A TinyMLaaS Ecosystem for Machine Learning in IoT: Overview and Research Challenges

Conference Paper · April 2021

DOI: 10.1109/VLSI-DAT52063.2021.9427352

3 authors, including:

Roberto Morabito
Ericsson
35 PUBLICATIONS **1,447** CITATIONS

SEE PROFILE

A TinyMLaaS Ecosystem for Machine Learning in IoT: Overview and Research Challenges

Hiroshi Doyu*, Roberto Morabito*, Martina Brachmann†
*Ericsson Research, Finland
†Ericsson Research, Sweden
{firstname.lastname}@ericsson.com

***Abstract*—Tiny Machine Learning (TinyML) is an emerging concept that concerns the execution of ML tasks on very constrained IoT devices. Although TinyML has generated a strong R&D interest around it, various challenges limit its effective execution in the constrained devices world, with the result of slowing down the development of a complete ecosystem around it. TinyML as-a-Service (TinyMLaaS) aims to fill the gap in this respect, with the definition of a set of guidelines that can enable an easier democratization of TinyML. In this paper, we describe how the "as-a-Service" model is bound to TinyML, by providing an overview of our concept and introducing the design requirements and building blocks that can make TinyMLaaS reality.**

I. Introduction

It has been predicted that there will be 26.9 billion connected devices by 2026, as part of the so-called Internet of Things (IoT) [1]. Computing and networking infrastructure such as cloud, fog and edge computing – which are classified depending on their resources and capabilities – together with IoT, machine learning (ML) has become the key technology enabler for many industries and domains such as automotive [2], smart cities [3], health care [4], and smart factories [5].

In the context of constrained IoT, the devices have considerably less capabilities than edge devices in terms of processing power and memory. In addition, they are also limited in their power resources as they often run using small batteries or energy-harvesting technologies. However, a recent new technology trend has emerged in the IoT landscape: ML in *constrained devices*. Combining ML and constrained devices is envisioned to have a great impact on the current IoT application landscape, in areas such as e-Health, smart agriculture and farming, production, and smart home [6] by involving tiny and energy-efficient always-on devices [7].

The concept that allows to fit ML models into constrained devices, without compromising their energy efficiency, is called *Tiny Machine Learning (TinyML)* [7]. TinyML encompasses very resource-constrained hardware, software, ML algorithms, compilers, and tools to squeeze a ML model into a few kilobyte of memory [8]. As these hardware platforms, compilers, and software tools are often tied to a specific vendor, the lack of interoperability among different solutions may undermine the other benefits deriving by the use of TinyML. To cope with this issue, we have recently proposed *TinyML as-a-Service (TinyMLaaS)*, a cloud- or edge- based service that simplifies the deployment of ML models into constrained devices and guarantees the desired interoperability [9].

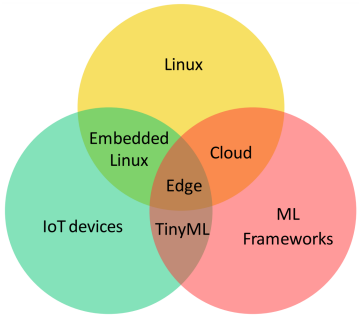

Fig. 1: The overlapping of technological areas and enablers. (This illustration is a slightly modified version of Fig. 1 in [10]).

In this paper, we extend our TinyMLaaS paradigm by identifying the steps that are needed to make TinyMLaaS interoperable with other peer systems, bearing in mind the ultimate goal of building a full ecosystem around it. We also highlight what are the key technical challenges to address for reaching this goal, as well identifying what are the most prominent research areas to investigate in order to bring significant benefits to the entire TinyML ecosystem.

II. Background

Before introducing TinyMLaaS and our vision of an ecosystem around it, this section provides the fundamental notions of TinyML and ML in constrained devices, useful to understand the remainder of the paper.

