Hands-on TinyML

Harness the power of Machine Learning on the edge devices

Rohan Banerjee

www.bpbonline.com

First published: 2023

Published by BPB Online
WeWork
119 Marylebone Road
London NW1 5PU

UK | UAE | INDIA | SINGAPORE

ISBN 978-93-55518-446

www.bpbonline.com

Dedicated to

My Beloved Little Niece:
Arjama

About the Author

Rohan Banerjee is a practicing data scientist having more than 12 years of relevant industry experience. He completed his M.Tech from IIT Kharagpur in 2011. His areas of interest include advanced data science, machine learning, embedded machine learning, digital signal and image processing. Rohan is currently associated with Baker Hughes Company. Before that, he was with TCS Research, Tata Consultancy Services where he published more than 40 technical papers in international conferences, journals, and also contributed in enhancing their intellectual property portfolio. Rohan is an avid reader of contemporary literature, a traveler, and a quiz enthusiast.

About the Reviewers

- **Tushar Chugh** is a machine learning engineer at Google, focusing on search ranking. With a background in robotics from Carnegie Mellon University, he previously contributed to GM's self-driving car perception systems. Passionate about applied machine learning, deep learning, and computer vision, Tushar has experience developing innovative technologies at Qualcomm and Microsoft in ML and tech domains.
- **Yogesh M Iggalore** has over a decade of experience in product development and is actively involved in all the disciplines of product architecture, hardware design, firmware development, testing, and cloud integration. He is currently interested in product development in TinyML.

Acknowledgements

There are many people I want to express my gratitude to. First and foremost, I would like to sincerely thank my family members for their unwavering support and encouragement throughout my journey — I could have never completed this book without their support.

I am grateful to various online resources, blogs, and materials that enriched my learning in order to write the book. I would also like to acknowledge the valuable feedbacks of my colleagues and co-workers during many years working in the tech industry. I am particularly grateful to Mr. Avik Ghose from TCS Research for providing me the opportunity to learn and work on TinyML. I gratefully acknowledge the effort of Mr. Tushar Chugh and Mr. Yogesh M Iggalore for their technical scrutiny and suggestions for improving the quality of this book.

My sincere gratitude also goes to the team at BPB Publication for being supportive enough to provide me quite a long time to finish the book and also for all the valuable editorial reviews.

Finally, I would like to thank all the readers who have taken an interest in the book. Your encouragement has been invaluable.

Preface

TinyML is an emerging trend in machine learning, that aims at deploying complex machine learning and neural network models on low-powered tiny edge devices and microcontrollers. Modern deep learning algorithms are computationally expensive and result in large model size. They are often hosted on dedicated servers having enormous computing resources. As users, we generate the data at our end and send them via the internet to process remotely. Owing to the limitations in network bandwidth, roughly 10% of all our data can be sent over the internet. Processing of the data on the edge can revolutionize the current paradigm. Thanks to TinyML, large machine learning models can be shrunk in order to effectively deploy on smaller devices having few hundred kilobytes of RAM and few megabytes of flash memory. Such devices can operate 24x7 with a minimum power consumption. Moreover, being entirely offline, the applications not only consumes zero network bandwidth, but also preserves user privacy.

TinyML is going to be the next big thing in machine learning. Major tech giants are heavily investing in standardizing the hardware and software stack. In this book, we cover the basic concepts of TinyML through practical coding examples to enable the readers to learn the basic concepts of TinyML and develop their own applications. Rather than discussing every single mathematical concept behind the machine learning algorithms, the book primarily focuses on end-to-end application development through coding examples. The projects covered in this book are implemented in open-source software commonly used in industry and academics.

This book is divided into 10 chapters. The details are listed as follows.

Chapter 1: Introduction to TinyML and its Applications – covers the basic concept of EdgeML and TinyML, their potential applications, and challenges. It briefly covers the hardware and software platforms required to create TinyML. We also discuss the process flow of creating TinyML applications.

Chapter 2: Crash Course on Python and TensorFlow Basics – covers the basics of Python which is now the de facto programming language in machine learning for both research and creating production ready software. We start with the basic concepts of Python along with various libraries such as NymPy, Matplotlib. The later part of the chapter covers the key aspects of TensorFlow. TensorFlow is a free

and open-source software library for machine learning and neural networks. The chapter briefly covers some of the fundamental concepts of TensorFlow through coding examples.

