Toward Secure TinyML on a Standardized AI Architecture

Towards Secure TinyML on a Standardized AI Architecture

Muhammad Yasir Shabir, Gianluca Torta, Andrea Basso, Ferruccio Damiani

Abstract Recently, ML tasks that have been traditionally associated with high-performance CPUs and GPUs, have started to be performed also on highly constrained devices at the far edge. This shift towards the devices, often named TinyML, has many well recognized advantages such as lower bandwidth requirements and energy consumption, cheaper prices, increased privacy, and scalability. However, it also poses serious challenges: first of all, it requires to handle even complex ML tasks with Microcontollers (MCUs) equipped with small memories, low-performance processors, and limited power supply; moreover, TinyML has to face the additional security threats that can specifically affect small devices, that usually have to rely on less support from the hardware and the OS to implement security, and once deployed in the field, can be exposed to physical threats. A first contribution of this work is to provide a thorough review of related literature to help delineate the state-of-the-art and classify existing approaches based on their scope, goals and employed technical solutions. A second contribution is to delineate a research program to advance such state-of-the-art, with a special focus on secure and energy efficient ML applications, in the context of a standardized component-based architecture recently proposed by the MPAI organization, which applies in particular to far edge AI applications.

Muhammad Yasir Shabir
University of Turin, Turin, Italy e-mail: muhammadyasir.shabir@unito.it
[Universiy of Kotli AJK,Pak e-mail: yasir.shabir14@gmail.com]

Gianluca Torta
University of Turin, Turin, Italy e-mail: gianluca.torta@unito.it

Andrea Basso
Synesthesia Innovation, Turin, Italy e-mail: andrea.basso@synesthesia.it

Ferruccio Damiani
University of Turin, Turin, Italy e-mail: ferruccio.damiani@unito.it

1 Introduction

Until very recently, Machine Learning (ML) applications have been hosted mostly on public clouds (or possibly on high performance on-prem systems), with organizations often relying on ready-to-go deployed models from cloud services, e.g., in many industrial applications. However, dependency on cloud-based machine learning services raises several challenges such as huge energy consumption, privacy issues, network and processing latency, and reliability issues [1, 10, 23].

TinyML is a revolutionary technology that allows ML to be run on embedded edge devices that have very limited processing power and memory. This technology has opened up a new realm of possibilities in the IoT world, enabling a variety of innovative applications such as predictive maintenance, anomaly detection, and smart monitoring. TinyML recognizes that the physical world is smarter than the existing scenario , and embedded edge devices can make decisions before seeking help from edge AI or cloud AI. This setting results in improvements such as energy efficiency, better privacy of local data processing, low processing latency, and minimal connectivity dependency.

In the present work, we want to explore TinyML applications, paying particular attention to two aspects that have been given little attention in earlier studies.

First of all, we pay close attention to the TinyML security problem. We investigated several methods and approaches that are used to secure the TinyML models because it is a main challenges in the field of machine learning specially for the memory constrained ML techniques.

Secondly, considering potential applications of TinyML (in fields such as healthcare, agriculture, education, transportation, etc.), we believe it is important to go beyond the typical model of a single ML module executed on a device. We envision TinyML applications that involve many modules, executed in a single or multiple, communicating devices. A relevant initiative that aims to standardize AI-based applications is carried by the *Moving Picture, Audio, and Data Coding by Artificial Intelligence* (MPAI)[1] organization. In particular, The MPAI AIF (AI Framework) specification [19], now an IEEE standard in its version 1.1, defines a framework for designing, implementing and deploying complex AI applications modeled as workflows of data connecting individual modules. The standard version 2.0, now under development, will cover also security functionalities of the AIF.

In the present work, starting from the state-of-the-art, we will delineate challenges and research directions to go from simple and monolithic models to full fledged standard-based, complex, and secure TinyML applications. The paper is structured as follows. In section 2 we will describe the MPAI organization and is initiatives that are most useful for the present discussion. In section 3 we will review several works from the literature, guided by our

[1] https://mpai.community/

focus on TinyML applications and their security. In section 4 we will outline the research directions towards our goals described above. Finally, in section 5 we will give some final remarks and conclude.

2 Background on MPAI

The MPAI community is a non-profit multinational organization with established rules, procedures, statutes and well organized bodies. Its main objective is to create data coding standards for AI applications.

MPAI activities include 15 standard development lines, ranging from audio to health, from autonomous cars to online gaming, and from XR to the metaverse.[2] Of particular interest in the present work, is the definition of a standard *Artificial Intelligence Framework* (MPAI-AIF), which aims at facilitating the construction and automation of mixed artificial intelligence - machine learning - data processing workflows for the application areas currently under consideration by MPAI. It is worth mentioning that some authors of the present paper are members of the MPAI-AIF DC (Development Committee), and that the research proposal described in the next sections partially depends on current and future developments of the framework.

Figure 1 displays the MPAI-AIF Reference Model for V1 of the standard, in which an AIF Implementation hosts the execution of *AI Workflows* (AIW), which are made up of fundamental processing units known as *AI Modules* (AIM).