A. What is TinyML?

We have defined TinyML in the introduction simply as the intersection between ML and constrained IoT devices and provide now a more technological point of view of TinyML in the following. Fig. 1 illustrates technology areas and enablers as circles and their common ground as intersections. For example, the world of *Embedded Linux* can be considered as rendezvous point between *Linux* and *IoT devices*, thus also acknowledging that *IoT device* capabilities stretch across the *edge*. *ML* was originally started, developed and evolved in the *cloud* with resource demanding software frameworks and large hardware resources such as graphics processing units (GPUs) and tensor processing units (TPUs). Now the computation is moving into the *edge* to run *ML* on less powerful computing resources but still with ML-supporting embedded OSs. *TinyML* represents

the connection point between *constrained IoT devices* and *ML* without any resource rich OS (e.g. Linux).

B. Machine Learning on the Tiniest End Devices

While a few years ago large data centers were needed for the execution of ML tasks, current ML models and system requirements to run these models have become very small so that they can even fit into tiny microcontrollers (MCUs) [11] – i.e., the unit in a constrained IoT devices that governs and runs the program. This possibility opens up new applications for ML using constrained IoT devices, including always-on inference for wake word detection [11], [12], human activity recognition [13], [14] and acoustic-anomaly detection [15].

Constrained devices have limited capabilities regarding their memory, processing power, and they are often battery-operated or utilize energy harvesting [16]. Constrained hardware platforms able to run ML are currently equipped at most with a 32-bit MCU, ∼500 KB of static RAM, a few megabyte of Flash memory and consume only a few milliwatts of power.

The limited capabilities of constrained devices entail several challenges for the execution of ML tasks. For example, when referring to ML training operations, these tasks are typically executed using dedicated ML frameworks designed to run efficiently on powerful machines often combined with dedicated hardware like GPUs or TPUs and with extensive math-libraries. Models created with these frameworks cannot be directly ported to constrained devices due to the lower hardware and software capabilities respect more powerful computing systems. As a consequence, the models need to be manually optimized during a labour-intense, error-prone and often performance-degrading process [17]. This includes deletion of complex computations that constrained devices are not able to perform, quantizing floating point operations to 8-bit integers and pruning less important parameters. Another challenge derives from the fact that IoT devices are designed, in some cases, for running custom-made applications. In fact, the peripherals and sensors that a constrained device is equipped with typically determine the application that runs on it and thus the model of the ML task used by the application itself. This implies that a model optimized to run on a specific constrained hardware platform may not run in other platforms due to different hardware capabilities, peripherals, sensors, and the use of vendor-specific libraries. The latter aspect and the effort it takes to port models from one platform to another result in a deficiency of ML models for constrained devices.

Despite the challenges of running ML directly on constrained devices, there are several benefits compared to running it in more resourceful computing environments [18]. A major advantage of running ML applications in constrained devices is the wide range of possible use cases it enables, as well as the *low costs* of the hardware, even when deploying large numbers of devices [6].

Ensuring a higher level of *data security* and *user privacy* is another advantage determined by the possibility of running ML tasks directly in the device [6]. Data can be sensitive and contain personal or business critical information. Therefore, enabling the possibility of making the data staying on-premise and there being processed, eliminates the need of sending raw data via the Internet to cloud-based services for further processing. This aspect also aligns to other emerging ML techniques, such as federated learning [19], which aim to address the above mentioned security and privacy issues is a similar fashion.

From the computer networking perspective, on-device ML allows to also reduce the amount of data offloading between the constrained devices and the cloud, consequently reducing the network bandwidth requirements. As a consequence, *energy efficient* communication technologies like NB-IoT or LTE Cat-M1 [20] can be used, in case communication with external services is required by the application. Energy efficiency is especially important considering that the constrained devices are most probably battery-operated and equipped with low-power MCUs that use up only a few milliwatts. Although MCUs holds a lower processing speed compared to more resourceful devices, ML predictions can possibly be attained faster, as the device must not offload the data, as well as wait for the computation of the ML prediction in the cloud and the transmission of the result to the device again.

III. TinyMLaaS as Machine Learning ecosystem enabler

In this section, we introduce TinyMLaaS. We explain how TinyMLaaS connects to TinyML and why we believe that TinyMLaaS can allow to democratize ML in the context of constrained IoT.

