TinyML: Current Progress, Research Challenges, and Future Roadmap

Muhammad Shafique
New York University Abu Dhabi, UAE
muhammad.shafique@nyu.edu

Theocharis Theocharides
University of Cyprus, Cyprus
ttheocharides@ucy.ac.cy

Vijay Janapa Reddy
Harvard University, USA
vj@eecs.harvard.edu

Boris Murmann
Stanford University, USA
murmann@stanford.edu

2021 58th ACM/IEEE Design Automation Conference (DAC) | 978-1-6654-3274-0/21/$31.00 ©2021 IEEE | DOI: 10.1109/DAC18074.2021.9586232

ABSTRACT

TinyML: tiny in size, BIG in impact!

This paper highlights the current progress, challenges and open research opportunities in the domain of tinyML, benchmarking, and emerging applications for Edge-AI.

KEYWORDS

TinyML, Machine Learning, Deep Learning, Benchmarking, Edge-AI.

1 INTRODUCTION

Tiny machine learning (tinyML) is a fast-growing field of machine learning technologies and applications including algorithms, hardware, and software capable of performing on-device sensor data analytics at extremely low power, hence enabling a variety of always-on use-cases and targeting battery-operated devices. tinyML systems are slowly adopted for multiple commercial applications and new systems on the horizon, and at the same time, significant progress is being made on algorithms, networks, and models. Further, what was initially considered low power applications, is now mainstream and commercially available. There is therefore a growing momentum demonstrated by the technical progress, ecosystem development and the need for benchmarking and evaluation methodologies. In this paper, we present an overview of the current state of the art, while we also identify challenges and opportunities. We also provide our vision for the road ahead.

The fundamental goal of tinyML is to utilize cross-layer design approaches and deploy ML inference to ultra-low-power devices such as microcontrollers or custom-designed circuits that consume under or about a milliWatt of power and can last for months if not years with a single recharge of battery. To achieve that, we must address several key challenges that ML inference faces when shifted towards the edge, including power consumption, latency, network throughput, reliability, robustness and data privacy. While these challenges must be addressed, obviously, the inference accuracy must remain acceptable at all levels, especially as the criticality of the applications vary significantly. Figure 1 presents an overview of the evolution of deep learning towards tinyML. Starting from the re-emergence of DNNs driven by advancements in hardware technology, the figure highlights the progress in accelerator design, use of approximate computing, and techniques for automated search of efficient models/architectures. It also highlights the emerging technologies and computing paradigms that can unveil orders of magnitude of efficiency gains.

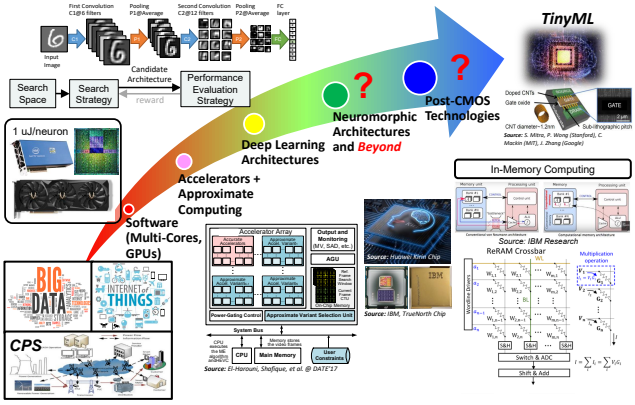

***Figure 1:* Evolution of Deep Learning towards tinyML**

Traditionally, neural networks and other relevant ML algorithms and models were developed to address specific challenges, mostly focusing on computer vision applications, such as object detection and classification. Over the last decade, however, we have seen a revolution unparalleled to what we have witnessed so far in terms of the applications that can benefit from all sorts of inference problems; from natural language processing, near-sensor intelligent data analytics to the abundance of smart devices that are expected to reach well over 50 billion over the next few years. Most of these devices are equipped with some sort of ML inference, such as always-on voice recognition, human activity recognition, and many more, while at the same time, the devices become smaller, faster, more energy-efficient, and most importantly cheaper, creating huge opportunities for a complete ecosystem that encapsulates several stakeholders across academia and industry, extending well beyond the traditional semiconductor industry, and at the same time augmenting our society economically, socially and of course scientifically. As a result, inference has migrated from the cloud towards the edge, creating several challenges, but at the same time, providing us with an abundance of opportunities.

