Applied TinyML

End-to-end machine learning for microcontrollers with examples

Ricardo Cid

www.bpbonline.com

First Edition 2025

ISBN: 978-93-65890-716

LIMITS OF LIABILITY AND DISCLAIMER OF WARRANTY

The information contained in this book is true and correct to the best of author's and publisher's knowledge. The author has made every effort to ensure the accuracy of these publications, but the publisher cannot be held responsible for any loss or damage arising from any information in this book.

All trademarks referred to in the book are acknowledged as properties of their respective owners but BPB Publications cannot guarantee the accuracy of this information.

Dedicated to

My dear daughters:
Dianna
and
Luna

About the Author

Ricardo Cid is a mechanical engineer specialized in mechatronics who has been working with embedded systems for more than 20 years. At school, he was already helping a team at the engineering institute (UNAM) connect a washing machine to the internet even before the term IoT was coined (to this day, he has not figured out why connecting a washing machine to the internet is necessary). After finishing school, he moved to NYC to design stage robots for artists in his free time while working as the head of engineering for a very successful company in the travel industry, where he pioneered creating systems in the early days of the cloud, acquiring extensive experience in big data and massive traffic volumes.

Around 2015, he took a six-month residency at the Museum of Arts and Design at Columbus Circle with a project that earned him multiple awards, consisting of creating 3D-printed mechanisms that danced to music. Because of his unique skill set, which included mechanical engineering, electronics, enterprise software, APIs, cloud, and big data, one of the biggest and most prestigious real estate portfolios in NYC offered him the unique opportunity to build a smart building operating system from scratch, using more than 15 skyscrapers as a sandbox.

During that amazing gig, Ricardo experimented with massive amounts of data and conceived a series of applications for a then-new type of technology called machine learning. Ricardo and his team conceived dozens of prototypes, some of them never saw the light, but they laid the foundation for a series of algorithms that eventually saved hundreds of megawatts in multiple buildings across the U.S., including those of the federal government and the largest bank in the world.

At the end of his sixth anniversary at that company, Ricardo realized there was a massive opportunity to bring much of that intelligence to edge devices, avoiding critical cybersecurity and reliability single points of failure. In 2023, Ricardo created a design studio exclusively dedicated to architecting and building edge solutions that run ML models in constrained environments.

About the Reviewer

Martin Yanev is a highly accomplished software engineer with nearly a decade of experience across diverse industries, including aerospace and medical technology. Over his illustrious career, Martin has carved a niche for himself in developing and integrating cutting-edge software solutions for critical domains such as air traffic control and chromatography systems. Renowned as an esteemed instructor and computer science professor at Fitchburg State University, he possesses a deep understanding of the full spectrum of OpenAI APIs and exhibits mastery in constructing, training, and fine-tuning AI systems. As a widely recognized author, Martin has shared his expertise to help others navigate the complexities of AI development. With his exceptional track record and multifaceted skill set, Martin continues to propel innovation and drive transformative advancements in the field of software engineering.

Acknowledgement

I would like to thank, first of all, you, the reader, for picking up this book. I hope that by the end of it, you are inspired to create awesome systems and gadgets that inspire more people to join you in this adventure.

I would also like to express my gratitude to my entire family for supporting me during the creation of this book. They actively participated in the data acquisition of many of the use cases by driving me around while I was holding an accelerometer, placing sensors in the refrigerator for me, playing notes on the piano to test the classification model, taking pictures of their hand signals, holding a sensor while they jumped to simulate free fall, and many more. We did this together.

Nothing could have happened without the BPB team, to whom I am eternally grateful. From believing in the book concept, being patient during the revisions, accepting my changes, guiding me through the process, and helping me get to the finish line; all with the best attitude and kindness there could be.

And finally, I need to recognize the entire TinyML ecosystem and its awesome community. All the tools, all the SDKs, books, blog posts, etc are pure gold, thank you. Special thanks to Nordic, Bluetooth, Arduino and Edge Impulse.

Preface

The idea for this book was born from my fascination with the intersection of machine learning and embedded systems. In the last few years, I have seen how AI has transformed industries. Many of the most powerful applications remain locked within cloud servers and large computing infrastructures. My goal with this book is to help bridge that gap by showing readers that it is possible to bring intelligence to edge devices and make TinyML accessible to engineers, makers, and innovators who want to deploy machine learning in real world, low power environments.

TinyML is the latest frontier for AI and ML models, as its constraints are everywhere. Memory is scarce, and computing is smaller by orders of magnitude compared to traditional computing used to train and run the models. Therein, however, lies one of the greatest opportunity windows in many years. When you think about the future, you do not imagine just text editors that help you write a letter but an interactive physical world, appliances and stand-alone devices transacting with people in the most natural way, friendly furniture, smart clothing, and personalized gadgets. All of that is the great promise of TinyML. It is our once-in-a-lifetime opportunity to not only live in the future but to create it.

