A Review on Machine Learning in IoT Devices

Imopishak Thingom[1], N. Basanta Singh[2]

[1]NIELIT, Guwahati,
imothingom@yahoo.com
[2]MIT, Manipur
basanta_n@rediffmail.com

Abstract:
Deployment of machine learning and AI continue to rapidly expand across a range of applications. Billions of devices are getting interconnected using IoT and the number of connected devices continues to grow exponentially. In edge computing the computation and intelligence are being brought closer to the edge devices like the gateways. Recently there is a growing interest in running machine learning models in the low-end devices particularly in embedded devices like microcontrollers (MCUs). Unlike high performance CPUs and GPUs these are resource constrained devices in terms of memory and computing power. This new paradigm of bringing machine learning and embedded devices together is referred to as TinyML. In this paper we present the motivation behind this paradigm, the challenges, and approaches to overcoming these challenges, popular frameworks and future directions.

Keywords: Edge computing, IoT, microcontroller, machine learning, embedded system, tiny machine learning

I. INTRODUCTION

IoT is being deployed in a wide range of applications with billions of connected devices. These IoT devices generate huge amount of raw data. Traditionally, these data are processed in the cloud as embedded and IoT devices are resource constrained and does not have the computing power to run machine learning models. Much of these embedded IoT devices are MCUs like ARM Cortex with limited memory and storage. It is estimated that there there are around 300 billion microcontrollers today [1]. This opens great possibilities for building an intelligent and interconnected world [2]. This has motivated the idea of deploying neural networks on these tiny devices. TinyML evolved from this idea. This new paradigm of TinyML is aimed at deploying small machine learning models on these devices with sufficient speed and accuracy. Considering the huge potential, AI and TinyML are being adopted for United Nations' Sustainable Development Goals [3].

II. EDGE COMPUTING AND ML

Cloud computing has limitations such as high latency, security, privacy and high bandwidth requirements. To overcome these edge computing evolved where some of the the work are offloaded on the edge devices like gateways and closer to the devices which generate the raw data. Edge computing [4] refers to the enabling technologies which allow computation to be performed at the edge of the network, on downstream data on behalf of cloud services and upstream data on behalf of IoT services. The idea is to allow computing and analysis closer to the end device. In IoT applications this allows aggregation and analysis of enormous amount of data close to the source of the data [5].However, in the traditional IoT the end devices merely transmit the raw data to the cloud for inference. The large AI model runs in the cloud with huge amount of memory and processing power. Here the main objective is accuracy and speed. As the number of devices continues to grow there is an increasing need to bring intelligence to the IoT devices such as sensors and embedded microcontrollers. Also, the increasing use of mobile devices has motivated the use of deep learning in these devices [6]. This requires that neural networks are more compact and efficient run in these devices. The goal is to bring intelligence to the low-end embedded device like the microcontrollers and sensors [7]. However; these devices have limited resources like onboard memory. On the other hand, machine learning models and frameworks like Tenosorflow require plenty of resource like memory and processing power. These rely on powerful CPUs and GPUs.

III. TINYML

Microcontrollers are ubiquitous devices used in various devices from TVs to cars. They form a large of the part of the embedded devices. However, these microcontrollers are resource constrained such as the amount of RAM or the clock speed. For example a typical ARM Cortex M microcontroller has upto 128 KB RAM, upto 1MB flash with a clock speed of 128MHz[8] However they have important characteristic i.e. they consume much less power in the range of microwatts or milliwatts and cost less. This is much less compared to a powerful modern CPU or a GPU. Thus, embedded microcontrollers

play vital role in IoT where they are deployed as the end-devices for collecting huge amount of raw data and providing the control logic. As we move away from the cloud and closer to the source of the data, we try to build more intelligence into the edge devices and the embedded IoT devices as well. However, it is a huge challenge to be able to bring intelligence to the devices such as an 8-bit microcontroller.