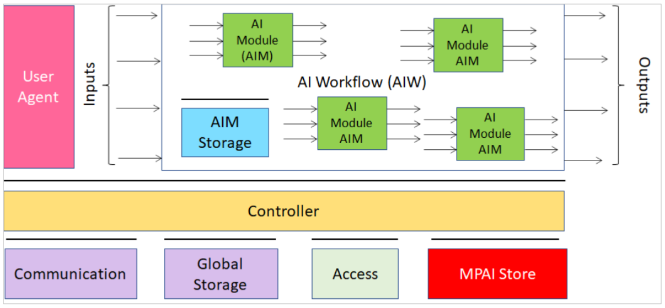

Fig. 1: Reference Model of MPAI-AIF V1.

The standard architecture described by the MPAI-AIF Technical Specification [19] (adopted by IEEE as standard IEEE 3301-2022) has the following key features:

1. No dependency on specific Operating System;

[2] https://mpai.community/standards/

2. Modularity: AIWs are built on AIM components;
3. Standard interfaces encapsulate components;
4. The MPAI Store (a repository of AIWs and AIMs) gives access to verified components; and
5. Components can be implemented in software, hardware or a mix of the two (hybrid).

A prototype of the MPAI-AIF V1 has been implemented for highly-constrained MCUs [2]. It is written in the C language on top of the Zephyr RTOS.

Currently, the MPAI-AIF DC is working on the definition of the 2nd version of the standard [20], which concentrates on adding security services. Figure 2 shows the MPAI-AIF V2 Reference Model.

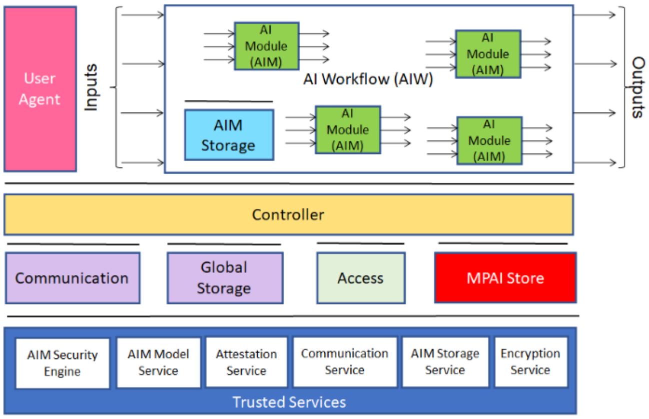

Fig. 2: Reference Model of MPAI-AIF V2.

In particular, MPAI-AIF V2 defines a set of trusted services covering:

1. Trusted execution of AI algorithms such as NN inference (Security Engine);
2. Trusted handling of AI Models such as (Deep) Neural Networks (Model Service);
3. Attestation of the trustworthiness of HW/SW configuration (Attestation Service);
4. Trusted communication (Communication Service);
5. Trusted storage (Storage Service); and
6. Cryptography, including encryption, signing, hashing, ... (Encryption Service).

3 Literature Review

In this section we report a quite extended literature review, which is not intended as a generic survey on TinyML (such as [5]), but rather a focused selection of works in the literature that may provide the basis on which we pursue our research goals. In particular, we are interested in:

- Complex applications consisting of several cooperating modules, possibly running on several devices;
- Systems that are executed exclusively (or prevalently) on constrained, far edge devices; and
- End-to-end security of the systems, from the single-device level to the local network-of-devices, to external communications, e.g., with the cloud.

The following sub-sections are based on the careful review of 18 works (denoted as **P1** to **P18**), that are described and classified along dimensions that emerge from the above list of interests.

3.1 TinyML Systems

According to different studies (e.g., [22, 17, 11]), TinyML is designed to operate with a power consumption of a few milliwatts. To achieve this, the involved devices may utilize hardware acceleration and power management modules to optimize the energy efficiency of the system. The software used in TinyML must also be as compact as possible to minimize RAM requirements and power consumption. Moreover, TinyML requires to optimize machine learning models to provide better accuracy under resource-constraints. When this is possible to achieve, IoT-based embedded edge devices can perform a range of tasks without relying on the cloud. In general, TinyML systems can include energy-harvesting edge devices, battery-operated embedded edge devices, sensors, and in general devices that can store just a few tens or hundreds of KB in RAM.

Our investigation of works that address constrained, intelligent devices executing ML (and AI in general), started with the following questions. Below, we'll go through each of these questions in detail. These are:

1. **Q1**. In the investigated system, is AI executed on end devices, edge, or both?
2. **Q2**. Is AI used for optimizing some system behavior (e.g. communication between device and edge, energy, ...)?
3. **Q3**. Is AI used for ”normal” system operations (e.g., image or sound recognition) ?
4. **Q4**. Is there cooperation between AI-enabled devices?

Examples of TinyML applications include smart wearables, health monitors, environmental sensors, and smart agriculture systems [24]. Smart wearables, for example, can monitor vital signs such as heart rate and detect irregularities to alert the wearer or healthcare professionals. Health monitors can be used to detect changes in patients' conditions and notify caregivers. Environmental sensors can detect changes in air quality, temperature, and humidity levels, allowing for better control of indoor environments. Smart agriculture systems can help farmers optimize crop yields by monitoring soil moisture, temperature, and other environmental factors. Table 1 reports the type of application and the problem solved by 10 of the works we have revised.

Table 1: TinyML Applications.