Chapter 3: Gearing with Deep Learning – briefly talks about neural networks. We begin with the concept of a simple Artificial Neural Network (ANN), various activation functions, and backpropagation to learn the weights. Later, we talk about Convolutional Neural Network (CNN), a popular deep neural network architecture used in modern image processing and computer vision applications.

Chapter 4: Experiencing TensorFlow – guides us to develop our first neural network using TensorFlow and Keras. Keras is a set of deep learning APIs in Python, running on top of TensorFlow, providing high level of abstraction in developing large neural networks. We begin with implementing a simple ANN for classification of handwritten digit images. Later, we implement our first CNN architecture.

Chapter 5: Model Optimization Using TensorFlow – talks about how a large TensorFlow model can be effectively compressed in order to deploy on smaller edge devices using TensorFlow Lite. We create a base CNN model using TensorFlow and convert it into the lighter TFLite model. The chapter also covers TensorFlow Model Optimization Toolkit, a software library for optimizing large neural networks for easy deployment and execution. We learn about different model optimization techniques, such as quantization, weight pruning and weight clustering through coding examples using the APIs provided by TensorFlow Model Optimization Toolkit. Finally, we summarize the impact of various optimization techniques on the base CNN in terms of model size and accuracy.

Chapter 6: Deploying My First TinyML Application – guides us to create the first real TinyML application on Raspberry Pi, a commercially available low-powered edge device. We create a neural network for classification of offline images on Raspberry Pi. The chapter covers two important topics, MobileNet and transfer learning. MobileNet is an optimized neural network architecture specially designed for low-powered mobile edge devices. Transfer learning is another interesting concept in machine learning, where we can reuse a pre-trained model on a new

problem. Transfer learning is particularly useful when we do not have sufficient training data to create a model from scratch.

Chapter 7: Deep Dive into Application Deployment – guides us to implement a more practical TinyML application of real-time on-device person identification from live video stream recorded by a camera. The application is again deployed on Raspberry Pi using various open-source software.

Chapter 8: TensorFlow Lite for Microcontrollers – covers the basics of TensorFlow Lite for Microcontrollers, a highly optimized software tool for porting TensorFlow models on low-powered microcontrollers. We implement a simple neural network that modulates the voltage output of a linear potentiometer and successfully deploy it on Arduino Nano 33 BLE Sense, the recommended microcontroller board for creating TinyML applications.

Chapter 9: Keyword Spotting on Microcontrollers – guides us implementing an on-device speech recognition application. Keyword spotting is an important requirement in modern voice assistant services, such as Amazon's Alexa or Apple's Siri. In this chapter, we implement a simple keyword spotting application on Arduino. We first implement a basic keyword detection system using TensorFlow to understand the key concepts of audio processing. Later, we implement a real keyword spotting application using Edge Impulse, a free software platform for designing end-to-end TinyML application and deploy on an Arduino device with minimum code writing.

Chapter 10: Conclusion and Further Reading – summarizes our learnings in the book and covers some recent trends in TinyML.

Code Bundle and Coloured Images

Please follow the link to download the
Code Bundle and the ***Coloured Images*** of the book:

https://rebrand.ly/q35pmfm

The code bundle for the book is also hosted on GitHub at **https://github.com/bpbpublications/Hands-on-TinyML**. In case there's an update to the code, it will be updated on the existing GitHub repository.

We have code bundles from our rich catalogue of books and videos available at **https://github.com/bpbpublications**. Check them out!

Errata

We take immense pride in our work at BPB Publications and follow best practices to ensure the accuracy of our content to provide with an indulging reading experience to our subscribers. Our readers are our mirrors, and we use their inputs to reflect and improve upon human errors, if any, that may have occurred during the publishing processes involved. To let us maintain the quality and help us reach out to any readers who might be having difficulties due to any unforeseen errors, please write to us at :

errata@bpbonline.com

Your support, suggestions and feedbacks are highly appreciated by the BPB Publications' Family.