A. What is TinyMLaaS?

With TinyMLaaS, we aim to build a higher-level abstraction of TinyML software that is as hardware and software agnostic as possible. Furthermore, we will do this in an "as-a-Service" fashion. Why? The advantages of using specialized hardware for ML must be balanced with the use of dedicated ML compilers that adapt a certain ML model to the targeted hardware platform. This hardware and associated compilers' heterogeneity (i.e. application of various kinds of special hardware) generates additional fragmentation. It also offers poor flexibility against the possibility of easily switching hardware context due to the need to re-compile the ML inference model for the targeted device. ML compilers are very powerful tools and we do not want to disregard their important role in the ML ecosystem. This software abstraction is the foundation of TinyMLaaS – a cloud or edge service designed to host a wide set of ML compilers. It is the job of these compilers to convert a specific ML inference model into the appropriate format for being executed in the served device. Fig.2 shows how the different TinyMLaaS components interact between each other. The numbered purple circles indicate the order in which the different sub-processes take place. To tailor an ML inference model for running in a specific device, TinyMLaaS needs to gather some information about the device itself, such as CPU type, RAM and ROM size, available peripherals, underlying software, and the correct inference model to process. The TinyMLaaS backend will select the most suitable ML

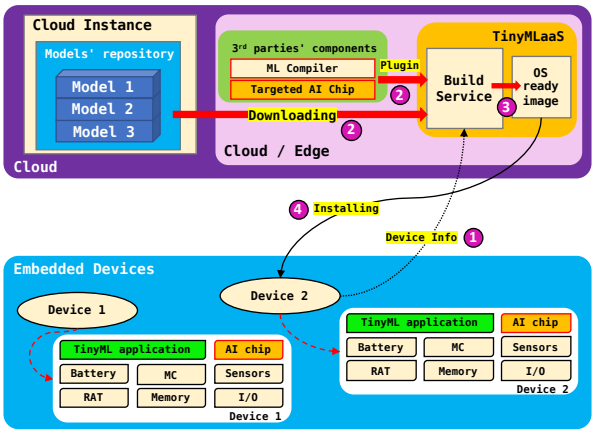

Fig. 2: TinyML as-a-Service block diagram

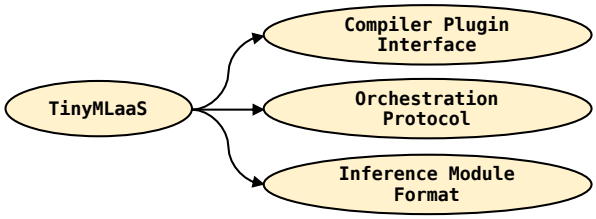

Fig. 3: TinyMLaaS Main Components.

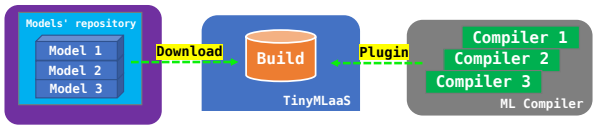

Fig. 4: TinyMLaaS Compiler Plugin Interface.

compiler and generate the compiled ML inference module on the basis of the above parameters. The generated ML inference module is then downloaded and installed on the designated device. Our example implementation of TinyMLaaS uses OMA LightweightM2M protocol [21] also to benefit from its Firmware-over-the-air (FOTA) and Software-over-the-air (SOTA) update capabilities. The integration between LwM2M and IPSO Objects is harnessed, with the aim of using a standardized model when end-devices and a TinyMLaaS instance exchange device characteristics information, including the representation of ML algorithm installed.

B. Why Ecosystem Matters

As stated in previous section, building and compiling ML inference models in a seamless way and agnostically of the underlying hardware platform is the desirable outcome for each ML application developer and TinyMLaaS represents an optimal solution for achieving this. However, the impact generated by TinyMLaaS can be even more significant if standardized mechanisms are introduced in order to allow different TinyMLaaS-based systems talk to each other. The interoperability among such systems could generate advantages both from the technical and business point of view, as it would allow chip manufacturers and software developers to impact the ML landscape with a common underlying architecture and, by consequence, encouraging also a fairer competition between small and medium-sized enterprises (SMEs) and tech giants. To this extent, the TinyMLaaS paradigm needs to be accommodated within a wider context that includes a functional ecosystem built around it. The goal of this section is to outline what are the additional components needed to make TinyMLaaS interoperable with other peer systems.

