The Challenges of TinyML Implementation: A Literature Review

Riya Adlakha and Eltahir Kabbar

https://doi.org/10.34074/proc.240120
Correspondence: Eltahir Kabbar, ekabbar@unitec.ac.nz

This publication may be cited as:

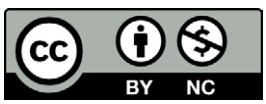
Adlakha, R., & Kabbar, E. (2024). The Challenges of TinyML Implementation: A Literature Review. In H. Sharifzadeh (Ed.), *Proceedings: CITRENZ 2023 Conference, Auckland, 27–29 September* (pp. 160–169). ePress, Unitec. https://doi.org/10.34074/proc.240120

Contact:
epress@unitec.ac.nz
www.unitec.ac.nz/epress/
Unitec
Private Bag 92025, Victoria Street West
Tāmaki Makaurau Auckland 1142
Aotearoa New Zealand

ISBN: 978-1-99-118344-6

ePress
Research with impact
Unitec

Unitec is a business division of Te Pūkenga – New Zealand Institute of Skills and Technology



ABSTRACT

This study aims to sensitise and summarise the tiny machine learning (TinyML) implementation literature. TinyML is a subset of machine learning (ML) that focuses on implementing ML models on resource-constrained devices such as microcontrollers, embedded systems, and internet of things (IoT) devices. A systematic literature review is performed on the works published in this field in the last decade. The key focus of this article is to understand the critical challenges faced by this emerging technology. We present five significant challenges of TinyML, namely, limited and dynamic resources, heterogeneity, network management, security and privacy, and model design. This article will be of interest to researchers and practitioners who are interested in the fields of ML, IoT and edge computing.

KEYWORDS

TinyML, edge computing, machine learning, IoT, microcontrollers

BACKGROUND

With the evolution in hardware, integrating enhanced machine learning (ML) models running on low-power embedded devices such as microcontrollers has become possible. This convergence of ML and embedded systems is called tiny machine learning (TinyML). TinyML enables assembling ML to hardware segments without processing the data in an external location. Processing and execution occur at the edge, making the applications close to the data source. This concept is called edge computing. In recent times, edge computing has become a shining example of an emerging technology. The technology has numerous applications in healthcare, finance, smart cities and transportation. TinyML delivers promising results when deployed on small edge devices that facilitate fast processing and data analysis without needing a server response. Figure 1 showcases an architectural framework of TinyML to compute and analyse data from multiple IoT devices into an edge device (microcontroller unit [MCU], for example), eliminating the requirement for processing at cloud servers.

The three basic tool-sets required to enable TinyML for processing and predicting results are hardware, software and libraries (Janapa Reddi et al., 2022). Various hardware platforms are TinyML aware, such as Arduino Nano 33 BLE Sense, Apollo3, Nicla Sense ME, ST IoT Discovery and Nordic Semi nRF52840 DK, to name a few. Most hardware operates at a flash memory of less than 1 MB and SRAM of less than 1 MB. Most of these are battery operated and have Li-Po on top of a DC power source. ARM Cortex M4 stands out as the most popular and widely used microprocessor. TensorFlow Lite (TFL) and uTensor are open-source software designed to run ML on MCUs and rapidly deploy IoT devices. Edge Impulse, another software, is a cloud service that enables TinyML by running ML models for edge devices. It supports AutoML for platforms at the edge. The capability of local execution is also supported with Python, C++ and SDK, along with numerous platforms such as smartphones endowed with Edge Impulse, to build training models.