Paper	**Application Type**	**Solved Problem**
P1 [29]	Intelligent Edge Computing Framework for AI-Drive IIoT	Scheduling problem
P2 [8]	Intelligent data transportation routing	Intelligent and secure edge-enabled computing
P3 [16]	Smart & Sustainable Campus	Indoor Air and Temperature Quality monitoring
P4 [27]	Intelligent and Efficient MEC-access control	Battery Prediction for Mobile-Edge Computing
P5 [13]	Learning IoT in Edge	Wireless Cameras to monitoring and identify
P6 [28]	Tasks-oriented Edge Computing	Reduced the bandwidth for transmission data
P7 [26]	Machine Learning Classifier Execution	ML frameworks for classification
P8 [18]	Face Mask detection Technique	Quantize the CNN model
P9 [22]	Human-Centric embedded machine learning	Emotion-aware facial recognition
P10 [14]	Signal recognition and channel estimation in OFDM systems	Tiny AI for Channel estimation and signal detection

Table 2 reports, for each of the papers, the answers to the four questions outlined above.

In **P1** [29], the authors identify the most challenging issue in IIoT (Industrial IoT) contexts, which is ensuring a limited processing latency. The proposed framework considers heterogeneous resources and identifies the best combination of hardware to handle various AI tasks in the needed processing time. AI operations are implemented using a variety of AI libraries, such as PyTorch or TensorFlow, and the framework may link diverse AI tasks and AI hardware in a transparent way. An important issue is scheduling heterogeneous hardware for improved energy efficiency when performing various AI activities at the intelligent edge. Because unique AI hardware consumes less energy when doing specialized jobs, the proposed framework's schedul-

Table 2: TinyML Systems Main Characteristics.

Paper	Q1	Q2	Q3	Q4
P1 [29]	Both(Edge and Device)	Energy Optimization	Images	Yes
P2 [8]	Edge	Communication Efficiency with low energy consumption	Sensor data	No
P3 [16]	Edge	Energy Optimization	Air Quality and Air Temperature	No
P4 [27]	Server	Energy Optimization	No	No
P5 [13]	Edge	No	Video Sensing	Yes
P6 [28]	Both (Edge and Device)	Communication Optimization	Images	Yes
P7 [26]	Edge and Device	No	Voice, Image, Sound	No
P8 [18]	Device	No	Images	No
P9 [22]	Device	No	Images	No
P10 [14]	End devices	Yes	optimize signal recognition	Yes

ing method takes into account the task type and latency needs to minimize energy consumption while ensuring a limited processing delay.

Paper **P2** [8] presents an intelligent and secure edge-enabled computing model (ISEC) for sustainable cities using Green IoT. The aim of the proposal is to improve communication efficiency in terms of energy optimization and provide security for data transmission. The model exploits deep learning techniques for the data routing, by first predicting the best routes for the data transmission between edge servers, and then for training the sensors to make more accurate predictions for finding a transmission route. Hashing with chaining techniques is adopted to provide an efficient security solution.

In paper **P3** [16] the authors discuss the design methodology used for the architecture specification of the IPVC Smart & Sustainable Campus (IPVC-S2C), a FIWARE-based platform with edge-enabled intelligence targeting higher education institutions. The system is applied to the Indoor Air Quality monitoring on the campus. The authors introduce the core elements and ICT infrastructure required to support the implementation of the platform, while maintaining compatibility with legacy systems.

In paper **P4** [27], authors consider a dynamic MEC-access control problem in an IoT scenario that consists of k MEC (Mobile-Edge Computing) servers and one IoT device with energy harvesting (EH) capability. Then, they propose an intelligent Long Short-Term Memory (LSTM) enhanced Deep Q-Network (DQN) access control algorithm for learning the optimal access control strategy from/to the IoT device. The learned strategy maximizes the long-term average up-link transmission rate whilst minimizing the transmissions energy consumption.

In paper **P5** [13], the authors first introduce deep learning for IoT in an edge computing environment. The authors claim to have developed a novel offloading scheme to improve the performance of IoT deep learning applications with edge computing. In order to adapt deep learning models for IoT to edge computing, the authors develop an elastic model, which offloads (part of) a Deep Network to edge nodes. Finally, they test a deep learning model for IoT with extensive experiments in a given edge computing environment, comparing the results of their proposed edge computing method with conventional solutions.

The authors of paper **P6** [28] address the problem of effectively reducing the bandwidth needed for the transmission of data from the IoT devices to edge computing servers, such that AI model inference of IoT data can be performed. To solve the aforementioned challenge, they design TORC (Tasks-oriented Edge Computing), a multi-task AI model inference framework for IoT edge computing with support for temporally dynamic task importance and bandwidth variation. In TORC, a neural network runs on IoT devices and encodes the data (that is intended for AI analysis tasks on the edge) into a compact representation, based on bitrate weights and importance of tasks. The output of the encoder is then quantized and compressed with an algorithm entropy coding. An edge server receives the encoded data with limited bandwidth, and decodes it with the primary purpose of maintaining the accuracy of AI-based analysis tasks as high as possible.

The authors of paper **P7** [26] propose techniques that preserve the quality of the ML model regardless of the available RAM, namely, they allow fast classification using 0-bytes of SRAM. The proposed technique targets the deployment of models built with standard ML frameworks for classification, such as Python scikitlearn. To show the comparability of the results, the authors compare with known models ported with sklearn, m2cgen and emlearn libraries, and claim that the proposed technique is four times better than the competitors.

In paper **P8** [18], the authors propose a face mask detection technique applicable to smart IoT devices with very low memory. The proposed method is implemented on a tiny development board with memory constraints, such as an ARM Cortex-M7 microcontroller clocked at 480 Mhz and 496 KB frame buffer RAM. The model is quantized to further reduce its size using TensorFlow Lite Framework.

