TinyML for Small Microcontrollers

César A. Estrebou[1][0000−0001−5926−8827], Marcos D. Saavedra[2], Federico Adra[2], and Martín Fleming[2]

[1] Instituto de Investigación en Informática LIDI, Facultad de Informática, Universidad Nacional de La Plata
{cesarest}@lidi.info.unlp.edu.ar
[2] Facultad de Informática, Universidad Nacional de La Plata

Abstract. This paper describes the progress made in the context of a research and development project on machine learning techniques and algorithms applied to small microcontrollers. The beginning of the development of EmbedIA, a machine learning framework for microcontrollers, is presented. The experiments carried out comparing the proposed framework with other similar frameworks such as *Tensorflow Lite Micro*, *µTensor* and *EloquentTinyML* show an important advantage with respect to memory and inference time required by small microcontrollers.

Keywords: Machine Learning · Embedded Systems · Microcontrollers · IoT · Convolutional Neural Networks · TinyML

1 Introduction

While machine learning is a term that encompasses many different approaches to solving problems, TinyML is a subset of machine learning that refers specifically to the application of machine learning on resource-constrained devices like microcontrollers.

In the past few years, the world has been experiencing a real proliferation of new smart devices. One of the most interesting aspects of these is that, for the first time, they are capable of running machine learning models on the device itself. One of the main causes of this evolution is the paradigm shift where all the information that was sent to the cloud for processing, began to be processed at the edge to avoid problems [5, 10] related to bandwidth, response delays, high computational and storage costs, higher energy consumption, among others.

However, machine learning is a complicated discipline and making it work on small devices creates more than interesting challenges. This is why TinyML has become a popular area of research and development, where machine learning and in particular deep learning have the potential to provide powerful solutions [9] as long as they can be adapted to devices with limited hardware resources.

In this context, in 2021 we initiated this research and development project that aims to document, study and implement traditional machine learning techniques adapted to small devices. In this paper we present the progress made during the course of the past year.

2 Software and Hardware for TinyML

2.1 TinyML Microcontrollers

From the hardware point of view, the term TinyML is associated with relatively powerful devices, with significant memory and computing capabilities for what is considered a ”traditional” microcontroller. On the other hand, TinyML is closely related to the IoT area, so it is usually considered only devices with wireless communication.

Within the project, all microcontrollers with a minimum capability to run machine learning algorithms are included for experimentation. The choice of models is based on aspects such as local availability, low cost, low/medium computational capacity and availability of open source software for application development. Regarding connectivity, both IoT and non- IoT devices are considered, since from a machine learning point of view there are many popular devices with interesting hardware capabilities without this feature.

Currently, the project has several development boards for testing different implementations of machine learning algorithms. The table 1 shows the microcontrollers that have been tested as well as their technical characteristics.

Table 1: Relevant technical features of the MCUs used in the project.

Board	MCU	Cores	Clock (Mhz)	Memory (KiB)			FPU	Connectivity
				Bits	Data	Prog.		
Arduino Mega	ATmega2560	1	16	8	8	256	No	No
Stm32f103c8t6	Arm Cortex-M3	1	72	32	20	64	No	No
Stm32f411ceu6	Arm Cortex-M4	1	100	32	128	512	Si	No
NodeMCU	Tensilica L106	1	80	32	80	512	No	Wi-Fi
ESP32-WROOM	Xtensa LX6	2	160	32	520	448	Si	Wi-Fi+BT
Raspberry Pi Pico	RP2040	2	133	32	264	2048	No	No

2.2 On-Line Platforms and Open Source Frameworks

There are a variety of on-line platforms (*AlwaysAI*, *Edge Impulse*, *Qeexo*, *Cartesiam.AI* and *OctoML*, among others) that significantly simplify the development and deployment of machine learning applications on microcontrollers. Some perform automatic exploration of solutions for data, others allow the configuration of models from data and others optimize models for deployment on microcontrollers.

Regarding open source libraries and frameworks, there are several alternatives but few of them provide support for neural networks and even fewer support convolutional neural networks. Among those analyzed within the project, the following can be mentioned *EloquentTinyML* [3], *Tensorflow Lite Micro*[4] [1], *µTensor* [5], *EdgeML* [2] and *CMSIS-NN* [7]. In general, they all require experienced users with respect to the generation of machine learning models as well as the microcontroller development platform.