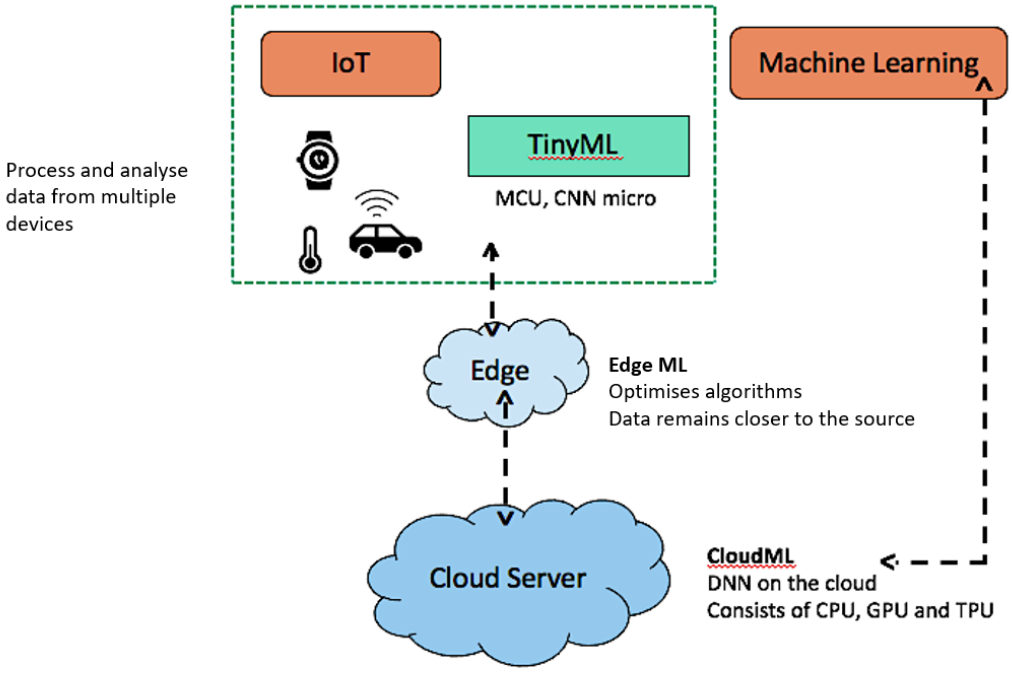


Figure 1. Enabling IoT devices with cloud computing, edge computing, and TinyML.

TinyML has a range of opportunities, given that the microcontrollers consume relatively less energy than the graphic processing units (GPUs), which enables the IoT devices that work with MCUs to be placed anywhere without plugging in (Sanchez-Iborra & Skarmeta, 2020). Given their low energy requirement, they can also be paired with devices driven by large batteries, allowing them to emerge as connected smart devices, such as smart watches and doorbells. In addition, MCUs are more cost effective compared to high-end processors, making the TinyML systems cost effective. With the TinyML model running on edge, the data is stored, processed and analysed internally rather than at an external server or cloud. These advantages increase data privacy, reducing the risk of compromising sensitive information. With complete independence, embedded devices increase autonomy by eradicating interference from outside sources. Thus, decisions and information can only be accessed and shared within the system (Soro, 2021).

One recent study that informs this research is Han and Siebert (2022), who presented a systematic literature review of TinyML focusing on five aspects: hardware, framework, datasets, use cases and algorithms. This study adds to their work by focusing on understanding the challenges associated with TinyML implementation in detail. We also explore the history, benefits and implementation of TinyML in the following sections.

HISTORY OF TINYML

TinyML emerged from the internet of things (IoT). It is a game changer, emphasising that big is not always better. Traditional methods included putting complex ML models into hardware for developing applications and products. The computing was performed on the cloud, which introduced the hurdle of latency and depended entirely on connectivity. All these factors made computing not only slower but also expensive and inefficient. The development of TinyML by Pete Warden bridged the gap between intelligence and embedded systems. More and more companies across the globe are moving towards TinyML. Azip, a leading company in AIoT (artificial intelligence for IoT) (“Artificial intelligence of things,” 2023), is making TinyML models available for high-performance and intelligent product solutions.

KEY ENABLER: TINYML AS-A-SERVICE

Conventional ML processing is powered by cloud providers that run on efficient CPUs, GPUs and tensor processing units (TPUs). However, embedded systems face constraints because of these devices' limited computation and processing powers to run complex ML models. TinyML as-a-Service (TinyMLaaS) is modelled to solve this fundamental problem for compact devices. Using an ML compiler, it aims to transform such models to fit the target device's resource size. It uses techniques to squeeze the model size; for example, quantising with fewer computing bits, pruning less critical parameters, and fusing multiple computational operators into one (Doyu et al., 2020). The framework uses ML compilers to generate optimised low runtime for popular ML models. TinyMLaaS also includes specialised ML models for embedded hardware accelerators, which are chip-manufacturer dependent.