Recently, there has been great interest from the academia and industry to deploy ML on resource constrained embedded devices such as MCUs. However, they have memory in kilobyte whereas a CNN would require memory in tens of GBs with millions of parameters [9].TinyML is a paradigm which refers to machine learning on resource constrained devices like embedded microcontrollers and sensors [10,11]. In a blog by Pete Warden in 2018, 'Why the Future of Machine Learning is Tiny' [12] he proposed running ML on tiny MCUs. TinyML is an intersection of machine learning algorithms and embedded systems allowing machine learning inference with ultralow power consumption (less than 1mW). TinyML uses inference of a pre-trained model to run on device like microcontroller. This uses model compression like pruning and quantization. Advantages of running ML on devices like MCUS are latency (some inference is done locally), reduced data transmissions (basic inference is done without pushing the data to the cloud, energy efficiency (microcontrollers consume little power), low cost, privacy (some data processing is done locally so less data is transmitted) and autonomy [13]. Figure 1 gives an overview of how TinyML provides data processing in context of cloud and edge computing environments.

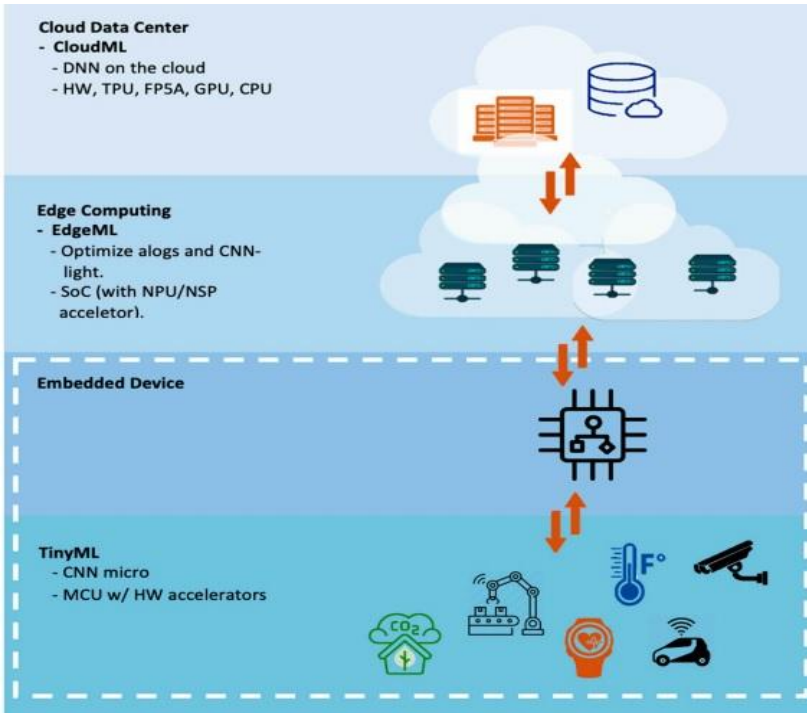

Fig. 1. A framework of IoT applications with cloud computing, edge computing and TinyML

A. TinyML challenges and solutions

There are several challenges in deploying ML in devices like the MCU due to reasons including constrained resources and high power consumption required to run ML models. The on board memory in IoT devices like an MCU are few hundred kilobytes of SRAM and around 1MB of flash memory and they are expected to run on batteries for months or years.

B. Technique for overcoming the challenges

Traditional deep neural networks achieve speed accuracy with models consisting of millions of parameters and with high computing power. To enable deployment of TinyML model compression techniques are used. Compression of the neural network model is used to reduce machine learning models so that they are able to run with minimal resources. A number of approaches are used to such as pruning and quantization [14].Model pruning allows reduction in number of connections between layers [15]. In quantization, the network size is reduced by reducing the number of bits used to represent the weights like low precision arithmetic [16]. Weights which are represented in floating-point numbers are represented as fixed-point integers. Low precision number is widely used for model reduction. However reduction in bits precision using fixed-point can reduce the performance due to limited range [17,18]. A tapered-fixed point format may be used with unequal magnitude spacing and flexible dynamic range [19]. The efficiency of such tapered fixed-point numeral format compared with the standard fixed-point numerical format on multiple benchmarks is explored in a framework called a TENT [20]. In this a tapered fixed-point quantization algorithm is introduced. This showed that tapered fixed-point format has better performance than a

fixed point format by reducing quantization error with only moderate increase in power consumption. Network compression is used to reduce the network coefficients using some lossless coding [21]. In knowledge distillation a well optimized model('student') model is created from a pre-trained model('teacher') model [22, 23]. Another approach is to improve the neural search architectures based on reinforcement learning and evolutionary search algorithms. Automated mobile neural architecture search (MNAS) is a CNN for mobile devices where latency is considered with a factorized hierarchical search space[24].