This paper highlights four aspects of tinyML; cross-layer design approaches, the range of applications (and the associated challenges

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and opportunities), existing design and evaluation frameworks, and our view for the road ahead.

2 CROSS-LAYER DESIGN FLOW

Over the past few years, researchers have proposed various techniques for accelerating DNN inference. These techniques include both hardware-and software-level optimizations. At the hardware-level, specialized accelerators have been proposed for efficient processing of data, and at the software-level, Neural Architecture Search (NAS), pruning, and quantization techniques have been proposed to design compact and efficient models for resource-constrained IoT devices. Figure 2 presents a cross-layer design flow that synergistically connects all the optimization techniques for building ultra-efficient DNN-based systems. In the following paragraphs, we briefly describe each type of optimization.

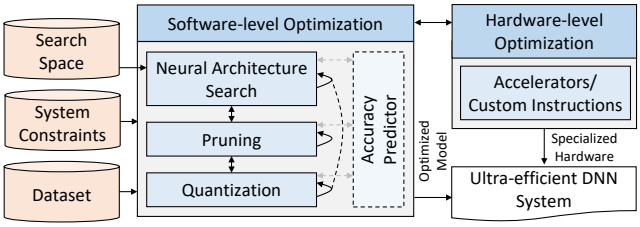

Figure 2: **Cross-Layer Design Flow**

Hardware Accelerators: In comparison to general-purpose hardware, specialized hardware can offer significant efficiency gains. As the dot product operation is the primary operation involved in DNN processing, researchers mainly focus on accelerating vector/matrix multiplication operations. Several accelerator designs have been proposed, based on different optimizations in the dataflow, to improve data localization and parallel processing [4]. Moreover, works like [11] propose to improve scheduling to reduce expensive off-chip memory accesses. Processing-in-memory (PIM) is an emerging trend in this direction, as it allows to perform in-place computations without moving data to the CPU [13][17]. Emerging technology-based devices like ReRAM-based devices [17][5] are becoming more-and-more capable of supporting high-precision in-memory operations that can offer orders of magnitude of efficiency gains.

Neural Architectural Search (NAS): It is the foremost technique for automating the process of designing specialized models that surpass the performance of handcrafted DNNs while meeting the system/user-defined constraints. The quality of the resultant model highly depends on the quality of the search space and the search strategy. Many recent works consider NAS as a pathfinding problem by jointly training sub-models rather than training each sampled sub-model from scratch to improve the efficiency of architecture search [14][2]. These techniques allow to consider larger search space due to their efficiency and thereby lead to better results.

Pruning: Iterative pruning techniques have shown remarkable potential for reducing the size and complexity of DNNs. These techniques can be classified into unstructured (or fine-grained) pruning and structured (or course-grained) pruning. Unstructured pruning techniques, in general, offer more reduction in model size; however, they can have an adverse effect on the system's performance when used for CPUs- and GPUs-based platforms [20]. Structured pruning, especially channel pruning, techniques are mainly used in the context of tinyML, as they can be easily integrated in a joint optimization framework and have low overheads [19].

Quantization: Besides pruning, a reduction in bit-width of model parameters also reduces the storage requirements of a DNN. It also enables the use of simplified operations, which further boosts efficiency gains. Mixed low precision quantization techniques that leverage AutoML to optimize the bit-width of each weight and activation tensor are commonly used to meet memory constraints [16][18].