Piracy

If you come across any illegal copies of our works in any form on the internet, we would be grateful if you would provide us with the location address or website name. Please contact us at **business@bpbonline.com** with a link to the material.

If you are interested in becoming an author

If there is a topic that you have expertise in, and you are interested in either writing or contributing to a book, please visit **www.bpbonline.com**. We have worked with thousands of developers and tech professionals, just like you, to help them share their insights with the global tech community. You can make a general application, apply for a specific hot topic that we are recruiting an author for, or submit your own idea.

Reviews

Please leave a review. Once you have read and used this book, why not leave a review on the site that you purchased it from? Potential readers can then see and use your unbiased opinion to make purchase decisions. We at BPB can understand what you think about our products, and our authors can see your feedback on their book. Thank you!

For more information about BPB, please visit **www.bpbonline.com**.

Join our book's Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com

Table of Contents

Chapter 1

Introduction to TinyML and its Applications

Introduction

The year 2022 brought **Artificial Intelligence** (**AI**) to a new level of endless possibilities through the applications of a Generative Pre-training Transformer, ChatGPT. ChatGPT is an AI language model that uses advanced machine learning models to generate human-like text. Needless to say that AI and Machine Learning are hot topics in modern technology. We are living in a world where we are using them everywhere in our day-to-day activities, knowingly or unknowingly. Although the two terms, AI and Machine Learning, are often used synonymously, there are subtle differences between them.

Artificial Intelligence is the science of imbibing human-like intelligence in machines via computer programming to make them behave like humans and, therefore, solve real human problems. In short, through AI, a computer system tries to simulate human reasoning using maths and logic. AI can be applied to many different sectors and industries, including but not limited to healthcare industries for suggesting drug dosage, banking and finance sector for identifying suspicious activities, self-driving cars, and so on. Machine Learning is a subset of AI, where a machine is programmed to learn from past experience in order to predict the outcome of a future event without explicitly being programmed for that. The concept is analogous

to the way we all learn. We gather knowledge from various mediums, for example, reading books, guidance and advice from parents and teachers, and from our day-to-day experiences. Based on that knowledge, we can act in a new situation, like writing in an exam. Machine Learning has many sub-fields. Deep learning is a subset of machine learning that simulates the behavior of the human brain through a specially designed architecture called the **Artificial Neural Network** (**ANN**). Deep learning can deal with large unstructured data with minimum human interaction, and hence, has gained lots of attention in recent times. In today's world, we are immensely dependent on Machine Learning and deep learning techniques in our daily activities. Our smartphones are loaded with numerous applications directly using machine learning. When you click a photo on your smartphone and upload it on social media, it automatically detects various objects in the photo, the place where it is captured, and even suggests you to tag your friends who are present in the photo. All these happen thanks to some Machine Learning applications such as object detection, geo-locating mapping, and face recognition. Similarly, while searching for news on the internet, we often opt for searching by voice. It falls under speech recognition, another popular application of Machine Learning. When you shop at an e-commerce site, you are often surprised at how the website accurately knows your preference and recommends you accordingly. This also happens because of some Machine Learning algorithms that learn from your past purchase history and recommend you accordingly.

Machine Learning, particularly deep learning algorithms, are computationally expensive and often require a powerful hardware accelerator like a **Graphics Processing Unit** (**GPU**) to operate. Such applications typically run on large computers and dedicated data centers. However, the data is generated by the users, on their personal devices like smartphones. In the traditional approach, user data is sent to a dedicated remote server machine via the internet for running the machine learning jobs. However, is this practically feasible? We generate gigabytes of data every day. Is it practically possible to send all these data to a remote system? That would consume enormous network bandwidth. What about the network delay? Recently, there has been a trend called Edge ML that aims at shrinking the machine learning models to run them on edge devices like our smartphones.