In paper **P9** [22], the authors propose a TinyML technique in smart sensing devices used to detect the presence or absence of fruits in images of plants, focusing on energy consumption in various IoT scenarios. TinyML and LoRaWAN have been used in the proposed model to determine whether an energy-efficient model is capable of detecting fruits availability. The proposed model was tested for accuracy with various experiments and was shown to achieve 90% accuracy and three times energy savings on battery-powered sensors compared to similar cloud-based applications. The proposed techniques are based on the development of embedded vision AI algorithms with TinyML. This technique allows knowing when and where attention is needed by the plants, through LoRaWAN communication.

In paper **P10** [14], the authors discuss a Tiny-ML method for hardware-efficient channel estimation and signal detection in low-cost, low-power edge devices. The method replaces large dense layers with small cascading sublayers to reduce computation and storage requirements. The authors also introduce a rank-restricted back-propagation algorithm to facilitate fast training. The proposed techniques are validated using computer simulations of orthogonal frequency-division multiplexing (OFDM) systems using IEEE Std 802.11a-1999 and the Wireless World Initiative (WINNER II) as the channel model. The authors compare the computations and memory storage of the Tiny-ML approach with a benchmark Fully Connected Deep Neural Networks (DNN) and found that the Tiny-ML scheme can achieve 2.5-3 times acceleration in model training and 4.5 times acceleration in model inference while reducing on-chip storage by roughly 4.5 times in comparison to the FC DNN.

3.2 Resources and Environments

One of the most significant challenges of working with TinyML is the constraints placed on resources like memory and power. In order to create applications that are both efficient and effective, researchers must develop new algorithms and methods for optimizing resource utilization. As a result, the field of TinyML research is constantly evolving, with new techniques and tools being developed all the time.

Based on our study and their analysis, we have drawn a table that shows some characteristics of the models used in the TinyML we have analyzed, and of the SW/HW environments adopted. See Table 3.

3.3 Security

Secure TinyML entails the integration of a number of security measures to protect the data processed and produced by machine learning models running on micro-controllers. These techniques include secure boot techniques that ensure a secure device startup and the integrity of the software during the boot process. To secure data exchanged between the device and the cloud or other endpoints, cryptography, such as encryption and decryption algorithms, is also essential.

Table 3: TinyML Resources and SW Environments.

Paper	Train	Model Size	Board Type or MCU	SW Environment
P1 [29]	Yes	1 to 300 MB	TensorRT 7.1.3	Simulated on NS-3
P2 [8]	No	data packet size 64 bits	ARM based 4 devices used	Tensorflow Lite
P3 [16]	No	–	ARM based 4 devices	Tensorflow Lite
P4 [27]	No	–	–	–
P5 [13]	No	Video data 95kb/s	Simulation	Python 2.7
P6 [28]	No	Images sized 256×512 pixels	Simulation	PyTorch
P7 [26]	Yes	1MB	Arduino Nano and Mega, Blue Pill, ESP 32,01s,8266	Python scikitlearn
P8 [18]	Yes	138KB after full quantization	ARM Cortex-M7 microcontroller	TensorFlow-Lite
P9 [22]	Yes	–	Smart-watches and tiny personal assistants	Tensor Flow Lite
P10 [14]	Yes	Proportional to the recommended compression ratios	All tiny devices	Computer Simulations

Access control mechanisms must be implemented by secure TinyML systems to ensure that only authorized users may access the device and data. This is enabled via user authentication techniques like passwords or biometric identification. In order to protect data transmissions and avoid data surveillance and eavesdropping, protected TinyML additionally uses secure communication protocols like Secure Socket Layer (SSL).

In a variety of fields, including healthcare, industrial IoT, and smart homes, the use of secure TinyML has become increasingly important. Examples include real-time patient monitoring and diagnosis in healthcare using TinyML models running on microcontrollers. To prevent data breaches and maintain patient privacy, it is necessary to process sensitive data, such as patient information and vital signs, with a substantial level of privacy and security.

Therefore, the implementation of secure TinyML is crucial for the adoption and deployment of TinyML solutions in various fields. The incorporation of secure boot processes, cryptography, access control, and secure communication protocols ensures the protection of sensitive data and devices from unauthorized access and cyber-attacks.

The following list of security measures can be implemented to make TinyML more secure:

1. Encryption: Data, models, and firmware can be encrypted to prevent unauthorized access.

2. Authentication: Authentication protocols can be used to ensure only authorized users can access the system.
3. Secure Communication: Secure communication protocols like SSL may be utilized for secure communication between memory-constrained devices and towards external targets such as cloud servers.
4. Access Controls: Access controls can be implemented to ensure that only authorized users get access to the system.
5. Protection of User Data Privacy: privacy approaches, including differential privacy, can be used to ensure user data privacy.

Some TinyML hardware and software security are shown in Table 4. Table 4 reports information about 10 of the works we have revised that discuss security features. Two of the papers (**P2** and **P9**) have already been mentioned in the tables above, while eight papers **P11 - P18** are specifically relevant to security.

Table 4: Secure TinyML.