The above methods are based on reducing the number of parameters and FLOPS. However, tight memory constraints in MCUs lead to small model capacity or small input image size. These are acceptable for image classifications but not for other dense task such as object detection which limits ML on low-end devices like MCUs. It is seen that there is a memory bottleneck due to imbalanced activation memory distribution. MCUNet v2[25] is a patch – based inference which reduces peak memory usage of existing networks by up to 8x. Here a patch-by patch execution order is utilized for the initial memory intensive stage ofconvolutional neural network. This is unlike the conventional method where there is a layer-by-layer execution. Existing TinyML frameworks such as TFLM (Tensor Flow Lite for micros) uses layer-by-layer execution. Figure 2. Depicts a TinyML work flow. This process consists of training, optimizing, and deploying.

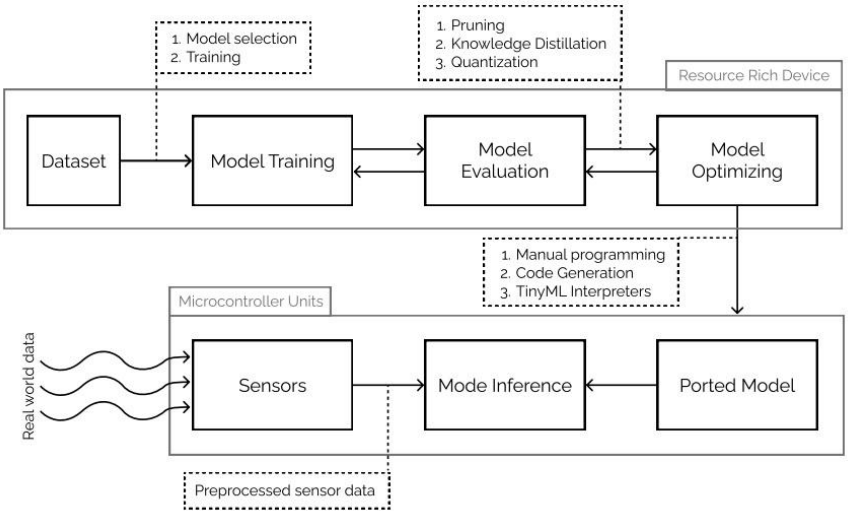

Fig. 2. A framework of IoT applications with cloud computing, edge computing and TinyML

C. Tiny ML Frameworks

Some TinyML software frameworks are Tensor Flow Lite for micros, uTensor, STMCubeAI, Edge ML, Edge Impulse and MinUnML. Also platforms such as Neuton provides AutoML or no-code features for easy deployment of TinyML. The features of some of the frameworks are given in Table 1.

TABLE 1 FEATURES OF TINYML FRAMEWORKS

Framework	**Algorithms**	**Compatible Platforms**	**Output Languages**	**Interoperable External Libraries**	**Publicly Available**	**Main Developer**
TensorFlow Lite	Neural networks	ARM Cortex-M	C++	TensorFlow	Yes	Google
NanoEdge AI Studio	Unsupervised learning	ARM Cortex-M	C	-	No	Cartesiam
STM 32Cube. AI	Neural networks	STM32	C	Keras TensorFlow Lite Caffe ConvNetJs Lasagne	Yes	STMicroelectronics
uTensor	Neural networks	mBed boards	C++	TensorFlow	Yes	Particular developer
ELL	Neural Networks	ARM Cortex-M ARM Cortex-A Arduino	C, C++	CNTK Darknet ONNX	Yes	Microsoft

micro:bit

Tensor flow Lite and Tensor flow Light for Micros

Tensorflow light [26] is a set of tools that enables on-device machine learning by helping developers run their models on mobile, embedded and edge devices. It is optimized for on-device machine learning by addressing key constraints such as latency, privacy, and connectivity, size and power consumption. It supports multiple platforms such as Android and iOS devices, embedded Linux and microcontrollers. It also has various languages support such as Java, Swift, Objective-C, C++ and Python.

TensorFlow Lite Micro (TFLM) is an open-source ML inference framework for running deep-learning models on embedded systems [27,28]. TFLM was created to address serious issues in adoption of TinyML such hardware heterogeneity, fragmented eco system, lack of benchmark, lack of optimization and resource constraints and developments in deep learning. TFLM enable deployment of TinyML across different architectures. TFLM is an interpreter- based machine learning framework. It is designed to run machine learning models on microcontrollers and other devices with only a few kilobytes of memory. The core runtime fits in 16 KB on an Arm Cortex M3 and can run many basic models. It doesn't require operating system support any standard C or C++ libraries, or dynamic memory allocation. It supports various development boards such as Arduino Nano 33 BLE Sense,SparkFun Edge,STM32F746 Discovery kit,AdafruitEdgeBadge,Adafruit TensorFlow Lite for Microcontrollers Kit,Adafruit Circuit Playground Bluefruit,Espressif ESP32-DevKitC,Espressif ESP-EYE and others.