In this book, we are going to introduce TinyML, which takes Edge ML one step further and allows it to run machine learning algorithms even on the smallest microcontrollers. It is a subset of applied machine learning that fits large machine-learning and deep learning models to tiny embedded systems running on microcontrollers or other ultra-low power processors. Technically, embedded systems need to be powered by less than 1 milliwatt so that they can run for months, or even years, without needing

to replace batteries. TinyML is one of the hottest trends in the field of embedded computing. Research suggests that global shipments of TinyML devices will reach 2.5 billion by 2030. Several tech-giants are currently working on chips and frameworks that can be used to build more systematized devices in order to standardize the field. TinyML is expected to cause ground-breaking advancements in complex machine learning tasks to solve our day-to-day problems.

Structure

In this chapter, we will discuss the following topics:

- Brief overview of Machine Learning
  - Supervised Machine Learning
  - Unsupervised Machine Learning
- Machine Learning and Deep Learning
- Edge computing and TinyML
- Applications of TinyML
- Hardware for deploying TinyML
- Software for TinyML
- Process flow of creating TinyML applications
- Prerequisites—hardware and software

Objectives

TinyML is a subfield of modern Machine Learning that aims at compressing large Machine Learning models to deploy them on low-powered, low footprint, resource-constrained edge devices and microcontrollers. Though it sounds amazing, deploying a large Machine Learning model on a smaller edge device is not easy. A reduction in model size often comes with a degradation in performance. Hence, rather than compression, the main focus is on optimizing a model for a target device.

This book is intended to cover the fundamentals of TinyML through practical projects so that the readers can have an in-depth idea of how TinyML works with some hands-on experience. The primary objective of this book is to make you familiar with TinyML programming using open-source software packages so that you can create your own TinyML projects from scratch. Rather than detailing the underlying complex mathematics involved in machine learning and deep neural network algorithms, our key focus is to learn the programming aspects using practical

examples. However, interested readers are encouraged to learn the mathematical aspects from various available resources for a better understanding of how various machine learning algorithms were actually derived.

In this introductory chapter, we will briefly cover the fundamentals of TinyML. We will begin with the key aspects of machine learning and deep learning. Then, we will talk more about TinyML as a technology, its applications, and the hardware and software recommended to create real TinyML projects.

Brief overview of Machine Learning

Before starting with TinyML, we should have some fundamental concepts of machine learning. Machine Learning is a branch of artificial intelligence that focuses on developing computer algorithms based on data to imitate the human learning process. Now, the question arises: where do we need machine learning?

Suppose you have measured the temperature of the day as 25 degrees in the Celsius scale using a thermometer and wish to convert it to the Fahrenheit scale. There is a well-known formula for doing the conversion, which is given by: $\frac{C}{5} = \frac{F-32}{9}$. You can simply put C = 25 in the equation and get *F* as 77. Here, you have both the data and the rule that relates to the data. However, the situation is quite different in real-life applications where you have data, but you often do not have a known mathematical formula to relate them. For example, suppose you want to predict the price of a house in a suburban locality in Delhi. What would you typically do? There is no known mathematical formula to solve the problem. Machine learning can help us to do so. Machine learning is all about data. If we have the right amount of data, it can help us to find suitable relations between them. In order to predict the price of a new house, you first need to gather certain information for a few other houses in the same locality to empirically estimate the price of a new house. For example, you could collect the area of those houses, the number of rooms, the distance of the properties from the main road, and so on. You also need to collect the current prices of those houses. Here, the price of the house can be considered as a dependent variable, which is determined by the independent variables such as the area of the house, number of rooms, distance from the main road, and so on. The dependent variables are also called the target values or labels in some cases, and the independent variables are called as features. With machine learning, we can build a model to find the relationship between the dependent and the independent variables. The resulting model can predict the price of another house if the features are provided as input.

Similarly, suppose a pharmaceutical company is planning to launch a new blood pressure-controlling medicine. Before that, they want to investigate the impact of that medicine on people. If the recorded stable blood pressures after the intake of certain dosages of the medicine are experimentally noted on a diverse group of people, a machine learning model can be created to predict what should be the ideal medicine dose for a patient having a certain range of blood pressure.

Machine learning approaches primarily fall into two categories, **supervised** and **unsupervised** machine learning.