Both online platforms and open source frameworks/libraries are mostly developed and optimized for specific architectures such as *ARM Cortex-M*, for 32-bit architecture or microcontrollers with support for FPU, DSP and/or SIMD instructions. This excludes devices with different architectures or without hardware for specialized mathematical computing. Another limitation that these libraries usually have is that they are developed for C++ 11 and rely on heavy software architectures, object-based with inheritance and polymorphisms that increase the size of the programs and slow down algorithm inference time. This approach may be feasible for microcontrollers with good memory size and hardware resources that accelerate mathematical computing, but it is unsuitable for microcontrollers with low computing power and limited hardware resources.

2.3 Development of a New Open Source Framework

As mentioned in the section 2.2 the online platforms and frameworks have important limitations. In this context, it was decided to start the development of EmbedIA, an open source framework to implement machine learning solutions on microcontrollers with important hardware limitations. In this first stage it was decided to focus on the implementation of neural networks, giving priority to the layers and functions of convolutional neural networks.

EmbedIA is implemented in C, C++ and Arduino compatible code so that it can be compiled on any platform that supports these programming languages. It provides functionalities to perform inference and debugging of the models from the microcontroller. Currently, it supports different neural network layers and activation functions including convolutional, depthwise, binary, pooling, flatten, fully connected, batch normalization, ReLU, sigmoid and softmax. It integrates optimizations for 32-bit, 16-bit and 8-bit fixed-point arithmetic. This reduces program size and RAM usage and speeds up inference time on microcontrollers without floating-point support.

To improve the performance of data memory usage, a swap buffer is implemented to minimize the amount of dynamic memory requirements and avoid fragmentation, which is indispensable for those microcontrollers that do not implement good memory management.

In addition, a conversion tool is provided to transform a model generated with *Tensorflow/Keras* to its equivalent in C code. It allows to generate a C, C++ or Arduino project that includes the functions to perform the inference on the converted model, possibility to use fixed point or floating point and debugging functions for the model.

[3] https://github.com/eloquentarduino/EloquentTinyML
[4] https://www.tensorflow.org/lite
[5] https://github.com/uTensor/uTensor

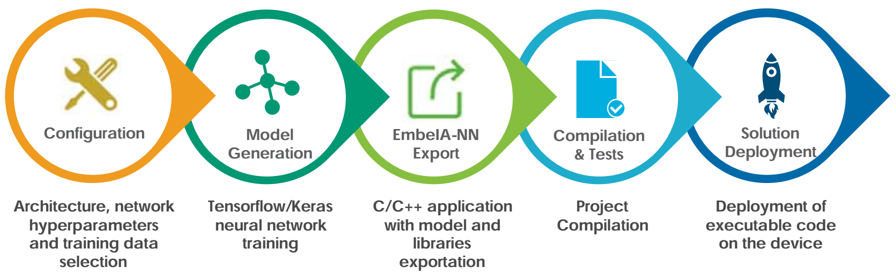
Configuration
Model Generation
EmbeIA-NN Export
Compilation & Tests
Solution Deployment
Architecture, network hyperparameters and training data selection
Tensorflow/Keras neural network training
C/C++ application with model and libraries exportation
Project Compilation
Deployment of executable code on the device

Fig. 1: Steps of development process with the EmbedIA Framework.

3 Experiments Performed and Results Obtained

A series of experiments have been performed on several convolutional neural network [6] (CNN or ConvNet) models to compare the capabilities of EmbedIA with respect to other frameworks/libraries. Initially, we considered to test the performance of *Tensorflow Lite Micro*, *EloquentTinyML*, *CMSIS-NN* and *EdgeML* but, unfortunately, the latter two do not have support for the microcontrollers used.