Paper	**HW Security**	**SW Security**	**Security Techniques**
P2 [8]	No	Yes	Symmetric cryptography is performed using the Diffie-Hellman algorithm
P9 [22]	No	Yes	Practical implications of adversarial threats on HC-EML applications
P11 [3]	No	Yes	Cutting-edge technologies, such as TinyML,HE, and FL
P12 [15]	No	Yes	Secure Aggregation based on Cryptographic Schemes
P13 [12]	No	Yes	Protect the model's intellectual property using security techniques
P14 [6]	Yes	Yes	Protect the AI models developed and the training data used
P15 [25]	Yes	Yes	OTA-TinyML tested 6 ML models remotely from 4 memory units on 7 MCU boards
P16 [7]	No	Yes	EC-IG prototype: OP-TEE + QEMU on Armv8-A
P17 [9]	NO	Yes	Security and privacy architecture for HioT
P18 [4]	No	Yes	Securing Models: Defending Against Three Standard Adversarial Attacks

In paper **P11** [3] , the authors discuss a study that focuses on Wearable Devices (WD) used for healthcare monitoring. The authors categorizes these technologies and identify privacy and security risks connected to their use. In addition, the authors review a number of cutting-edge technologies, such as TinyML, Homomorphic Encryption (HE), and Federated Learning (FL), that may be capable of dealing with these challenges. This study offers a taxonomy of threats and attacks that may compromise the security and privacy of WD. The study's main objectives are to emphasize the significance of protecting private health information, and to propose technologies that could help reduce the associated risk. In particular, the authors identify (i) the user behavior and perception, (ii) health-related data transfer, and (iii) data storage, as the three primary challenges involving security and privacy in WDs.

The authors of paper **P12** [15] discuss a research study that focuses on secure aggregation for federated learning using cryptographic schemes. The authors define the problem, categorize existing solutions, analyze challenges, and examine recent solutions of other studies. They then propose an improved definition of secure aggregation for federated learning and single out the four main privacy-enhancing technologies used to achieve secure aggregation. Namely: (i) differential privacy; (ii) Trusted-Execution Environment (TEE); (iii) secure shuffling under anonymity assumptions; and (iv) cryptography.

In paper **P13** [12], authors focus on investigating other challenges that arise when edge ML is deployed and how to solve them, rather than the computational aspect. The authors explore the operational side of edge ML and discuss approaches that can be applied to various use cases, such as image recognition and natural language processing but not limited to these research areas. The paper assumes that the ML model is a deep neural network, but the discussed approaches are useful for other types of ML models as well. The authors focus on protecting the intellectual property of the ML model, particularly for TinyML and explain the challenges associated with protecting the model, providing solutions to address these challenges. The suggested techniques include model encryption, obfuscation, watermarking, and secure model training.

Paper **P14** [6] discusses the challenges faced by service providers when deploying AI models on IoT devices in a secure manner. The provider must consider the trustworthiness of the supporting platform and the range of potential threats to the device, such as direct attacks on the model IP or security of input data. To scale deployments to multiple devices and OEMs, it becomes necessary to separate untrusted code from the high-value model IP. The article emphasizes the need for secure inference outputs to prevent tampering and ensure the integrity of the AI model throughout the supply chain. The authors mention a DevSecOps workflow integrating standardized tooling for configuration, packaging, and image signing while selecting cryptographic algorithms that fit within the resources of constrained IoT devices.

In paper **P15** [25], authors propose a novel technique, called over-the-air (OTA) TinyML, which allows for the remote deployment of memory constrained machine learning models on IoT devices. The mentioned technique enables IoT devices to perform tasks such as firmware updates, reconfiguration, and repurposing. The authors first discuss the challenges of OTA ML deployment over IoT devices from both research and engineering perspectives. Second, they claim that their OTA-TinyML framework facilitates the fetching, storage, and execution of TinyML models on resource-constrained IoT devices. It downloads the C source file of ML models from a web server to the embedded IoT devices using HTTPS. OTA-TinyML is tested by remotely fetching six types of ML models, storing them on four types of memory units, and then loading and executing them on seven popular MCU boards.

In paper **P16** [7], with the aim of enabling Industry 4.0 vertical and horizontal integration, the authors propose an Edge-computing based Industrial Gateway (EC-IG) for interfacing information technology and operational technology. They design and develop a working prototype to demonstrate remote production-line maintenance with a focus on security and the edge paradigm. The EC-IG brings computational resources and data storage closer to data sources, improving security aspects and enhancing the efficiency of the overall system.

The authors of paper **P17** [9] focus on the security and privacy of the Internet of Things (IoT) in healthcare applications. They highlight the challenges while implementing security frameworks and focus on the need for effective security and privacy solutions. The article aims to provide current challenges to the security and privacy of IoT in healthcare and to encourage the adoption of robust security measures to protect sensitive patient data. A security and privacy architecture for HioT is proposed.

In paper **P18** [4] the authors discuss the application of edge AI through a human-centric perspective, presenting a pipeline for developing human-centric embedded machine learning (HC-EML) applications using a generic human-centric AI (HCAI) framework. Authors also analyze the privacy, trustworthiness, robustness, and security aspects of HC-EML applications, including challenges and possible solutions. The paper includes a case study on human facial emotion recognition (FER) based on the AffectNet dataset, analyzing the effects of input quantization on the security, robustness, fairness, and trustworthiness of an EML (Embedded Machine Learning) model. The FER model is found to be heavily influenced by certain facial features such as eyes, alar crease, lips, and jaws. Moreover, the input quantization is biased against dark skin faces, possibly due to low-contrast features.