As introduced in [9], we identified three main components – namely compiler plugin interface, orchestration protocol, and inference module format – that, if standardized, would represent the core of a TinyMLaaS ecosystem (Fig. 3) and would shorten the way towards the desired interoperability. The role played by these three ecosystem's elements is explained in detail below.

C. Compiler Plugin Interface

The *compiler plugin interface* represents a sort of middleware abstraction that operates in order to consistently bridge the requests submitted to the TinyMLaaS front-end with the compiling parameters requested by the TinyMLaaS back-end. The front-end is designed in order to receive and accept requests from a plethora of heterogenous connected devices (e.g. MCU). Such requests are sent with the purpose of seeking the use of a specific ML compiler. The different ML compilers require different customization parameters before being executed for processing the compilation output. Therefore, we envisioned the need of defining a common standard that makes possible to consistently enclose such parameters among all the ML compilers available in the back-end, without the need to adapt the parameters to the device in use. Fig. 4 depicts such process.

D. Orchestration Protocol

The *orchestration protocol* component plays a key role upon interaction between end-devices and the TinyMLaaS infrastructure, on the basis of well-defined APIs. The functionality provided by this module is two-fold. First, it interacts with the end-devices in order to gather information about their baseline and real-time software and hardware capabilities. This allows to ensure that any request coming from a given device is properly managed by the TinyMLaaS back-end, as in turn will tailor the compiling process exactly according to the information transmitted by the orchestration component. Second, it offers Firmware-over-the-air (FOTA) and Software-over-the-air (SOTA) update capabilities, useful to comply with the devices' requests. As mentioned in the previous section, we rely on LwM2M also as enabling technology for the orchestration protocol because of its suitability to the embedded IoT context and its standardized features in terms of device management, machine-to-machine (M2M) communication and standardized data representation models [21] . In Fig. 5, a simplified illustration of the orchestration process is reported.

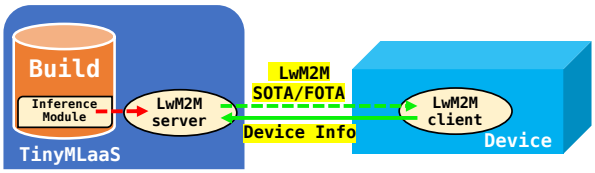

Fig. 5: TinyMLaaS Orchestrator.

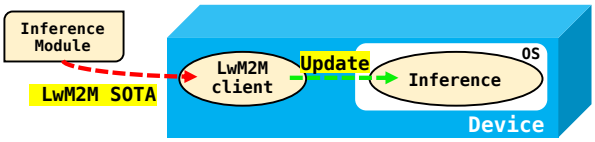

Fig. 6: TinyMLaaS Inference Module.

E. Inference Module Format

The *inference module* role is to provide, through a LwM2M-based SOTA and a predefined representation format, the output processed by the ML compilers. In other words, such module represents the component that make the ML inference application available to the device. Once again, we reiterate the importance of defining a standardized format that allows to represent the functionalities of a wide range of ML compilers and inference applications, as well as the heterogeneous features of multiple types of real-time OS and hardware chipsets. In a similar way to the orchestration protocol case, the output format of such an application is tailored by the ML compiler according to the underlying software and hardware characteristics of the device that is using it. The large number of heterogeneous devices and the lack of a consistent inference format model means that this process remains fragmented. In this respect, we look forward to bigger steps towards a standardization process of the inference format model, so as to ensure an easier ML software portability between devices. Fig. 6 shows the interaction between *inference module* and *device*.

In conclusion of this section, we introduce Fig. 7, which shows a more detailed characterization of the three interfaces introduced earlier. Such representation outlines both additional features and deployment options that are not thoroughly discussed in this paper because of space limitations.