In a typical TinyMLaaS ecosystem, an appropriate ML model is generated using lightweight machine-to-machine (LwM2M) software with an 'on-the-fly' model inferencing module. TinyMLaaS is a demand-driven cloud service that resolves privacy issues by keeping the data on-premise.

RESEARCH METHODS

To identify the critical challenges of TinyML implementation, a detailed literature review, guided by Brereton et al. (2007), was conducted using relevant keywords. The researchers then identified the articles that discussed TinyML implementation, eliminating any study out of the scope. Once the final list was devised, the researchers analysed and categorised the literature into five themes. The taxonomy of the literature categorisation is discussed in the following section.

RESULTS AND DISCUSSION

This section outlines the five key challenges and problems encountered in research and deployment pertaining to the numerous TinyML applications: limited and dynamic resources, heterogeneity, network management, security and privacy, and model design. We review the five main challenges and propose solutions in the following section. Figure 3 illustrates the taxonomy of obstacles TinyML faces.

Limited and Dynamic Resources

TinyML devices often have constrained energy, memory, and computing capabilities. These constraints present a barrier when deploying sophisticated machine-learning models that demand a lot of computational power.

Limited power: Maintaining accuracy throughout the range of TinyML devices is challenging because their power consumption can vary greatly. As articulated by R. Kallimani et al. (2023), the performance of the algorithms is severely hampered by the lack of power at the edge devices. Secondly, establishing what comes inside the scope of the power measurement is problematic when data pathways and preprocessing procedures differ dramatically between devices. TinyML frameworks may encounter energy mismanagement because sensors and other accessories are frequently attached to edge devices (Yelchuri & R, 2022). As a result, developing a power-efficient TinyML system remains a significant challenge.

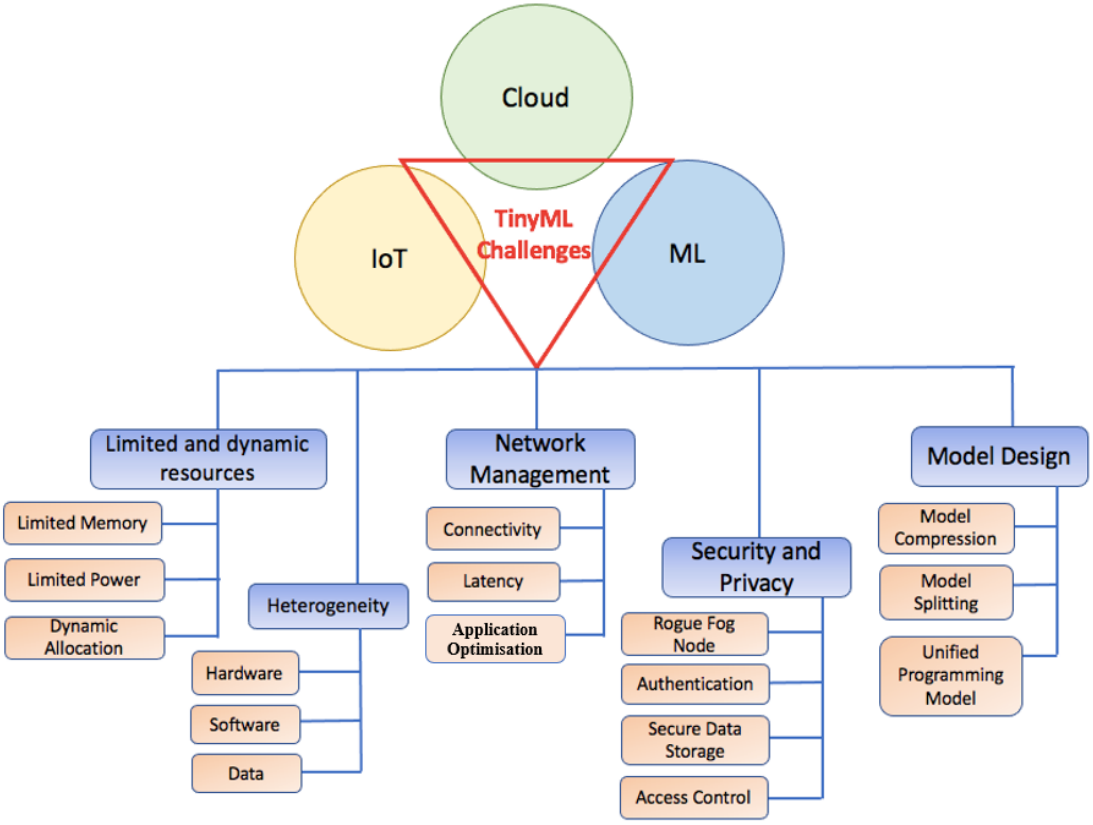


Figure 3: Taxonomy of TinyML challenges.