4 Towards Complex, Secure TinyML Applications

4.1 State of the Art and Research Gaps

The literature review presented in section 3 has allowed us to get an understanding of the state-of-the art in technologies related to secure Tiny ML applications. From our analysis, we have drawn the following observations:

1. Many *edge-AI* systems still rely on powerful edge nodes, or even the cloud, as fundamental components to address complex AI tasks.
2. Systems that are strongly based on constrained devices (TinyML) tend to have simple architectures where:
   - end devices perform a simple ML task (e.g., image classification) individually, and then
   - communicate their results to *upper level* system components (e.g. edge servers) for storing and analysing them.
3. Various model optimization techniques exist that make it possible to store AI models and perform inferences even in highly constrained devices.
4. Security of TinyML systems inherits methods for protecting the Intellectual Property (IP) of AI models from the more general ML security area (e.g., watermarking, obfuscation, cryptography.)
5. Lightweight cryptographic technologies can be adopted for authentication, confidentiality, and integrity of communications, while hardware-backed Trusted Execution Environments (TEE) can be exploited to guarantee security within single devices.

Therefore, the main gaps we have found towards our goal of enabling complex, secure TinyML applications can be summarized as follows:

1. There is currently a lack of research and implementations of complex, cooperative TinyML applications, i.e., applications composed by several modules, possibly running on separate, heterogeneous devices and exchanging information through a wireless network. Some challenges posed by such applications include:
   - *interoperability* of application modules at the data level, i.e., the data produced by a module must be consumable by other modules, possibly created by different vendors and/or reusable in different contexts;
   - seamless *connection* of modules on a single device, or multiple devices to build an end-2-end application from sensor inputs to the final outputs; and
   - automatic *mapping* of the modules of a complex application into a network of devices, if no device is sufficiently powerful to completely host it, or partitioning is convenient for other reasons such as energy or computational efficiency.

   As we shall see, the MPAI-AIF specification [19] is a very promising starting point for addressing these challenges, although it needs to be extended in a suitable way.
2. The main gap that we could find in the state-of-the art of security is the lack of an application programmer-friendly framework addressing all the security aspects relevant for complex TinyML applications:
   - Portability across a variety of MCUs and RTOSes.

- Coverage of (i) single device security, (ii) communication security, (iii) Intellectual Property (AI models) security.

The MPAI-AIF V2 specification under development [20] can be at the same time a source and a target of the developments for addressing these open challenges (see section 4.2).

4.2 Sketch of a Proposal

As an example of a potential complex TinyML application, let us refer to the MMC-UST (Multi-Modal Conversation - Unidirectional Speech Translation) defined by MPAI in [21]. Figure 3 shows the workflow associated with the Use Case.

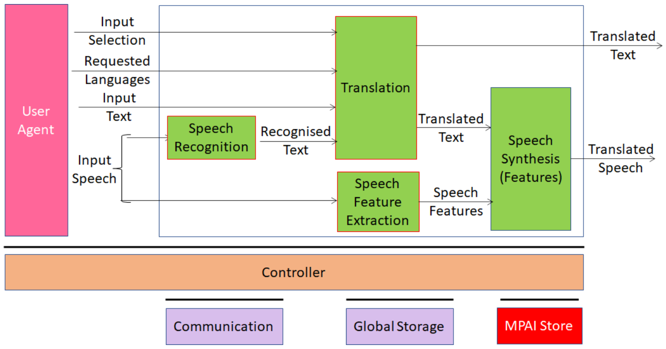

Fig. 3: MPAI MMC-UST Use Case.

Without going into the details of the UC, which are beyond the scope of the present work, we note that:

- the input speech (coming from microphones) is processed by two modules, one for extracting the text (Speech Recognition), and one for extracting additional features, such as the intonation (Speech Feature Extraction);
- the extracted text is processed by a Translation module into a translated text; and
- the features and translated text are used as inputs of a Speech Synthesis module that produces an output audio, possibly sent to a speaker.

The *data-interoperability* of the modules is addressed by the MPAI MMC-UST standard, which defines, e.g., the syntax and semantics of the Speech Features produced by the Speech Feature Extraction module and consumed by the Speech Synthesis module.

On the other hand, the MPAI specifications do not say anything about whether the modules should be hosted on a single device, or *distributed* on several devices. The following is one of many possibilities: the microphone, the Speech Recognition and the Speech Feature Extraction modules are hosted on device D_{IN}; the Translation module is hosted on device D_{TR} (likely, a sufficiently powerful edge node, or even the cloud); and the Speech Synthesis module is hosted on device D_{OUT} which is directly connected to speakers. As already pointed out in section 4.1, we need a transparent way (to the module programmer) to distribute modules across several devices, and possibly to do this in an *automated and dynamic* way. In other words, given the *metadata* describing the modules of the Use Case and their data connections, we would like to automatically map the application to different devices without having to recompile its modules; and, if conditions change (e.g., a communication link becomes broken or too slow), to adjust the mapping to best fit the new conditions.

In Figure 4, we have depicted a schema of a TinyML application that is mapped to three different devices by an Application Partitioner.