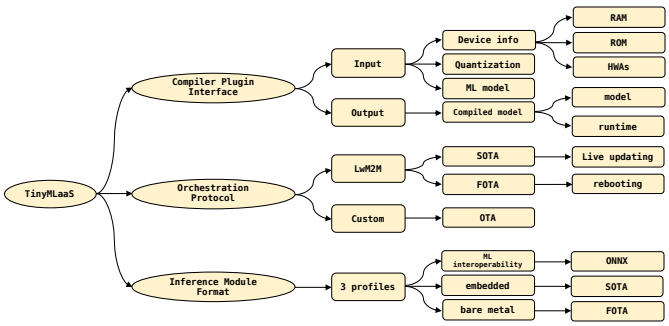

Fig. 7: Detailed Overview of TinyMLaaS Interfaces.

IV. Research Challenges

In this section, we highlight some of the most prominent R&D challenges connected to TinyMLaaS.

Optimizing the trade-off between ML performance and constrained devices resources represents a major requirement for any TinyML-based system. It consists in ensuring the optimal compromise between the ML model size, accuracy and performance and the very limited capabilities and energy efficiency requirements of the MCUs executing these models.

TinyML-based model's splitting. Models' splitting refers to the set of techniques used for executing distributing training operations among multiple devices ensuring the same level of accuracy. Defining analogous the same techniques as they are in TinyML-centric systems is unfeasible. This leads to the need of introducing new model's splitting and parallelism techniques that take into account all the constraints characterizing such scenarios (e.g., limited devices' resources, intermittent connectivity, etc.) and the need of reducing communication overhead and utilizing computing resource efficiently.

Unified ML compiler-centric architecture. The ML ecosystem fragmentation is not an exclusive problem of ML-based IoT systems. The need of standardizing the interactions between heterogeneous components represent a requirement that can be disjointed by the IoT context and reformulated as ML Compiler as-a-Service (MLCaaS). MLCaaS ecosystems can enable an even greater seamless migration of a ML inference services execution between heterogeneous components. However, enabling MLCaaS requires a major effort in the form of open-source initiatives and in the definition of well-defined standards and regulations.

Dynamic computation allocation between end-devices and the edge/cloud. Optimizing offloading and onloading of ML service execution between edge and cloud has lately represented a compelling research challenge with many solutions proposed for it. However, in the context of TinyMLaaS, these already defined policies are unlikely to be sufficient. New algorithms that consider the different computational power and specifications of the processing entities, as well as the need of consistent portability among heterogeneous devices required in the case of local and parallel processing.

Connectivity driven ML inference provisioning. IoT devices can embed multiple radio access technology (RAT) interfaces. In specific scenarios, it becomes essential understanding what is the most adequate communication interface to be used when the ML inference service execution requires device-to-edge or device-to-cloud communication. Developing efficient mechanisms that allow to tackle this challenge must be introduced, taking into account the constrained devices requirements, the characteristics of the data to be sent, etc.

Security and Privacy. System's reliability, data security and privacy represent three inherent advantages deriving by the possibility of performing on-device data processing [6]. However, TinyMLaaS encompasses frequent interactions with edge and cloud facilities. New end-to-end security mechanisms

are required for guaranteeing adequate security standards, but with the strict requirement that such mechanisms must be as lightweight as possible in order to minimize the overhead generated by their usage. Although emerging protocol extensions such as Object Security for Constrained RESTful Environments (OSCORE) [22] seem to be suitable for tackling this challenge, additional studies are needed in order to empirically demonstrate the feasibility of such integration.

V. Conclusion

In this paper, we introduced the concept of TinyMLaaS. We presented what are the motivations, requirements and design principles behind this new paradigm, emphasizing also what are the steps needed in order to build a full ecosystem around it. Finally, we also discussed additional key technical challenges and identified open questions for future research in this area.

References

[1] F. Jejdling, P. Cerwall, A. Lundvall, P. Jonsson, S. Carson, R. Möller, S. Davis, P. Linder, A. Gomroki, A. Zaidi, A. C. P, M. Opsenica, I. Sorlie, S. Elmgren, G. Blennerud, H. Baur, R. Svenningsson, B. Heath, J. Bugel, S. John, and S. Schwartz, "Ericsson Mobility Report," Nov 2020.

[2] M. Syafrudin, G. Alfian, N. L. Fitriyani, and J. Rhee, "Performance Analysis of IoT-Based Sensor, Big Data Processing, and Machine Learning Model for Real-Time Monitoring System in Automotive Manufacturing," *Sensors*, vol. 18, 2018.