Joint Optimization: A naive method of combining individual optimization techniques is to apply them one after the other, i.e., find an efficient design through NAS, apply a pruning technique, and then apply quantization. Such coupling of optimization techniques leads to suboptimal results. To achieve optimal results, joint optimization is required. However, it increases the search space by orders of magnitude, as the number of combinations of hyperparameters grows exponentially. Therefore, efficient search strategies have been proposed to reduce the search time and CO_2 emissions [19][2]. For example, APQ [19] employs efficient NAS and accuracy predictors to find an effective architecture and compression strategy. Moreover, while designing DNNs for highly resource-constrained MCUs, system-level constraints have to be considered. Towards this, system-model co-design approaches such as MCUNet [12] are highly effective.

3 APPLICATIONS

What fuels this unprecedented growth of the tinyML movement, is essentially the vast application spectrum made possible with the evolution of tinyML. Applications are driving the need for more efficient tinyML systems and further advancement across the whole design flow, and in turn, applications generate the necessary datasets, and the associated financial growth that powers further research advancements. Initially, tinyML applications focused mostly on vision, gradually integrating natural language processing and over the last couple of years, predictive modeling, data analytics and pattern recognition/classification. We present an overview of the application domains, providing a high-level discussion of the challenges and opportunities.

With regards to the application domains, tinyML is now successfully deployed over a large number of applications, ranging from healthcare, surveillance and security, smart things (ranging from sensors to cities), industrial monitoring and control, finance and administration, and many more domains that span across our every day lives and communities [8][15]. A small sample along with some challenges and opportunities are listed next:

- Healthcare: Wearables such as smart watches, that are equipped with an abundance of body mark sensors that measure vitals such as heart rate, blood oxygen concentration, activity, etc. and provide in real-time an accurate representation of the current health state of the user in a private, secure and reliable manner. Smart camera sensors able to monitor patients at their own premises, and quickly assist by notifying care-takers, real-time diagnostics and assistance, personalized and translational medicine, etc.
- Security and Surveillance: Camera sensors equipped with hardware that is able of performing relatively fast and accurate video analytics, applied in several domains such as tele-health, security and surveillance, services, monitoring, navigational devices, etc.

- Smart Things (IoT): Always-on wake modules (motion, voice and activity) such as intelligent assisting devices, translation and communication applications, natural language processing, personalized and user-customizable services, virtual and augmented reality, etc.
- Industrial Monitoring and Control: Manufacturing and assembly line quality assurance, real-time assembly machine diagnostics, etc.
- Smart and Secure Societies: Intelligent transportation systems, water distribution systems, smart energy grids, disaster relief and emergency response technologies, education, etc.

Obviously the above mentioned application areas are just a fraction of the tinyML spectrum, however, they have served as the foundations of advancement of tinyML, by creating an ecosystem which (a) provides the necessary services to society enabling acceptance by the society without the associated mistrust that usually surrounds artificial intelligence, (b) generates the necessary data sets that fuel the tinyML industry to improve and design more efficient products, and (c) stimulates via new opportunities further research and development.