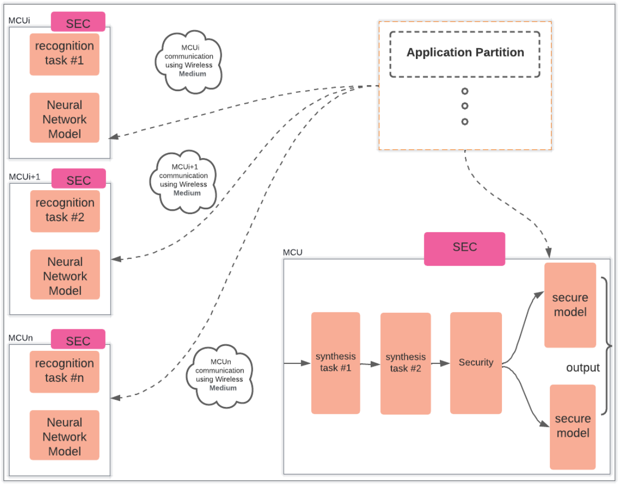

Fig. 4: Schema of a Distributed TinyML Application.

The Application Partitioner could be one of the devices that will host the application, or an external node (edge, cloud). The fundamental point is that it must have access to metadata describing the application as well as to metadata describing the available devices in the local network. We can build upon and extend features offered by the MPAI-AIF to realize this architecture:

- MPAI-AIF already defines an architectural component named MPAI Store (Figure 1), from which metadata about complete Use Cases (AIWs) as well as their individual components (AIMs) can be downloaded. We need to extend such metadata in order to include resource usage of modules (input and output data size and rate, RAM, flash, ...).
- We also need to associate metadata to the network devices (clock speed, RAM, flash, sensors, wireless communications, ...); there should be a way for the Application Partitioner to know such metadata, either from a static Knowledge Base or by collecting it through some form of discovery over the local network of devices.
- We need a mapping algorithm, which decides how to partition the application and allocate its parts to different devices; many approaches for this kind of allocation are possible:
  - optimal or approximated;
  - centralized or distributed (in the latter case, the devices themselves may be involved in the computation of the partition); and
  - static or adaptive (in the latter case, it is required to monitor the application to gather the information that may call for adjustments of the partition).
- We need a way to communicate to each device the part of the application that it has been assigned, and how such part connects with other parts hosted on different devices.

As mentioned above, the new MPAI AIF specification V2 under development will define security services offered by the AIF (Figure 2). Since some of the authors of this paper participate to the Definition Committee (DC), the standard specification itself is going to incorporate some features required by our proposed architecture. Without committing to what will be included in the standard and what will be left (at least temporarily) outside, we can point out the following features:

- A Model Service will have to provide functions for the secure download and update of TinyML models (most often, but not necessarily, Neural Networks that have been optimized).
- A Communication Service will have to provide functions for the secure exchange of data between modules. The modules using this service will not know whether the module(s) they want to exchange data with are local or remote, and will be determined by how the application has been partitioned.
- A Firmware Update service will have to provide secure update of parts of the Firmware (modules of the TinyML application).
- Appropriate programmer-friendly tools will have to be available at firmware compile-time to:
  - configure the memory layout of the device distinguishing Secure and Non Secure areas;

- configure secrets (e.g., keys) to be provisioned on the device and provision them; and
- configure the firmware for including just the security algorithms actually needed by the application (and their dependencies).

5 Conclusions

The recent trend of performing machine learning tasks on memory-constrained devices at the edge, known as TinyML, it offers several advantages such as lower energy consumption and increased privacy. However, it also presents significant challenges such as performing complex ML tasks on microcontrollers with limited resources. Additionally, the security threats associated with these devices are a major concern.

In this work we have provided a review of related literature and proposed a research directions to advance the state-of-the-art in secure and energy-efficient ML applications, with a special focus on a standardized architecture proposed by the MPAI organization for far edge AI applications. Overall, addressing the challenges of TinyML while leveraging its benefits has the potential to transform the field of AI and bring advanced machine learning capabilities to smart devices at the edge.

Acknowledgements This publication was partially supported by the Italian PRIN project "CommonWears" (2020HCWWLP) and the EU/MUR FSE PON-R&I 2014-2020. The work is also part of the project NODES which has received funding from the MUR – M4C2 1.5 of PNRR with grant agreement no. ECS00000036. The work was also carried out within the Agritech National Research Center and received funding from the European Union Next-GenerationEU (PIANO NAZIONALE DI RIPRESA E RESILIENZA (PNRR) – MISSIONE 4 COMPONENTE 2, INVESTIMENTO 1.4 – D.D. 1032 17/06/2022, CN00000022). This manuscript reflects only the authors' views and opinions, neither the European Union nor the European Commission can be considered responsible for them.

References

1. Arfat, Y., Mittone, G., Esposito, R., Cantalupo, B., DE Ferrari, G.M., Aldinucci, M., et al.: A review of machine learning for cardiology. Minerva cardiology and angiology pp. 1–23 (2021)
2. Basso, A., Bortoluzzi, D., Torta, G.: Implementation of an iot wearable prototype on a standard ai architecture. In: 2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 1–5. IEEE (2022)