Supervised Machine Learning

Supervised machine learning approaches take both features and corresponding targets or labels as input to create a model which can be used to predict the target value of a new unseen example data using the features. Supervised machine learning algorithms are commonly used for classification and regression. In classification, a machine learning model is designed to predict a discrete class label from the features, for example, predicting the presence of a cat or a dog in an image, or identifying numerical digits from handwritten expressions. In regression, a machine learning model predicts a continuous value, for example, predicting the price of an asset or predicting the salary of a person. *Figure 1.1* provides a basic block diagram showing various components of the supervised learning approach:

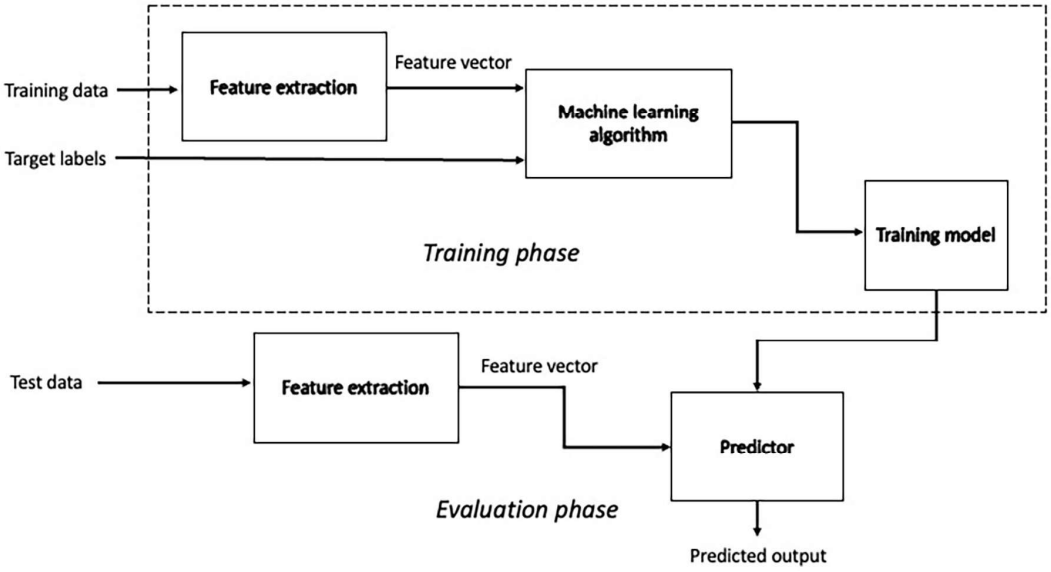


***Figure 1.1**: Block diagram of supervised learning approach*

Supervised learning involves two phases, training and evaluation. During training, it takes labeled training data as input and tries to create a mathematical relationship between them by adjusting some parameters, which are called the model weights. Once the training is done, the model can be used for the prediction of unseen test data. Feature extraction is a very important step in machine learning. The relevant set of information extracted from the input that directly determines the target values is called as features. In the previous example of house pricing, the number of rooms or area of the house can be considered as features. Similarly, for a classification problem, if you are given labeled images of cats and dogs as input, color of the animal in the image, its facial structure, the presence or absence of whiskers, and so on could be the relevant feature to create the classifier.

Training of a supervised learning algorithm has the following three basic components:

- A **decision process** that makes a guess of the target values from the input features.
- An **error function** that finds how good the guess is with respect to the actual target values or labels.
- An **optimization process** that iteratively adjusts the decision process via modifying the model weights to reduce the error between the guessed and the actual target values.

Linear regression, logistic regression, support vector machine, and Artificial Neural Networks (ANN) are popular examples of supervised learning algorithms.

Unsupervised Machine Learning

Unsupervised machine learning algorithms deal with unlabeled data. That means you only have the features but not the labels. Such algorithms try to find the hidden patterns of the input data based on the features and group them together to form clusters. All data in a particular cluster share similar properties. Unsupervised learning is typically used in applications such as customer segmentation, similarity detection, product recommendation, data dimensionality reduction, and so on, where you really do not know the target labels. A few examples of unsupervised learning algorithms are principal component analysis and K-means clustering.