Another important aspect about this book is its breadth. The principles of each concept are explained lightly; however the variety of domains covered is wide. Instead of focusing solely on the data science aspect of the TinyML models, we include system design (requirements are mapped to an hypothetical business problem that needs to be solved), data acquisition challenges (rather than just assuming the existence of a ready made data set available on the internet), feature extraction (which is directly impacted by data quality, a real issue in production environments), model design (understanding how the model works under the hood is critical in constrained environments where resource waste is not an option), electronic component configuration (using existing electronic components instead of just running in simulators), networking (since real world solutions are often distributed across a surface rather than confined to a single location), power considerations (as connection to the electric grid cannot be assumed), and bill of materials research (because cost is a fundamental metric in real-life systems).

Chapter 1: Foundation and Methodology - This chapter walks the reader through the concepts and terminology used throughout the book. It also established the scope of each one of the use cases in this book.

Chapter 2: Sound Classification - This chapter focuses on identifying the features that define sounds according to their specific use cases. We utilize Mel Frequency Cepstral Coefficients (MFCC) for human voices, which excel in recognizing linguistic attributes such as inflection rhythm and silences. Fourier Coefficients are employed for transient sounds to delineate the sound by its power distribution across frequencies, which is essential for capturing the unique energy profile of each sound. Periodic tones are analyzed through spectrum analysis to uncover their distinctive frequency signatures. We employ Convolutional Neural Networks to leverage the spatial representation of sound (across time and magnitude) for pattern recognition, training the model to convert the output layer into a probability distribution used to classify the sound sample.

Chapter 3: Movement Classification - In exploring movement classification, we gather data from three orthogonally placed accelerometers to accurately describe object movement. By breaking down the accelerometer readings into thirteen distinct features and segmenting the wave into small time windows, we calculate metrics such as root mean square, kurtosis, spectral kurtosis, and skewness. These metrics reveal critical aspects of movement, from the average power indicating the movement's energy to kurtosis and skewness, offering insights into the frequency and suddenness of movements. The rich dataset feeds into a dense neural network, enabling precise classification of movements ranging from gentle to vigorous.

Chapter 4: Image Classification - Focusing on image classification, we acquire images through a direct connection to a digital camera interfaced with a microcontroller. Preprocessing steps include adjusting the image to a square format and, if color is not critical, converting it to 8-bit grayscale to streamline object recognition models' application. Images are resized to a standard size to accommodate the neural network's capacity. Rather than training from scratch, we apply industry standard TinyML models and a technique called transfer learning to efficiently train the model with our dataset, achieving a fast and accurate system applicable across various machine vision scenarios.

Chapter 5: Object Tracking - This chapter introduces tracking, a technique often paired with Image Classification to consistently identify an object across sequential video frames. Whether tracking moving or static objects, the method involves first classifying objects with a Convolutional Neural Network and then comparing their sequential positions to deduce if they are the same based on proximity. Successful tracking assigns consistent

IDs to objects, maintaining identity even through temporary classification lapses or visual obstructions, showcasing the robustness of a good tracking algorithm.

Chapter 6: Sensor Fusion - Sensor Fusion is examined as an advanced method of combining data from multiple sources to generate new insights. We categorize sensor fusion into complementary, competitive, and cooperative types. Complementary fusion brings together different data perspectives of the same event, while competitive fusion seeks reliability through redundancy in data acquisition. Cooperative fusion merges data from unrelated events to synthesize new information, highlighting the technique's versatility in enhancing system perception.

Chapter 7: Deep Learning Regression - This chapter explores using deep neural networks to infer numerical values from one or more data sources. We demonstrate how integrating feature engineering with deep learning can expand the realm of regression applications. We discuss developing control systems using regression models with multiple inputs and outputs that apply continuous learning and adaptation to current conditions. This chapter illustrates how regression can forecast future conditions based on historical data, opening new possibilities for predictive analytics.

Chapter 8: Anomaly Detection - This chapter examines the anomaly detection method, which aims to identify occurrences that deviate from expected patterns. Distinguishing itself from classification, anomaly detection focuses on detecting behaviors or events previously unseen by the model.

Code Bundle and Coloured Images

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

https://rebrand.ly/m6t6yc9

The code bundle for the book is also hosted on GitHub at
https://github.com/bpbpublications/Applied-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
Foundation and Methodology

Introduction

If asked what can analyze its surroundings, understand ongoing events, and respond without external help, your likely response would be a living organism, particularly an animal. This is correct, but you could also accurately say an **Applied TinyML** system. A TinyML application gathers data, analyzes it, extracts insights, and reacts to the environment at or near the data source. This method offers significant advantages: it ensures privacy by keeping data local, enables real-time decision-making without cloud delays, eliminates single points of failure by distributing decision-making, reduces communication costs, and creates a scalable system with thousands of devices working towards a goal. Additionally, distributing computation lowers power consumption, making battery or solar power feasible for off-the-grid use.