[3] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: a survey," *Digital Communications and Networks*, vol. 4, 2018.

[4] S. Durga, M. R. Nag, and E. Daniel, "Survey on Machine Learning and Deep Learning Algorithms used in Internet of Things (IoT) Healthcare," in *Proceedings of the International Conference on Computing Methodologies and Communication (ICCMC)*, 2019.

[5] A. Angelopoulos, E. T. Michailidis, N. Nomikos, P. Trakadas, A. Hatziefremidis, S. Voliotis, and T. Zahariadis, "Tackling Faults in the Industry 4.0 Era—A Survey of Machine-Learning Solutions and Key Aspects," *Sensors*, vol. 20, 2020.

[6] R. Sanchez-Iborra and A. F. Skarmeta, "TinyML-Enabled Frugal Smart Objects: Challenges and Opportunities," *IEEE Circuits and Systems Magazine*, vol. 20, 2020.

[7] R. David, J. Duke, A. Jain, V. J. Reddi, N. Jeffries, J. Li, N. Kreeger, I. Nappier, M. Natraj, S. Regev, R. Rhodes, T. Wang, and P. Warden, "TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems," 2020. [Online]. Available: arXiv:2010.08678v2

[8] C. Banbury, C. Zhou, I. Fedorov, R. M. Navarro, U. Thakker, D. Gope, V. J. Reddi, M. Mattina, and P. N. Whatmough, "MicroNets: Neural Network Architectures for Deploying TinyML Applications on Commodity Microcontrollers," 2020. [Online]. Available: arXiv:2010.11267

[9] H. Doyu and R. Morabito, "How can we democratize machine learning on IoT devices?" 2020. [Online]. Available: https://www.ericsson.com/en/blog/2020/2/how-can-we-democratize-machine-learning-iot-devices

[10] ——, "TinyML as-a-Service: What is it and what does it mean for the IoT Edge?" 2019. [Online]. Available: https://www.ericsson.com/en/blog/2019/12/tinyml-as-a-service-iot-edge

[11] P. Warden and D. Situnayake, *TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-low-power Microcontrollers*. O'Reilly Media, 2020.

[12] Y. Zhang, N. Suda, L. Lai, and V. Chandra, "Hello Edge: Keyword Spotting on Microcontrollers," 2018. [Online]. Available: arXiv:1711.07128

[13] P. M. Scholl, N. Kücükyildiz, and K. van Laerhoven, "When Do You Light a Fire? Capturing Tobacco Use with Situated, Wearable Sensors," in *Proceedings of the ACM International Joint Conference on Pervasive and Ubiquitous Computing*, 2013.

[14] R. Chavarriaga, H. Sagha, A. Calatroni, S. T. Digumarti, G. Tröster, J. D. R. Millán, and D. Roggen, "The Opportunity Challenge: A Benchmark Database for on-Body Sensor-Based Activity Recognition," *Pattern Recognition Letters*, 2013.

[15] E. Rushe and B. M. Namee, "Anomaly Detection in Raw Audio Using Deep Autoregressive Networks," 2019.

[16] C. Bormann, M. Ersue, and A. Keränen, "Terminology for Constrained-Node Networks," RFC 7228, 2014.

[17] M. Sponner, B. Waschneck, and A. Kumar, "Compiler Toolchains for Deep Learning Workloads on Embedded Platforms," in *Proceedings of the International Research Symposium on Tiny Machine Learning (tinyML)*, 2021, to Appear.

[18] H. Doyu and R. Morabito, "TinyML as a Service and the challenges of machine learning at the edge," 2019. [Online]. Available: https://www.ericsson.com/en/blog/2019/12/tinyml-as-a-service

[19] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated Machine Learning: Concept and Applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, 2019.

[20] GSMA, "Mobile IoT in the 5G Future – NB-IoT and LTE-M in the Context of 5G," Tech. Rep., 2018.

[21] "Lightweight M2M (LWM2M)." [Online]. Available: https://omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/

[22] G. Selander, J. Mattsson, F. Palombini, and L. Seitz, "Object security for constrained restful environments (oscore)," *Work in Progress*, 2019.