Machine Learning and Deep Learning

Deep learning is the newest yet most popular branch of machine learning that works particularly well on unstructured data. Deep learning algorithms can be considered as mathematical evolution of traditional Machine Learning algorithms. Refer to the

basic block diagram of supervised learning in *figure 1.1*. A machine learning algorithm cannot learn from raw unstructured data. It first needs to extract a set of relevant features, and the features are then used to train the model. Feature extraction is a manual process in traditional machine learning. Finding the optimum feature set is probably one of the most difficult tasks, which might require domain expertise in the field application. Deep learning algorithms can directly take raw data as input and can extract the relevant features automatically, therefore, bypassing the need for manual feature extraction. Deep learning techniques are particularly useful to process unstructured data (for example, text and images).

Deep learning approaches analyze data in a way similar to the human brain. They have a layered structure of **Artificial Neural Network** (**ANN**), which is inspired by the biological nervous system. An ANN comprises of neurons or nodes in a layered structure where each layer is connected to another layer to analyze complex patterns and relationships in data. A typical neural network requires huge training data but minimum human intervention to function. The ability of deep learning algorithms to work with minimum human intervention makes them extremely popular in modern data science in diverse practical applications such as computer vision, speech recognition, natural language processing, and so on. In this book, we will heavily use deep neural networks in various projects, primarily the **Convolutional Neural Network** (**CNN**).

Edge computing and TinyML

Machine learning, more specifically deep learning algorithms, are computationally expensive. In reality, it may take from several hours to several days to train a large neural network model using sophisticated hardware acceleration platforms such as **Graphics Processing Unit** (**GPU**) or **Tensor Processing Unit** (**TPU**). Such hardware platforms are maintained by large enterprises at large distributed data centers. As individual users, we generate data at our end in our personal edge devices, like smartphones or tablets, in the form of text, audio, video, or image. However, the devices we possess are not always capable of running complex machine learning models for an application. Machine Learning operations are traditionally performed on the cloud. Users' data is typically sent to the backend data center that hosts the machine learning model via the internet for processing, and the result is transferred back to our device. For effective data management and processing, all our devices are connected to the internet to create an ecosystem called the **Internet of Things** (**IoT**). The interconnection between traditional machine learning and IoT is no doubt effective as we get all our jobs done seamlessly. However, it has its own drawbacks. A few key challenges are listed as follows:

- **Data privacy and security**: In traditional machine learning, IoT devices send their data to a cloud network for processing. This is prone to cyber-attacks, and hence, has severe security and privacy issues.
- **Power consumption**: Machine learning models consume enormous power. A research team at the University of Massachusetts estimated in 2019 that deep learning sessions of a machine learning model could generate up to 626,155 pounds of CO_2 emission, which is roughly equal to the carbon emission of five cars over their lifetime.
- **Network bandwidth and latency**: It would require an infinite bandwidth to support hundreds and thousands of IoT devices, to continuously stream their data to the cloud for processing. Another key aspect is of network latency. Latency is termed as the time lag in sending and receiving the data between an IoT device and the server over the network. In slower networks, the latency is higher, and the user often needs to wait for a long period of time to get a response from the server. It is undesirable for user engagement in real-time applications.

Edge AI and Edge ML have emerged as the next frontier of development for IoT systems. In Edge AI, data is produced, handled, and processed locally. Instead of sending to the cloud, the analytics happens in the edge device, such as smartphones, single board computers, IoT devices, or edge servers. Real-time processing allows a faster response and reduced latency and bandwidth use. Applications of Edge AI can be seen in object detection, speech recognition, fingerprint detection, autonomous driving, and so on.

Tiny machine learning, commonly known as TinyML, takes Edge AI one step further in order to run machine learning algorithms even on the smallest microcontrollers with the least amount of power possible. TinyML is a rapidly growing field in machine learning. Instead of GPUs or microprocessors for computation, TinyML entirely relies on less capable processing units that consume very less power, typically in the range of a few milliwatts. Such processors are frequently Cortex-M based, having only a few hundred kilobytes of **Random Access Memory** (**RAM**), a few megabytes of flash memory, and clock rates in the tens of megahertz. Therefore, TinyML applications ensure low power consumption, low latency in running a machine learning model. They also ensure to preserve user privacy as the data is entirely being processed on the edge device.

