TinyML an IoT approach to Machine Learning

Leonardo Serilli, Valerio Lionetti
November 6, 2021

Abstract

The **TinyML** paradigm proposes to merge Machine Learning based algorithms within small objects powered by **Microcontroller Units** (MCUs) . The world has over 250 billion microcontrollers (IC Insights, 2020), with strong growth projected over coming years, providing them with ML capabilities would bring a wide range of application never seen before. The scope of this work is to provide a survey of **opportunities**, **challenges**, **applications**, **prospective** and current **frameworks** available at the state of the art within this novel vibrating field.

1 Introduction

In past decades data production has experienced unprecedented growth, Machine Learning has the opportunity to spread, revolutionizing our way of understanding the world by enabling the inference of valuable information and knowledge from huge amount of data, so far invisible to human eye. Until now it has exploited the data processing and storage capabilities provided by **cloud** technologies these large neural network models focus primarily on accuracy and speed, as they have access to unlimited computational resources and memory.

The **TinyML paradigm** proposes a different approach, it wants to integrate Machine Learning based mechanisms within small objects powered by **Microcontroller Units** (MCUs) to have a wide range of applications and services that do not need the omnipresent processing support from the cloud, which is **power consuming and involves data security and privacy risks.** MCUs instead are tiny processing entities usually embedded in daily-use appliances that can be massively produced at inexpensive cost. The spread of this **frugal objects** in our lives due to the emergence of the Internet of Things (IoT) is skyrocketing, so providing them with cognitive skills bring opportunities for developing a huge number of collective intelligence applications.

The contributions of this work are: a wide discussion of the **TinyML concept as well as advantages and challenges** that it brings, A view through the **applications** that can benefit from it and a survey of some **available frameworks** for integrating ML within MCUs.

2 methodology used for research papers

For the purpose of our work we looked for papers in dedicated website as **Research Gate**[16], **Ieeexplore**[17] and **Cornell University**[18], searching for keyword related to TinyML topic. By the lack of material we went also through papers referenced by the collected ones.

3 State of the art

There are three kind of tinyML framework:

1. the first is about **Adapting already trained models** to the restrictions of MCUs. They use to take models generated by well-known ML libraries such as **TensorFlow, Scikit-Learn, or PyTorch** to porting their code to be executed on MCUs with they constrained resources.

2. The second approach tries to **integrate ML libraries within MCUs**, in order to generate models from data retrieved by the own device, enabling the implementation of unsupervised learning algorithms and reducing the transmission of data.

3. the last approach is dedicated to **implement co-processors that acts as the main computing unit on ML-specific tasks** on MCUs. This strategy permits to enhance the computation performance, but it is the less common approach as the price and complexity of devices.

Google has launched **TensorFlow Lite** [9] (and **TensorFlow Lite Micro** [3]), which includes a set of tools to adapt Tensor Flow models aiming a number of algorithms from the **NN family** to making them runnable in embedded Linux, MCUs or Mobiles. The optimized code, provided in C++ requires 32-bit processors to be run. It has been successfully tested in devices with ARM Cortex-Mseries processors, e.g., Arduino nano, and other architectures such as ESP32.
Another related Google's initiative has been the **adaptation of the GO programming language** to microcontrollers.[12]
Microsoft has also contributed to the TinyML scene by releasing its open source **Embedded Learning Library (ELL)** [10], this framework permits to design and deploy pre-trained ML models onto constrained platforms, e.g., ARM Cortex-A and Cortex-M architectures such as Arduino, Raspberry Pi.
The Fraunhofer Institute for Microelectronic Circuits and Systems (IMS) has developed the **Artificial Intelligence for Embedded Systems (AIfES)**[12] library, which is a C-based and platform-independent tool for generating NNs compatible with a range of MCUs, e.g., Arduino UNO, ATMega32U4, or STM32 F4 Series.
MicroML[14] is a project that permits to port Support