4 FRAMEWORKS AND BENCHMARKING

4.1 Frameworks and Datasets

Among the most important challenges in advancing tinyML systems and technologies, are frameworks which enable seamless integration of all the algorithms and optimizations at all layers, developed by the community. Such frameworks are gradually starting to emerge, that target popular microprocessors and microcontrollers. Amongst those leading the effort is Tensoflow with its Lite edition (and Micro edition [6], geared towards embedded microcontrollers) that targets mobile and IoT devices. Similar frameworks include microTVM [3] (geared towards microcontrollers), TinyEngine (based on OpenGL Wrappers), and several up and coming frameworks. Such frameworks offer the designer the opportunity to easily design ML models, then optimize them for ultra-low power devices, without the user's prior knowledge of the hardware or even the optimization technique followed. This leads to the so called "democratization" of AI [9], that is, anyone wishing to develop AI applications, no longer requires access to expensive hardware, cloud services and expensive design tools and frameworks; instead, everyone can now develop tinyML applications that can run on cheap, off-the-shelf microcontrollers, without detained knowledge of the technology and any complex optimization techniques such as pruning and quantization. The rise of open-source software necessitated a framework revolution (new standards, tools, licenses, etc.) to overcome the problems facing large distributed teams working on enormous code bases. Today, machine learning (ML) builds atop this vibrant and creative open-source ecosystem and toolchain – leading frameworks such as TensorFlow, PyTorch and Keras are open and boast thousands of contributors around the globe. But ML is utterly unlike explicitly written software and relies heavily on data to implicitly define program behavior. Shifting the importance from explicit code to data introduces an even more significant and challenging problem: too little data. This translates to an important challenge concerns datasets, used for bulding accurate ML models and also for evaluating these models.Datasets are the rocket fuel of machine learning, and to maintain the tinyML movement, we rely on publicly available open-source datasets. To bring about equity in ML, widening access to open datasets is crucial. We must build low-cost efficient data engineering platforms that can enable ML for everyone.

4.2 Benchmarking

tinyML has been proven to play a key role in the design of next generation systems and as it is being deployed in a plethora of applications and systems, across industry-segments. This has not only led to its broad applicability in a wide range of applications, but has also increased the functional design and test requirements of tinyML systems. To reap higher yield from the tinyML applications and systems, real-time performance, power and system efficiency are key. Given the rapid advances, across industry and academia in this direction, benchmarking these ML applications and systems is critical. Such benchmarking will also help the community to analyze and compare the solutions in a fair manner [1].

A key challenge that we identify is our primary focus on metrics, and methods to assess tinyML, and more so on test chips and systems. This challenge brings together cross-layer aspects relating to deployment of tinyML on systems, architectural considerations, and design methodologies for a truly efficient deployment. In addition to tinyML-specific research for accelerator design, challenges relating to the co-existence of ML and non-ML hardware, robust tinyML architectures for different scales (IoT, embedded and microcontroller systems, bare metal sensors) are of additional interest.

To summarize, benchmarking tinyML systems presents us with several challenges and opportunities. For example, we are interested in metrics and methods of assessment of the tinyML Systems. We are also interested in cross-layer metrics and assessment of the efficiency of the deployment of such systems. We need to be able to benchmark and evaluate emerging design approaches, techniques and methodologies such as accelerators with approximate computing units, model compression techniques, reconfigurable and neuromorphic accelerators, hardware and software co-design in systems which contain inference hardware and non-ML hardware, and many more. Further, we need to be able to integrate these metrics and assessment methods into our design frameworks in a manner which is transparent to the designer, and enables the designed to holistically evalute a tinyML system.

5 FUTURE RESEARCH DIRECTIONS

TinyML is pushing the community to address problems that have not been investigated before at this scale. TinyML designers have to be aware about what data is being used, how it is used, and how it is communicated. We may be asked to build systems that are never networked, with software that is pre-installed and never updated. To further raise the bar, a tinyML device may never have access to network, implying that it cannot become a victim of a hostile attack, nor can it violate the user's privacy. However, such devices can never be updated with improved, more accurate models. There are many, many opportunities for tiny sensing devices with embedded machine learning. Almost all of these opportunities are in places where supplying power is not an option, and even when power is available, other constraints often require limited computing resources, small form factor, virtually zero cooling, and limited communication capabilities.

We face challenges across all the design spectrum; from the need to improve the design frameworks and possibly include custom-built hardware accelerators and reconfigurable fabric, to creating and having access to more and better datasets. At the same time, the technology that drives tinyML is also driven by the challenges associated with the end of Dennardian scaling and Moore's law; given that tinyML is pushing the limits with regards to power consumption, this becomes more important. Fortunately, any post-CMOS technology that can provide solutions, will also be adopted by tinyML. Moreover, ML inference requires extensive memory support, and memory technology challenges facing general computing today are also facing tinyML systems. A promising solution in terms of memory technology comes from the potential use of memristors, and also through emerging non-volatile memory technologies relying on solid-state devices [10].