Applications of TinyML

TinyML applications are extremely energy efficient. A standard **Central Processing Unit** (**CPU**) consumes around 70–85 Watts, and a GPU consumes up to 500 Watts of power to operate. On the other hand, TinyML models operate on microcontrollers that consume only a few milliwatts or microwatts. Such devices are intended to run for several weeks or even months without recharging or changing of the batteries. This brings down the overall carbon footprint. Processing at the edge also ensures low latency and improved data privacy. These devices are relatively basic in terms of computation hardware, making them available at a cheaper price. TinyML is successfully applied in various practical applications across industries, explained as follows:

- **Predictive maintenance**: Large industrial machines are prone to making faults. Predicting a fault of a machine ahead of time is important in any industry to avoid a potential shutdown. Under normal health conditions, most machines exhibit some standard properties in terms of mechanical noise, vibration, torque, and so on. A deviation from the normal range can be an alarm for the potential fault of the machine in the near future. Continuous monitoring of the machine is possible by gathering relevant information on various properties by installing sensors on the body of the machine for analysis. Such analysis is typically done 24 × 7 using small microcontrollers with minimum power consumption, as the frequent replacement of the device battery is impractical.
- **Healthcare**: TinyML is bringing in affordable solutions in early disease screening and medical diagnostics, which can be used in developing nations to supplement limited healthcare facilities. Off-the-shelf electronic devices in the form of wristbands or smart watches are readily available that use TinyML algorithms to measure physiological parameters like heart rate and blood pressure. Such devices can also predict abnormal heart rhythms like atrial fibrillation, which can be an early sign of a heart attack.
- **Agriculture**: There are mobile phone applications for assisting farmers to detect diseases in plants just by taking a few pictures of the diseased plants to run on-device machine learning algorithms for analysis. As the applications do not need images to send to the cloud, they can help the farmers in remote areas where stable internet connectivity remains an issue.
- **Voice-assisted devices**: Voice-assisted devices such as Amazon Echo, Google Home, or iPhone's Siri have become very popular these days. Such devices listen to your voice command and can act accordingly, such as playing your favorite music, turning ON/OFF the room light, and so on. This is a

perfect example of where TinyML and traditional machine learning work together. The microphone of the voice-assisted device continuously analyses the background sound to detect a wake-up keyword such as "OK Google!", "Alexa!", or "Hey Siri!". The keyword detection process has to be extremely light-weight, on-device, and low-powered. This is where TinyML is deployed. As soon as the keyword is detected, the device wakes up and records your following voice instruction like *"What's the weather going to be like today?"* or *"Play my favorite music."*, which is sent to the cloud for processing via more powerful natural language processing algorithms which are not possible to run at the edge.

- **Ocean life conservation**: TinyML applications are used for real-time monitoring of whales in North America to avoid whale strikes in busy shipping lanes.

Hardware for deploying TinyML

A complex deep learning model can have several thousand to millions of trainable parameters, resulting in a large model size in the range of several megabytes to gigabytes. In general, model size is not a big issue when machine learning applications are deployed on a remote server that virtually has infinite memory space for storage. However, the scenario is different in TinyML, having a few hundred kilobytes of RAM.

Deciding the correct hardware for deploying TinyML models is often challenging. You have to keep several factors in mind, for example, the device form factor for your application, how much memory storage your model requires, the maximum allowable power consumption by your application, what sensors you might require for the collection of data and their interfaces, whether you require on-device training, the approximate price of your application, and so on. Our smartphones and tablets are great examples of edge devices. The past few years have witnessed a rapid proliferation of smartphones. Modern smartphones are rich in computing resources and in-built sensors. You can even train medium size neural networks on them. Smartphones and tablets are a great choice to run Edge ML applications that involve a strong user interface, for example, on-device face recognition for person identification, high-definition videography, gaming, natural language processing, and so on.