Limited memory: Due to the resource-constrained nature of tiny devices, one of the primary issues with TinyML is limited memory. Inference models used in traditional ML benchmarks have significantly greater peak memory requirements (in the range of GBs) than TinyML devices can offer (Banbury et al., 2020). Common ML systems frequently don't have resource use restrictions while running on workstations or the cloud. As a subset of edge devices, frugal gadgets, however, are much more limited. Most MCUs come with memory measured in kilobytes rather than megabytes and processors that run at megahertz rather than gigahertz in the most recent mobile phones. The restricted memory on TinyML devices leads to two key issues:

> **Catastrophic forgetting:** One common implication of limited memory in TinyML is catastrophic forgetting. This is a phenomenon in which the model forgets previously learned information while continuing to learn new information. Catastrophic forgetting can be a significant worry in the setting of TinyML, because memory and computing resources are constrained (Rajapakse et al., 2023). TinyML models are frequently used on hardware with limited memory, making it challenging to retain vast amounts of data for the model's retraining on earlier jobs.
>
> **Volatility of SRAM:** Due to resource limits, MCUs' primary memory (SRAM) spans from a few to a few hundred kilobytes. Neural networks stored in a flash as a C/C++ array are handled as frozen graphs, which means that any amendments to this graph are not permitted. As a result, many systems train the existing model entirely or partially in SRAM without putting it in flash memory. Because SRAM is volatile, any progress in training a model is lost when the MCU is reset or powered off.

Dynamic resource allocation: TinyML's dynamic resource allocation is a significant challenge, especially when dealing with resource-constrained devices with low memory, processing power and energy. Dynamic resource allocation entails efficiently managing and distributing the ever-dynamic resources to various TinyML system

activities or components. Currently, the edge platform suffers from a problem with dynamic resource allocation, necessitating the development of techniques/algorithms for analysing dynamic data. New methods that consider the varying computational power and specifications of processing entities, as well as the requirement for consistent portability among heterogeneous devices in the event of local and parallel processing, are required (Doyu et al., 2021).

Heterogeneity

Due to the considerable variation of hardware, software, and data characteristics across different tiny devices and platforms, heterogeneity can appear in various ways in TinyML. The models must be compatible with the target device's hardware, software and data architectures. It can be challenging to ensure compatibility across multiple microcontrollers and embedded systems, because each platform may have its limits and optimisations. Below are the three significant types of heterogeneity observed in TinyML models.

Hardware heterogeneity: TinyML models are implemented on various hardware architectures, from microcontrollers to embedded systems and IoT devices. These devices may feature different processor types (for example, ARM or RISC-V), clock speeds, memory sizes and specialised hardware accelerators. Each hardware platform may have its own set of constraints and optimisations, making it challenging to create models that perform well across many devices. Furthermore, a significant challenge is the issue of standardising performance findings across different implementations (Banbury et al., 2020).

Software heterogeneity: TinyML's software heterogeneity refers to the variety of software components, frameworks, libraries and runtime environments utilised for deploying ML models on resource-limited devices. This variation is caused by changes in operating systems, ML frameworks and other software dependencies among small devices. The constraints of software heterogeneity include maintaining TinyML application consistency and performance, compatibility issues and program capability restrictions. Tiny devices frequently use several operating systems or RTOS, each with its own memory management and language support, which influences application behaviour. Furthermore, the availability and compatibility of ML frameworks may differ, resulting in model development and deployment inconsistencies. Software libraries, inference engines and compiler variations further complicate TinyML development and deployment.