While it's established that Von Neumann architectures are not efficient when it comes to ML applications, we must understand that sometimes, we may not have a choice due to various factors, cost being an important one. A small, cheap, microcontroller that relies on Von Neumann architecture, maybe all that we can utilize. That requires pushing our limits towards improving the overall design stack, by modifying our training approaches to consider the constraints of the targeted architectures; by utilizing better and customized datasets; by thinking outside the box and applying bio-inspired optimizations in efforts to minimize unnecessary computations; and by innovative optimization techniques that go beyond pruning and quantization. We also need to consider alternative neural network models, and customize them based on the application domain. For example, while deep neural networks have been proven quite efficient for video and image analytics, they are an overkill when it comes to a much simpler problem such as detecting a water leak from a home water monitoring device. Analog and spiking neural networks are also being investigated for tinyML applications [7].

At the same time, new computing paradigms are needed. In-memory computing for example has shown promising results for ML applications, but requires further research, understanding and development for being adopted by tinyML systems. Similarly, reconfigurable fabric has shown great potential as dedicated hardware accelerator for ML inference as it enables users to make the most effective use of the hardware with regards to each application. However, reconfigurable fabric for the scale of tinyML targeted systems, may not be currently an option due to its requirements in terms of power consumption, associated programming hardware and I/O and so on.

The abundance of challenges raised above however, provides similarly a lot of opportunities for research and development at all levels. Attempting to solve these problems requires multidisciplinary approaches that cover all layers, from software to algorithms to architecture and micro-architecture, and technology. Towards this end, neural architecture search (NAS) is becoming a major assistive tool. Even if it is still in its infancy stages, we believe that it may become a comprehensive potential solution that encapsulates the aforementioned constraints.

Last but not least, one of the most underrated challenges for tinyML involves the integration of the so called "ethical AI". As we discussed, the rapid growth of tinyML facilitated products and systems that are now integrated heavily within our society. We must ensure that these products are always in line with what we, as humans consider ethical. We must ensure that there is no bias in training and designing these systems, while attempting to reduce resources.

REFERENCES

[1] Colby R Banbury, Vijay Janapa Reddi, Max Lam, William Fu, Amin Fazel, Jeremy Holleman, Xinyuan Huang, Robert Hurtado, David Kanter, Anton Lokhmotov, et al. 2020. Benchmarking TinyML systems: Challenges and direction. *arXiv preprint arXiv:2003.04821* (2020).

[2] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. 2019. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791* (2019).

[3] Tianqi Chen, Thierry Moreau, Ziheng Jiang, Lianmin Zheng, Eddie Yan, Haichen Shen, Meghan Cowan, Leyuan Wang, Yuwei Hu, Luis Ceze, et al. 2018. {TVM}: An automated end-to-end optimizing compiler for deep learning. In *13th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 18)*. 578–594.

[4] Yu-Hsin Chen, Tien-Ju Yang, Joel Emer, and Vivienne Sze. 2019. Eyeriss v2: A flexible accelerator for emerging deep neural networks on mobile devices. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems* 9, 2 (2019), 292–308.

[5] Ping Chi, Shuangchen Li, Cong Xu, Tao Zhang, Jishen Zhao, Yongpan Liu, Yu Wang, and Yuan Xie. 2016. Prime: A novel processing-in-memory architecture for neural network computation in reram-based main memory. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 27–39.

[6] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, et al. 2020. Tensorflow lite micro: Embedded machine learning on tinyml systems. *arXiv preprint arXiv:2010.08678* (2020).