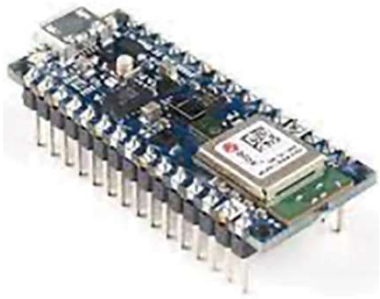
A **Single Board Computer** (**SBC**) is another popular device for edge computing in IoT-based applications. An SBC is a small portable computing device built on a single printed circuit board with a microprocessor, memory, and **input/output** (**I/O**) devices. Although an SBC has much smaller memory and lesser powerful processor

than a personal computer, it comes at a much cheaper price and drives significantly low power to operate. An SBC can draw its required power to operate from a power bank or the USB port of a computer. SBCs can easily interface with external sensors like servo motors and ultrasound sensors. They are typically used in academic projects and industrial applications where edge devices of small form factors are required to be directly connected to external devices for data collection and analysis. *Figure 1.2* shows a picture of a Raspberry Pi device, a popular SBC used in various commercial applications and academic projects. The device is powerful enough to run optimized deep learning models.

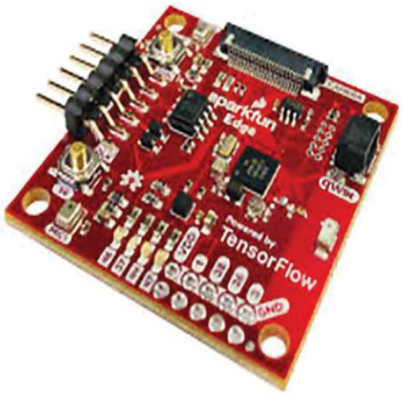
***Figure 1.2**: Raspberry Pi 3, Model B+, a popular single board computer*

When we think of deploying extremely low-profile TinyML applications to operate 24 × 7, the primary target hardware are the microcontrollers. A microcontroller is a compact **Integrated Circuit** (**IC**) designed to perform a specific task in an embedded platform. They are much smaller in size than a smartphone or an SBC and have much lesser computing resources. However, they are extremely low-powered. A microcontroller primarily contains a CPU that connects all other components in a single system. The CPU fetches data, decodes, and executes the assigned task. The CPU clock speed typically ranges between 16 megahertz and 64 megahertz in a microcontroller. They have a small amount of computation memory along with a certain amount of **Read Only Memory** (**ROM**) or flash memory for the storage of data and programs. The typical RAM size is 64 to 256 kilobytes, and the flash memory can be of 2 megabytes. There are several I/O ports to communicate with external devices. Microcontrollers may also contain one or more in-built timers and counters, **Analog to Digital Converter** (**ADC**) and **Digital to Analog Converter** (**DAC**), to read data from external sensors. Microcontrollers can be divided into various categories depending on the underlying architecture, memory, and instruction sets. They are used in applications such as machine health monitoring, space research, autonomous cars, and so on, which need to operate for a prolonged duration without frequently replacing the battery.

Microcontrollers are significantly different than a computer system. A computer system is designed to perform multiple different tasks concurrently, whereas a microcontroller is specifically designed for one particular application, such as turning ON/OFF an LED, rotating a servo motor, or controlling a robotic arm. Microcontrollers have a much-constrained hardware environment. The CPU clock speed of a powerful microcontroller can be up to 64 megahertz, with 256 kilobytes of RAM and only 1–2 megabytes of flash memory. On the other hand, modern computer systems come with several gigahertz of CPU clock speed, 8–16 gigabytes of RAM and terabytes of storage area. Microcontrollers do not have an operating system. They draw much smaller power compared to a computer, which is in the range of milliwatts or microwatts. Hence, they can operate for several weeks without recharging or replacing the battery, making them extremely popular for continuous operation in edge computing applications. *Figure 1.3* shows few of commercially available microcontroller units popularly used in TinyML applications:

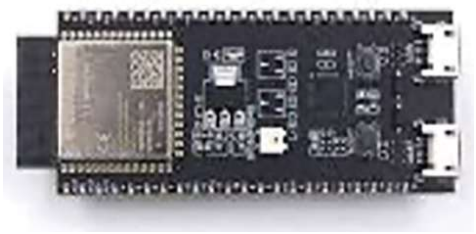


Arduino Nano 33 BLE Sense



SparkFun Edge

Raspberry Pi Pico



ESP-32-S3

***Figure 1.3**: Popular microcontrollers for TinyML applications*