Data heterogeneity: Handling data heterogeneity is difficult, because it requires careful consideration of data pretreatment, augmentation, and adaption approaches to maintain model resilience and generalisation. TinyML models are trained using data from various sources, such as sensors or edge devices, resulting in multiple data types and formats. Model accuracy depends on managing data quality and dealing with noisy or missing data (Kallimani et al., 2023). Variations in data distribution between devices can also impact model performance during deployment. TinyML application portability (Lakshman & Eisty, 2022) is affected by data heterogeneity, because models must be flexible to multiple data distributions, necessitating preprocessing, transfer learning, and robust generalisation techniques to maintain consistent performance across different devices and contexts.

Network Management

Network management is essential for dependable and effective communication among resource-constrained devices, edge nodes and cloud servers. Three challenges related to maintaining a consistent network for TinyML are discussed in this section.

Connectivity: Gateways in the architecture must be connected to the internet. Provisioning such connectivity necessitates both capital expenditure (CAPEX) and operating expenditure (OPEX), as well as administrative overheads associated with infrastructure maintenance (Zaidi et al., 2022). Local inference capabilities lessen reliance on connectivity, allowing for providing services in places where internet connectivity is sporadic or non-existent. Managing the network connectivity of small devices is critical for data transmission, model updates and contact

with cloud services. Tiny devices may have different communication capabilities, such as wi-fi, Bluetooth, Zigbee or cellular connectivity, each with limitations and trade-offs.

Latency: Real-time inference is critical for many applications, including robotics and voice-based assistive technology. For example, real-time inference is essential in responding to user commands and operating in a voice-controlled home automation system.

Sending data to the cloud for inference or training may result in network delays, rendering it unsuitable for time-sensitive, interactive applications. For example, offloading sensor data to a cloud server for processing in a real-time robotics application would result in considerable end-to-end latency (Chen & Ran, 2019). Applications with low-latency requirements can accomplish real-time inference and provide smooth user experiences by leveraging TinyML and edge computing, which makes them particularly successful in dynamic and time-critical contexts.

Application optimisation: Because of the nature of the computing platforms, applications deployed in edge-cloud computing settings confront a slew of challenges, including restricted bandwidth, unreliability and heterogeneity of wireless connections, and computation offloading. Changing workloads across different components has an impact on application performance. Maintaining quality of service necessitates elasticity and remedial skills. Thang Le Duc et al. have discussed techniques such as load balancing, application scaling and migration to achieve these needs for reliable resource supply in edge-cloud computing (Le Duc et al., 2019).

Security and Privacy

The leakage of private information, such as data, location, or usage, is a crucial challenge for end users using services such as cloud computing, wireless network, or IoT. In addition to encryption of private data, secure proxies required for rendezvous, communication and access control are currently not popular in TinyML implementation. Yi et al. (2015) have extensively presented the challenges of security and privacy problems related to TinyML implementation. This section discusses these challenges.

Security: As discussed in the work presented by Lopez et al. (2015), the coexistence of malicious and trusted nodes in distributed edge-based overlays is considered by edge-centric computing. This will necessitate using secure routing, redundant routing, trust topologies and past peer-to-peer research on this novel set. Below are the problems faced in obtaining a secure TinyML framework (Yi et al., 2015):

> **Rogue fog node:** A rogue fog node is a fog device or instance that lures end users into connecting by pretending to be genuine. This can lead to the threat of cyber-attacks, as the system's security will be compromised if connected to a fake node. Due to the dynamic instance creation and deletion in TinyML, it is difficult to block or blacklist the false nodes, making them prone to such attacks.
>
> **Authentication:** Authentication is a prime focus for TinyML, as many end users are connected to its devices and applications. Each node requires authentication, which poses a challenge to TinyML. In contrast to the conventional public key infrastructure authentication, face, fingerprint and touch-based authentication would be beneficial to transform the security standards used in TinyML.
>
> **Secure data storage:** Edge computing involves outsourcing user data and transferring user control over data to the edge, which has the same security risks as cloud computing. It is challenging to guarantee data integrity, because the outsourced data may be deleted or erroneously manipulated. Furthermore, unauthorised parties might exploit the uploaded data to threaten the TinyML systems.