Edge impulse

Edge impulse [29] is an ML platform for embedded ML or TinyML. It supports AutoML. After training on the platform, it can run on the edge device. It can also be integrated with other firmware and the model can be exported to C++ or Arduino library. It support several board such a Arduino Nano 33 BLE Sense, Arduino Nicla Sense ME, Arduino Nicla Vision, Arduino Portenta H7 + Vision Shield, Espressif ESP32, Himax WE-I Plus, Infineon PSoC 62S2 Wi-Fi BT Pioneer Kit, TI CC1352P Launchpad, Raspberry Pi RP2040 among others.

NanoEdge AI

NanoEdge[30] AI Library contains an AI model designed to bring machine learning capabilities to any C code, in the form of easily implementable functions, such as learning signal patterns, detecting anomalies, classifying signals, or extrapolating data. There are different types of NanoEdge AI Libraries for creating projects that can be created in NanoEdge AI. NanoEdge AI Library is an Artificial Intelligence (AI) static library originally developed by Cartesiam for embedded C software running on Arm Cortex microcontrollers. It comes in the form of a precompiled file that provides building blocks to implement smart features into any C code.

STM32Cube.AI

It is a code generation and optimization software that allows machine learning and AI related tasks easier for STM32 ARM Cortex M based boards[31]. Implementation of neural networks in STM32 board can be directly achieved by using STM32Cube.AI to convert the neural nets into an optimized code for most appropriate MCU. It can optimize the memory usage during run time. It can use any trained model by conventional tools such as TensorFlow Lite, Keras, qKeras, TFL,ONNX, Matlab,andPyTorch. This tool is an extension of the original STM32CubeMX framework that helps STM32Cube.AI to perform code generation for target STM32 edge device and middleware parameter estimation**.**

uTensor

uTensor[32] is a free and open source embedded machine learning infrastructure designed for rapid-prototyping and deployment. uTensor is a user-friendly workflow that enables machine learning inferencing on microcontrollers (MCUs) built on top of TensorFlow and Mbed. It is an extremely light-weight machine learning inference framework built on Tensorflow and optimized for Arm targets. uTensor is one of the first frameworks that bring machine learning inference onto the edge devices. The project includes an inference engine, a highly customizable graph processing tool and a data collection framework. The device code is optimized for hardware with only a few kilobytes of memory. uTensor consist of a standalone C++ runtime library and a code generator. The C++ library contains common kernel functions required in neural network evaluations. The code generator automates the process of creating C++ inference functions from models. The runtime library and an offline tool handles model translation work. The size of the core runtime is only ~2KB.

Embedded Learning Library (ELL)

The Embedded Learning Library [33] is an open source project to create a cross compiler for machine learning models. The ELL library has tools to convert a standard ML model, such as one created by CNTK or Darknet, into a universal .ell file (in JSON format). Then ELL tools can compile the .ell file into a model for a target device, such as a Raspberry Pi or an Android phone.

IV. CONCLUSION

TinyML is accelerating deployment of machine learning on devices like microcontrollers. This will allow the embedded and IoT devices to provide inference on the devices where data are generated without continuous dependence on cloud or edge (connectivity) infrastructure. Also, there are emerging approaches such as TinyML as a service (TinyMLaaS), reforming tinyML which will allow updating the model and AutoML. Also MLOPs will allow scaling TinyML to large-scale deployments. However there are challenges as the frameworks, tools, platforms and benchmarks continue to evolve for the diverse hardware devices from different vendors.

