, J., Škulj, G., Esnault, C., Puc, J., Vrabič, R., Podržaj, P.: Miniature Mobile Robot Detection using an Ultra-Low Resolution Time-of-Flight Sensor. IEEE Transactions on Instrumentation and Measurement. 1–1 (2023). https://doi.org/10.1109/TIM.2023.3318710.
21. Chand, P., Lal, S.: Vision-Based Detection and Classification of Used Electronic Parts. Sensors. 22, 9079 (2022). https://doi.org/10.3390/s22239079.
22. Zhu, S., Voigt, T., Perez-Ramirez, D.F., Eriksson, J.: A Low-resolution infrared thermal dataset and potential privacy-preserving applications. In: Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems. pp. 552–555. Association for Computing Machinery, New York, NY, USA (2021). https://doi.org/10.1145/3485730.3493692.
23. Yeduri, S.R., Breland, D.S., Skriubakken, S.B., Pandey, O.J., Cenkeramaddi, L.R.: Low resolution thermal imaging dataset of sign language digits. Data Brief. 41, 107977 (2022). https://doi.org/10.1016/j.dib.2022.107977.
24. Breland, D.S., Skriubakken, S.B., Dayal, A., Jha, A., Yalavarthy, P.K., Cenkeramaddi, L.R.: Deep Learning-Based Sign Language Digits Recognition From Thermal Images With Edge Computing System. IEEE Sensors Journal. 21, 10445–10453 (2021). https://doi.org/10.1109/JSEN.2021.3061608.