Privacy: Data transmission to the cloud raises privacy issues for users who own that data or whose actions are being traced in the data. This leads to the mental barrier of exposing sensitive and confidential information to the cloud without learning how the data will be consumed. Data, usage and location privacy are often the most challenging aspects of TinyML. Edge nodes fetch sensitive information produced by IoT devices and usually lack privacy-preserving methods. These methods cannot be deployed directly because edge computing has no reliable third party. Access control has been shown to be a reliable solution for maintaining user privacy while ensuring system

security. While access control in cloud computing is typically performed cryptographically for outsourced data, building access control that spans client, edge and cloud in fog computing will be challenging while continuously meeting the design and resource limitations.

Model Design

Machine learning researchers frequently concentrate on developing models with a smaller number of parameters in the deep neural network (DNN) model in order to lower memory usage and latency while achieving good accuracy when designing DNN models for resource-constrained devices. Major challenges that arise in the implementation of TinyML are model compression, model splitting and a unified programming model.

Model compression: To enable running ML models such as DNNs on the edge, model compression is crucial to compress the existing models while preserving high accuracy at the same time. Compression techniques such as parameter quantisation and pruning are useful in doing so (Chen & Ran, 2019). By changing the precision of DNN parameters from floating-point numbers to low-bit numbers, parameter quantisation reduces the computational load by avoiding costly floating-point multiplications. Parameter pruning, on the other hand, reduces the size and efficiency of the DNN model while maintaining acceptable performance by removing the least significant parameters, which are frequently the ones close to zero.

Model splitting: A DNN model can be processed using the model splitting technique across many computer devices or resources. This method is very helpful in context with limited resources, such as edge devices and microcontrollers. The method entails breaking up the initial DNN into more manageable sub-models or layers. These sub-models can then be installed and run on distinct edge devices with constrained memory and processing power. The goal of model splitting is to maximise inference efficiency and resource utilisation without adversely affecting the performance of the model as a whole.

Unified programming model: TinyML's unified programming model addresses the heterogeneity by offering a consistent and standardised framework for constructing machine learning models that can operate quickly on a diverse variety of edge devices. Automatic model optimisation and quantisation techniques are used to reduce huge models into compact forms suitable for edge devices. This approach emphasises low-latency and real-time inference, ensuring that TinyML models can accomplish tasks with little delay, making them suited for robotics, IoT, and other real-time use cases. TinyML intends to lower the barrier of entry for developers, speed the creation of TinyML applications, and encourage the use of machine learning in resource-constrained situations by providing a consistent programming model.

CONCLUSION

TinyML can transform multiple technologies such as cloud computing, IoT and ML. The open challenges identified in this study make TinyML unfeasible for some industries, such as finance and healthcare, due to the high demand for security and privacy. The results of our study indicate that limited resources, security and privacy are the most significant difficulties TinyML implementation faces. While the heterogeneity of hardware, software and data stands out as a challenge, there is a huge potential for this technology to shape the future of machine learning.

REFERENCES

Artificial intelligence of things. (2023, September 18). In *Wikipedia*. https://en.wikipedia.org/wiki/Artificial_intelligence_of_things

Banbury, C. R., Reddi, V. J., Lam, M., Fu, W., Fazel, A., Holleman, J., Huang, X., Hurtado, R., Kanter, D., Lokhmotov, A., Patterson, D., Pau, D., Seo, J., Sieracki, J., Thakker, U., Verhelst, M., & Yadav, P. (2020). *Benchmarking TinyML systems: Challenges and direction*. http://arxiv.org/abs/2003.04821

Brereton, P., Kitchenham, B. A., Budgen, D., Turner, M., & Khalil, M. (2007). Lessons from applying the systematic literature review process within the software engineering domain. *Journal of Systems and Software*, *80*(4), 571–583. https://doi.org/10.1016/j.jss.2006.07.009

Chen, J., & Ran, X. (2019). Deep learning with edge computing: A review. *Proceedings of the IEEE*, *107*(8). https://doi.org/10.1109/JPROC.2019.2921977

Doyu, H., Morabito, R., & Brachmann, M. (2021). A TinyMLaaS ecosystem for machine learning in IoT: Overview and research challenges. *2021 International Symposium on VLSI Design, Automation and Test (VLSI-DAT)*. https://doi.org/10.1109/VLSI-DAT52063.2021.9427352