REFERENCES

[1] IC Insights. 2020. MCUs Expected to Make Modest Comeback After 2020 Drop.
[2] Wu, Z., Qiu, K., Zhang, J. A Smart Microcontroller Architecture for the Internet of Things. Sensors. 2020, 20, 18211.
[3] Zennaro, M., Plancher, B., Reddi V. J. TinyML: Applied AI for Development Resources(52042), pp. 1-4.
[4] Guleria, C., Das, K., Sahu, A. A Survey on Mobile Edge Computing: Efficient Energy Management System. 2021 Innovations in Energy Management and Renewable Resources(52042), pp. 1-4
[5] Weisong S. Edge Computing: Vision and Challenges. IEEE Internet of Things Journel, vol. 3, No. 5, 2016
[6] Chen, Y., Zhang, Y., Zhang, Z.,Wang, Q., Chen, C., Zhang, Q. Deep Learning on Mobile and Embedded Devices: State-of-the-Art, Challenges, and Future Directions. ACM Comput. Surv. 2020.
[7] Sanchez-Iborra, R., Skarmeta, A. F. TinyML-enabled frugal smart objects: Challenges and opportunities. IEEE Circuits and Systems Magazine, vol. 20, no. 3, pp. 4–18, 2020
[8] https://www.st.com/en/microcontrollers-microprocessors/stm32f215ze.html.
[9] Dutta, L and Bharali, S. Tinyml meets iot: A comprehensive survey. Internet of Things, vol. 16, p. 100461, 2021.
[10] P. P. Ray. A review on tinyml: State-of-the-art and prospects. Journal of King Saud University-Computer and Information Sciences, 2021.
[11] Warden, P. and Situnayake, D., 2019. Tinyml: Machine learning with tensor flow lite on Arduino and ultra-low-power microcontrollers. O'Reilly Media.
[12] https://petewarden.com/2018/06/11/why-the-future-of-machine-learning-is-tiny/
[13] Sakr, F., Bellotti, F., Berta, R., De Gloria, A. Machine Learning on Mainstream Microcontrollers. Sensors 2020, 20, 2638.
[14] S. Han, H. Mao, W. J. Dally Deep Compression: Compressing Deep Neural Networks with Pruning, Trained Quantization and Huffman Coding.arXiv:1510.00149
[15] S. Han, J. Pool, J. Tran, W. J. Dally,Learning Both Weights and Connections for Efficient Neural Networks
[16] Gustafson,J.L., Yonemoto,I.T., 2017. Beating Floating Point at its Own Game: Posit Arithmetic. Supercomputing Frontiers and Innovations 4, 2 (2017),
[17] Rusci, M., Fariselli, M., Alessandro Capotondi,A., Luca Benini,L., 2020. Leveraging Automated Mixed-Low-Precision Quantization for tiny edge microcontrollers. arXiv:2008.05124
[18] Pouransari,H., Tu, Z.,Tuzel,O. 2020. Least squares binary quantization of neural networks. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. 698–699
[19] Gustafson,J.L. A Generalized Framework for Matching Arithmetic Format to Application Requirements. In https://posithub.org/.
[20] Langroudi, HF., Karia,V., Pandit, T ., Kudithipudi, D,. TENT: Efficient Quantization of Neural Networks on the tiny Edge with Tapered Fixed Point arXiv 2021 arXiv:2104.02233
[21] H. Hu, R. Peng, Y.W. Tai, C.K. Tang Network Trimming: A Data- Driven Neuron Pruning Approach Towards Efficient Deep Architectures.
[22] Hinton, G., Vinyals O., Dean,J.Distilling the Knowledge in a Neural Network
[23] Cerutti,G., Prasad,R., Brutti,A., Farella,E. 2019. Neural Network Distillation on IoT Platforms for Sound Event Detection. In Interspeech. 3609–3613
[24] Tan,M., Bo Chen,B., Pang,R., Vasudevan,V., Sandler, M.,Howard,A and Quoc V Le,Q.V. MnasNet: Platform-Aware Neural Architecture Search for Mobile. In CVPR, 2019
[25] Ji Lin ,Wei-Ming Chen, Han Cai, Chuang Gan, Song Han MCUNetV2: Memory-Efficient Patch-based Inference for Tiny Deep Learning,.arXiv:2110.15352
[26] https://www.tensorflow.org/lite
[27] https://www.tensorflow.org/lite/microcontrollers
[28] Robert David,R., Jared Duke,J., AdvaitJain,A., et al TensorFlow Lite Micro: Embedded Machine Learning on TinyML Systems. arXiv:2010.08678
[29] https://www.edgeimpulse.com/
[30] https://stm32ai.st.com/nanoedge-ai/
[31] https://www.st.com/content/st_com/en/campaigns/stm32cube-ai.html
[32] https://utensor.github.io/website/
[33] https://microsoft.github.io/ELL/