Vector Machine (SVM) and Relevance Vector Machine (RVM) algorithms to C code that can run in a number of MCUs, e.g., Arduino, ESP8266, ESP32, and others with C support. It interoperates with the widely-used Scikit-learn toolkit and transforms the models generated by this library in order to be executed on 8-bit microcontrollers with 2 kb of RAM. Also compatible with Scikit-learn, **sklearn-porter**[15] permits to transpile trained estimators to C, Java, JavaScript, GO, Ruby, and PHP.
Emlearn[13] is a framework that produces portable C99 code from Python libraries such as Scikit-learn or Keras. This library is compatible with generated models of a range of algorithms, e.g., RF, decision trees, Naive Bayes, or linear models, and has been tested in different platforms, namely, AVR Atmega, ESP8266 and Linux.

4 Analysis and Discussion

The advantages of a TinyML architecture are of course **energy efficiency and low cost** which is derived from the spread of iot and reprogrammability of devices. another advantage is the system reliability and data security because too much communication is energy consuming, so it's better to perform computation on end systems having fewer transmission which comports more **reliability** because we can use more sophisticated solutions for **security** by the fact that we transmit fewer data; this will of course reduce **latency**.

On the other hand, identified challenges are related to the **heterogeneity** as there are much different frugal object running **different protocol**;. from the producer prospective, those object must support a straight ML integration, **local data security** (they don't want an hacker have easy access to data stored by exploiting the hardware vulnerabilities as in the Miriai Botnet attack of 2016); from the **cloud prospective**, they should be able to **support api for an efficent exchange of data** between devices, and guarantee **Reliability** in case of failure.

Continued progress is limited by the lack of a widely accepted **benchmark** for these systems because Benchmarking allows us to measure and **systematically compare, evaluate, and improve the performance of systems** and is therefore fundamental to a field reaching maturity. [1] To make an example: a workaround for the problem of power consumption is **MLPerf Tiny**[2], the first industry-standard benchmark suite for ultra-low-power tiny machine learning systems.
A problem that general NN model must face is that different users have different patterns, and the model needs to be updated with field data on the fly. A **tinyOL**[6] system can be attached to an existing NN in MCUs as a new layer or replace a specific layer in the NN. It will process data stream to accomodate many MCUs constraints line data storage or computation overhead, and consequentially the battery consumption too.

The evolution of tinyML paradigm will bring a wide range of useful application, for example **wearables**, such as fitness monitors or "smart" watches are not that smart by now, as they rely on a Bluetooth-coupled smartphone which assumes the majority of processing tasks. Integrating ML within frugal objects permits their **independence from a master device**, which will notably enrich current applications landscape. It can improve **Health monitoring applications**, **augmented reality**, **real-time voice recognition**, **language translation**, **Smart cities** or **cognitive buildings**, current **IoT-based monitoring** and **surveillance systems**, **Smart Agriculture and Farming**, **Industry** and at least **cooperative-Intelligent Transportation Systems** (C-ITS) will be enriched.