3. Boumpa, E., Tsoukas, V., Gkogkidis, A., Spathoulas, G., Kakarountas, A.: Security and privacy concerns for healthcare wearable devices and emerging alternative approaches. In: Wireless Mobile Communication and Healthcare: 10th EAI International Conference, MobiHealth 2021, Virtual Event, November 13–14, 2021, Proceedings, pp. 19–38. Springer (2022)
4. Butt, M.A., Qayyum, A., Ali, H., Al-Fuqaha, A., Qadir, J.: Towards secure private and trustworthy human-centric embedded machine learning: An emotion-aware facial recognition case study. Computers & Security **125**, 103058 (2023)
5. Dutta, D.L., Bharali, S.: Tinyml meets iot: A comprehensive survey. Internet of Things **16**, 100461 (2021). DOI https://doi.org/10.1016/j.iot.2021.100461
6. Fletcher, B.: Confidential AI for MCUs. White paper (2021)
7. Gupta, S.: An edge-computing based industrial gateway for industry 4.0 using arm trustzone technology. Journal of Industrial Information Integration **33**, 100441 (2023)
8. Haseeb, K., Din, I.U., Almogren, A., Ahmed, I., Guizani, M.: Intelligent and secure edge-enabled computing model for sustainable cities using green internet of things. Sustainable Cities and Society **68**, 102779 (2021)
9. Karunarathne, S.M., Saxena, N., Khan, M.K.: Security and privacy in iot smart healthcare. IEEE Internet Computing **25**(4), 37–48 (2021)
10. Kausar, S., Huahu, X., Ahmad, W., Shabir, M.Y.: A sentiment polarity categorization technique for online product reviews. IEEE Access **8**, 3594–3605 (2019)
11. Khan, I., Guerrieri, A., Spezzano, G., Vinci, A.: Occupancy prediction in buildings: An approach leveraging lstm and federated learning. In: 2022 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech), pp. 1–7. IEEE (2022)
12. Leroux, S., Simoens, P., Lootus, M., Thakore, K., Sharma, A.: Tinymlops: Operational challenges for widespread edge ai adoption. In: 2022 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW), pp. 1003–1010. IEEE (2022)
13. Li, H., Ota, K., Dong, M.: Learning iot in edge: Deep learning for the internet of things with edge computing. IEEE network **32**(1), 96–101 (2018)
14. Liu, H., Wei, Z., Zhang, H., Li, B., Zhao, C.: Tiny machine learning (tiny-ml) for efficient channel estimation and signal detection. IEEE Transactions on Vehicular Technology **71**(6), 6795–6800 (2022)
15. Mansouri, M., Önen, M., Jaballah, W.B., Conti, M.: Sok: Secure aggregation based on cryptographic schemes for federated learning. Proceedings on Privacy Enhancing Technologies **1**, 140–157 (2023)
16. Martins, P., Lopes, S.I., Curado, A.: Designing a fiware-based smart campus with iot edge-enabled intelligence. In: Trends and Applications in Information Systems and Technologies: Volume 3 9, pp. 557–569. Springer (2021)
17. Merenda, M., Porcaro, C., Iero, D.: Edge machine learning for ai-enabled iot devices: A review. Sensors **20**(9), 2533 (2020)
18. Mohan, P., Paul, A.J., Chirania, A.: A tiny cnn architecture for medical face mask detection for resource-constrained endpoints. In: Innovations in Electrical and Electronic Engineering: Proceedings of ICEEE 2021, pp. 657–670. Springer (2021)
19. MPAI Community: Artificial intelligence framework (mpai-aif) v1.1. https://mpai.community/standards/resources/
20. MPAI Community: Artificial intelligence framework (mpai-aif) v2.0. https://mpai.community/standards/mpai-aif/about-mpai-aif/#V2, under development
21. MPAI Community: Multi-modal conversation (mpai-mmc) v1.2. https://mpai.community/standards/resources/

22. Nicolas, C., Naila, B., Amar, R.C.: Tinyml smart sensor for energy saving in internet of things precision agriculture platform. In: 2022 Thirteenth International Conference on Ubiquitous and Future Networks (ICUFN), pp. 256–259. IEEE (2022)
23. Rashid, M., Khan, M.A., Alhaisoni, M., Wang, S.H., Naqvi, S.R., Rehman, A., Saba, T.: A sustainable deep learning framework for object recognition using multi-layers deep features fusion and selection. Sustainability **12**(12), 5037 (2020)
24. Shafique, M., Theocharides, T., Reddy, V.J., Murmann, B.: Tinyml: current progress, research challenges, and future roadmap. In: 2021 58th ACM/IEEE Design Automation Conference (DAC), pp. 1303–1306. IEEE (2021)
25. Sudharsan, B., Breslin, J.G., Tahir, M., Ali, M.I., Rana, O., Dustdar, S., Ranjan, R.: Ota-tinyml: over the air deployment of tinyml models and execution on iot devices. IEEE Internet Computing **26**(3), 69–78 (2022)
26. Sudharsan, B., Patel, P., Breslin, J.G., Ali, M.I.: Ultra-fast machine learning classifier execution on iot devices without sram consumption. In: 2021 IEEE International conference on pervasive computing and communications workshops and other affiliated events (PerCom Workshops), pp. 316–319. IEEE (2021)
27. Xu, L., Qin, M., Yang, Q., Kwak, K.: Deep reinforcement learning for dynamic access control with battery prediction for mobile-edge computing in green iot networks. In: 2019 11th International Conference on Wireless Communications and Signal Processing (WCSP), pp. 1–6. IEEE (2019)
28. Zhang, J., Zhang, W., Xu, J.: Bandwidth-efficient multi-task ai inference with dynamic task importance for the internet of things in edge computing. Computer Networks **216**, 109262 (2022)
29. Zhu, S., Ota, K., Dong, M.: Green ai for iiot: Energy efficient intelligent edge computing for industrial internet of things. IEEE Transactions on Green Communications and Networking **6**(1), 79–88 (2021)