[7] Elisa Donati, Melika Payvand, Nicoletta Risi, Renate Krause, and Giacomo Indiveri. 2019. Discrimination of EMG signals using a neuromorphic implementation of a spiking neural network. *IEEE transactions on biomedical circuits and systems* 13, 5 (2019), 795–803.

[8] Igor Fedorov, Marko Stamenovic, Carl Jensen, Li-Chia Yang, Ari Mandell, Yiming Gan, Matthew Mattina, and Paul N Whatmough. 2020. TinyLSTMs: Efficient Neural Speech Enhancement for Hearing Aids. *arXiv preprint arXiv:2005.11138* (2020).

[9] Thomas K Finley. 2019. The democratization of artificial intelligence: one library's approach. *Information Technology and Libraries* 38, 1 (2019), 8–13.

[10] Can Li, Miao Hu, Yunning Li, Hao Jiang, Ning Ge, Eric Montgomery, Jiaming Zhang, Wenhao Song, Noraica Dávila, Catherine E Graves, et al. 2018. Analogue signal and image processing with large memristor crossbars. *Nature electronics* 1, 1 (2018), 52–59.

[11] Jiajun Li, Guihai Yan, Wenyan Lu, Shuhao Jiang, Shijun Gong, Jingya Wu, and Xiaowei Li. 2018. Smartshuttle: Optimizing off-chip memory accesses for deep learning accelerators. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 343–348.

[12] Ji Lin, Wei-Ming Chen, John Cohn, Chuang Gan, and Song Han. 2020. MCUNet: Tiny Deep Learning on IoT Devices. In *Annual Conference on Neural Information Processing Systems (NeurIPS)*.

[13] Onur Mutlu, Saugata Ghose, Juan Gómez-Luna, and Rachata Ausavarungnirun. 2020. A Modern Primer on Processing in Memory. *arXiv preprint arXiv:2012.03112* (2020).

[14] Hieu Pham, Melody Guan, Barret Zoph, Quoc Le, and Jeff Dean. 2018. Efficient neural architecture search via parameters sharing. In *International Conference on Machine Learning*. PMLR, 4095–4104.

[15] Davide Rossi, Francesco Conti, Andrea Marongiu, Antonio Pullini, Igor Loi, Michael Gautschi, Giuseppe Tagliavini, Alessandro Capotondi, Philippe Flatresse, and Luca Benini. 2015. PULP: A parallel ultra low power platform for next generation IoT applications. In *2015 IEEE Hot Chips 27 Symposium (HCS)*. IEEE, 1–39.

[16] Manuele Rusci, Alessandro Capotondi, and Luca Benini. 2020. Memory-Driven Mixed Low Precision Quantization for Enabling Deep Network Inference on Microcontrollers. In *Proceedings of Machine Learning and Systems (MLSys)*.

[17] Ali Shafiee, Anirban Nag, Naveen Muralimanohar, Rajeev Balasubramonian, John Paul Strachan, Miao Hu, R Stanley Williams, and Vivek Srikumar. 2016. ISAAC: A convolutional neural network accelerator with in-situ analog arithmetic in crossbars. *ACM SIGARCH Computer Architecture News* 44, 3 (2016), 14–26.

[18] Kuan Wang, Zhijian Liu, Yujun Lin, Ji Lin, and Song Han. 2019. Haq: Hardware-aware automated quantization with mixed precision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8612–8620.

[19] Tianzhe Wang, Kuan Wang, Han Cai, Ji Lin, Zhijian Liu, Hanrui Wang, Yujun Lin, and Song Han. 2020. APQ: Joint Search for Network Architecture, Pruning and Quantization Policy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[20] Jiecao Yu, Andrew Lukefahr, David Palframan, Ganesh Dasika, Reetuparna Das, and Scott Mahlke. 2017. Scalpel: Customizing dnn pruning to the underlying hardware parallelism. *ACM SIGARCH Computer Architecture News* 45, 2 (2017), 548–560.