Among the most relevant results we can highlight [3, 4] one where a convolutional neural network model was used to recognize a total of 36 classes associated with images of 26 letters and 10 handwritten digits. The four Embedia-NN implementations (8-bit, 16-bit and 32-bit fixed-point and floating-point) were compared with other similar frameworks/libraries on microcontrollers of different features, measuring data and program memory size required and inference time consumed. It was found that the four variants of the proposed framework clearly outperformed the implementations of *μTensor*, *Google Tensorflow Lite Micro* and *Eloquent TinyML*. In particular the 16-bit fixed-point implementation achieves, on average, a 5 to 10 times improvement in inference time, about 3 times the data memory requirements and 3 to 7 times the program memory requirements.

As part of the exploration of EmbedIA's capabilities we can mention the development of two interesting models that run on different microcontrollers. The first model runs on an 8-bit *Atmega 2560* with 8 Kib of RAM and recognizes 26 handwritten characters. The second runs on a *Tensilica Xtensa LX6* and recognizes 6 different voice commands. It should be noted that both models could not be used in other frameworks because they exceeded the amount of memory of the microcontrollers.

4 Final Comments

This paper has presented the progress of the research and development project on machine learning applied to microcontrollers with significant hardware limitations. It has started the development of EmbedIA, a machine learning framework for microcontrollers, which at this stage is focused on the implementation of neural network algorithms. So far, comparative performance tests with other frameworks show the potential of this new framework.

Currently, the implementation of convolutional layers for binary neural networks is being finalized and experiments are being prepared to compare them with the implementation of traditional convolutional networks. Different convolutional models are also being developed and tested for human detection on microcontrollers with a small camera.

In the future, it is planned to implement a convolutional layered optimization using an optimization called *patch-based interference* [8] which minimizes about 8 times the peak RAM requirement.

References

1. David, R., Duke, J., Jain, A., Reddi, V.J., Jeffries, N., Li, J., Kreeger, N., Nappier, I., Natraj, M., Regev, S., Rhodes, R., Wang, T., Warden, P.: Tensorflow lite micro: Embedded machine learning on tinyml systems (2021)
2. Dennis, Don Kurian and Gopinath, Sridhar and Gupta, Chirag and Kumar, Ashish and Kusupati, Aditya and Patil, Shishir G and Simhadri, Harsha Vardhan: EdgeML: Machine Learning for resource-constrained edge devices, https://github.com/Microsoft/EdgeML
3. Estrebou, C.A., Fleming, M., Saavedra, M.D., Adra, F.: A neural network framework for small microcontrollers. In: Proceedings of the XXVII Argentinean Congress of Computer Science. pp. 51–60 (2021)
4. Estrebou, C.A., Fleming, M., Saavedra, M.D., Adra, F., De Giusti, A.E.: Lightweight convolutional neural networks framework for really small tinyml devices. In: Narváez, F.R., Proaño, J., Morillo, P., Vallejo, D., González Montoya, D., Díaz, G.M. (eds.) Smart Technologies, Systems and Applications. pp. 3–16. Springer International Publishing, Cham (2022)
5. Farhan, L., Kharel, R., Kaiwartya, O., Quiroz-Castellanos, M., Alissa, A., Abdulsalam, M.: A concise review on internet of things (iot) -problems, challenges and opportunities. In: 2018 11th International Symposium on Communication Systems, Networks Digital Signal Processing (CSNDSP). pp. 1–6 (July 2018). https://doi.org/10.1109/CSNDSP.2018.8471762
6. Goodfellow, I.J., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge, MA, USA (2016), http://www.deeplearningbook.org
7. Lai, L., Suda, N., Chandra, V.: Cmsis-nn: Efficient neural network kernels for arm cortex-m cpus (2018)
8. Lin, J., Chen, W.M., Cai, H., Gan, C., Han, S.: Mcunetv2: Memory-efficient patch-based inference for tiny deep learning. In: Annual Conference on Neural Information Processing Systems (NeurIPS) (2021)
9. Sharma, K., Nandal, R.: A literature study on machine learning fusion with iot. In: 2019 3rd International Conference on Trends in Electronics and Informatics (ICOEI). pp. 1440–1445 (April 2019). https://doi.org/10.1109/ICOEI.2019.8862656
10. Shekhar, S., Gokhale, A.: Dynamic resource management across cloud-edge resources for performance-sensitive applications. In: 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. pp. 707–710 (May 2017). https://doi.org/10.1109/CCGRID.2017.120