Doyu, H., Morabito, R., & Höller, J. (2020). *Bringing machine learning to the deepest IoT edge with TinyML as-a-Service**. https://www.researchgate.net/publication/342916900

Han, H., & Siebert, J. (2022). TinyML: A systematic review and synthesis of existing research. *4th International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*, 269–274. https://doi.org/10.1109/ICAIIC54071.2022.9722636

Janapa Reddi, V., Plancher, B., Kennedy, S., Moroney, L., Warden, P., Suzuki, L., Agarwal, A., Banbury, C., Banzi, M., Bennett, M., Brown, B., Chitlangia, S., Ghosal, R., Grafman, S., Jaeger, R., Krishnan, S., Lam, M., Leiker, D., Mann, C., … Tingley, D. (2022). Widening access to applied machine learning with TinyML. *Harvard Data Science Review*, *4*(1). https://doi.org/10.1162/99608f92.762d171a

Kallimani, R., Pai, K., Raghuwanshi, P., Iyer, S., & L'opez, O. (2023). TinyML: Tools, applications, challenges, and future research directions. *arXiv*. https://doi.org/10.48550/ARXIV.2303.13569

Lakshman, S. B., & Eisty, N. U. (2022). Software engineering approaches for TinyML based IoT embedded vision: A systematic literature review. *4th International Workshop on Software Engineering Research and Practice for the IoT (SERP4IoT '22)*, 33–40. https://doi.org/10.1145/3528227.3528569

Le Duc, T., Leiva, R. G., Casari, P., & Östberg, P. O. (2019). Machine learning methods for reliable resource provisioning in edge-cloud computing. *ACM Computing Surveys (CSUR)*, *52*(5). https://doi.org/10.1145/3341145

Lopez, P. G., Montresor, A., Epema, D. H., Datta, A., Higashino, T., Iamnitchi, A., Barcellos, M., Felber, P., & Riviere, E. (2015). Edge-centric computing: Vision and challenges. *Computer Communication Review (CCR)*, *45*(5), 37–42. https://iris.unitn.it/retrieve/handle/11572/114780/429278/ccr15.pdf

Rajapakse, V., Karunanayake, I., & Ahmed, M. (2023). Intelligence at the extreme edge: A survey on reformable TinyML. *ACM Computing Surveys*, *55*(13s), 1–30. https://doi.org/10.1145/3583683

Sanchez-Iborra, R., & Skarmeta, A. F. (2020). TinyML-enabled frugal smart objects: Challenges and opportunities. *IEEE Circuits and Systems Magazine*, *20*(3), 4–18. https://doi.org/10.1109/MCAS.2020.3005467

Soro, S. (2021). TinyML for ubiquitous edge AI. *arXiv*. https://doi.org/10.48550/arXiv.2102.01255

Yelchuri, H., & R, R. (2022). A review of TinyML. *arXiv*. https://doi.org/10.48550/arXiv.2211.04448

Yi, S., Qin, Z., & Li, Q. (2015). Security and privacy issues of fog computing: A survey. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 9204, 685–695. https://doi.org/10.1007/978-3-319-21837-3_67

Zaidi, S. A. R., Hayajneh, A. M., Hafeez, M., & Ahmed, Q. Z. (2022). Unlocking edge intelligence through tiny machine learning (TinyML). *IEEE Access*, *10*, 100867–100877. https://doi.org/10.1109/ACCESS.2022.3207200

AUTHORS

Riya Adlakha is an aspiring researcher currently pursuing a Postgraduate Diploma in Applied Technology at Unitec, with a keen interest in diverse fields such as machine learning, data sciences, data and business analytics, and IoT. Her career as a Business Analyst at Amazon, India, served as the starting point for her foray into data-driven storytelling. Riya's work is an invitation to other enthusiasts to join them in their pursuit of knowledge and discovery. She is passionate about examining the transforming potential of research and is dedicated to making the difficult understandable, and works to close the knowledge gap in all of her endeavours. riya.work04@gmail.com

Dr Eltahir Kabbar is a Senior Lecturer at the School of Computing Electrical and Applied Technology, Unitec. His research interests include diffusion of innovation (DOI), technology adoption, and e-government systems. His publications include a book chapter, journal articles and international conferences. ekabbar@unitec.ac.nz