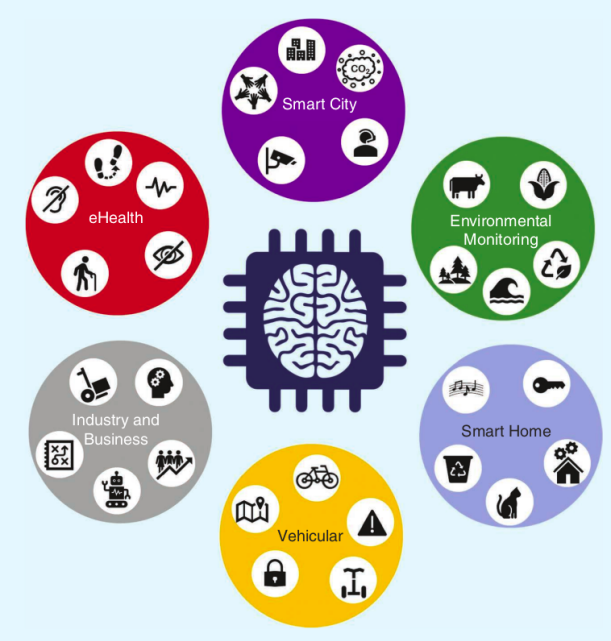

Figure 1: tinyML applications

5 Conclusion

The merging of ML and IoT ecosystem is capable of skyrocketing it's computational capabilities and possible applications. This work has provides a general look at opportunities and challenges in the fast growing paradigm of TinyML, so the application of ML algorithms to a distributed network of MCUs, to bring their smart capabilities at a level never seen before. Have been illustrated constraints within those frugal object must work, and the effort related to the evolution of communication technologies and protocols needed to provide an ubiquitous connectivity with the advantages they brings. The diversity of hardware and software needs to be addressed through benchmarking policies.

References

[1] C. R. Banbury et al., "Benchmarking TinyML Systems: Challenges and Direction," Mar. 2020, Accessed: Nov. 05, 2021. [Online]. Available: https://arxiv.org/abs/2003.04821v4

[2] C. Banbury et al., "MLPerf Tiny Benchmark," arXiv:2106.07597 [cs], Aug. 2021, Accessed: Nov. 05, 2021. [Online]. Available: http://arxiv.org/abs/2106.07597

[3] R. David et al., "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," Oct. 2020, Accessed: Nov. 05, 2021. [Online]. Available: https://arxiv.org/abs/2010.08678v3

[4] S. Soro, "TinyML for Ubiquitous Edge AI," Feb. 2021, Accessed: Nov. 05, 2021. [Online]. Available: https://arxiv.org/abs/2102.01255v1

[5] R. Sanchez-Iborra and A. F. Skarmeta, "TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities," IEEE Circuits and Systems Magazine, vol. 20, no. 3, pp. 4–18, 2020, doi: 10.1109/MCAS.2020.3005467.

[6] H. Ren, D. Anicic, and T. Runkler, "TinyOL: TinyML with Online-Learning on Microcontrollers," Mar. 2021, Accessed: Nov. 05, 2021. [Online]. Available: https://arxiv.org/abs/2103.08295v3

[7] B. Garbinato, R. Guerraoui, J. Hulaas, M. Monod, and J. H. Spring, "Pervasive Computing with Frugal Objects," in 21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07), May 2007, vol. 2, pp. 13–18. doi: 10.1109/AINAW.2007.287.

[8] H. Doyu, R. Morabito, and J. Höller, Bringing Machine Learning to the Deepest IoT Edge with TinyML as-a-Service*. 2020.

[9] "TensorFlow Lite | ML for Mobile and Edge Devices," TensorFlow. https://www.tensorflow.org/lite (accessed Nov. 06, 2021).

[10] "The Embedded Learning Library - Embedded Learning Library (ELL)." https://microsoft.github.io/ELL/ (accessed Nov. 06, 2021).

[11] "TinyGo Home." https://tinygo.org/ (accessed Nov. 06, 2021).

[12] "Artificial Intelligence for Embedded Systems - Fraunhofer IMS," Fraunhofer Institute for Microelectronic Circuits and Systems. https://www.ims.fraunhofer.de/en/Business-Unit/Industry/Industrial-AI/Artificial-Intelligence-for-Embedded-Systems-AIfES.html (accessed Nov. 06, 2021).

[13] J. Nordby, emlearn: Machine Learning inference engine for Microcontrollers and Embedded Devices. Zenodo, 2019. doi: 10.5281/zenodo.2589394.

[14] eloquentarduino, Introducing MicroML. 2021. Accessed: Nov. 06, 2021. [Online]. Available: https://github.com/eloquentarduino/micromlgen

[15] D. Morawiec, sklearn-porter. 2021. Accessed: Nov. 06, 2021. [Online]. Available: https://github.com/nok/sklearn-porter

[16] "ResearchGate | Find and share research," ResearchGate. https://www.researchgate.net/ (accessed Nov. 06, 2021).

[17] "IEEE Xplore." https://ieeexplore.ieee.org/Xplore/home.jsp (accessed Nov. 06, 2021).

[18] "Cornell University." https://www.cornell.edu/ (accessed Nov. 06, 2021).