In this chapter, we begin with an overview of data acquisition, discussing ideal data types and frequencies, peripherals, labeling techniques, and noise reduction. Next, we guide you through processing, feature selection, and extraction. We then examine **machine learning** (**ML**) model types, suitable architectures, and hyperparameter tuning for enhanced accuracy. After obtaining the model's refined result, we show how to write a program to use or communicate this result, including issuing commands to actuators and displays.

In distributed applications, data acquisition, processing, and actuation rarely occur on the same device, requiring a network for device communication. You will learn to use a pub/sub paradigm for scalable, decentralized communication. Finally, understanding power

consumption and selecting appropriate power sources is essential, as is preparing a bill of materials to ensure your project's financial and maintenance viability.

Structure

The chapter covers the following topics:

- TinyML in AI
- Designing and building a TinyML application

Objectives

The goal of this chapter is to introduce the terminology and concepts covered throughout the book. We begin with an overview of TinyML and its role within **artificial intelligence** (**AI**), then explore what constitutes a TinyML application. We outline the steps to create one, from defining the problem and developing the concept, to deciding on data acquisition, selecting features, creating the model, and crucially, deploying it while considering hardware, network, and power usage.

TinyML in AI

TinyML applications operate on devices with limited computational capacity and low power requirements. However, these constraints are technological, not goal oriented. If we could integrate the power of a **graphic processing unit** (**GPU**) into a microcontroller lasting a decade on one battery, the objective would remain a self-sufficient entity operating at the Edge. Thus, the industry often refers to these applications as EdgeML.

In a standalone application, each design decision affects every subsystem. The data's size and frequency determine the sensor type, exchange protocol, transmission network, processing microcontroller, storage memory, display interface, and power source.

TinyML is a subset of AI specifically tailored to apply AI's capabilities to the smallest and most energy-efficient hardware. While AI encompasses many technologies and applications, TinyML focuses on making a subset of these technologies work in the most resource-constrained scenarios.

The core concept behind minimizing AI model sizes lies in the realization that you can train models in powerful machines but deploy them in significantly lower-power microcontrollers. Typically, the training of TinyML models occurs on powerful cloud-based GPUs, utilizing vast datasets amounting to hundreds of gigabytes. Through employing various optimization strategies that reduce the model's dimensionality and accept minor, often imperceptible, losses in accuracy, the model's size can be reduced to just a few megabytes. This size reduction makes it feasible to deploy the models onto microcontrollers. However, this approach applies only to specific models with certain

types of inputs and outputs. Despite these limitations, the research community remains exceptionally vibrant, frequently publishing new white papers introducing advanced AI models into the TinyML sphere. The progression began with sound classification for microphone activation, extended to image classification, followed by anomaly detection, and eventually to compact language models.

Level of experience and knowledge required to use TinyML

The landscape of TinyML application development is transforming, making it more accessible and intuitive than ever before. Historically, modeling and implementing physical behaviors into applications required a deep understanding of mathematics and seasoned software engineering skills. However, ML has created a new, more natural methodology for development that is easier to understand and mirrors the human learning process. Just as individuals acquire language skills through trial and error rather than memorization of grammatical rules, ML models learn from examples. By presenting a dataset and indicating the desired outcome, the Model iteratively adjusts through multiple attempts, improving its accuracy over time.

This evolution in the development paradigm is significantly lowering the barriers to entry in the field of TinyML. By equipping non-technical individuals with user-friendly tools for data collection, modular processing components, model training interfaces, and practical usage examples, ML technology is becoming accessible to a broader audience. This democratization of technology empowers anyone interested in learning and applying ML, paving the way for widespread adoption and innovation across diverse domains. The implications of this shift are profound, offering the promise of unleashing creative solutions and applications by people who were previously excluded from the technological development process due to the complexity of traditional methods.

Designing and building a TinyML application

A TinyML application is a solution that tackles a practical problem that exists in a business, organization, or location. This book teaches readers how to define, design, plan, budget, and build TinyML applications.

A TinyML application has elements of data science, software, ML, electronics, networking, power management, and **user interfaces** (**UI**), among many others. It is, nevertheless, possible to simplify its components as abstract building blocks.

The best way to stay on track is to follow a methodology that allows us to see the big picture, eliminate unnecessary complexity, and tackle one aspect of the system at a time while staying focused on creating something valuable that delivers value.

Throughout this book, we use a 10-step method to define the problem, develop concepts that could become a solution, select the suitable model, select the right features, use the right tools to train the model, test it, and integrate all this into an embedded solution that can be deployed to a real environment.

The method's steps are as follows:

1. Define the problem
2. Concept development
3. Data acquisition
4. Feature engineering
5. Model creation
6. Integrating the ML model
7. Hardware, electronics and connectivity
8. Networking
9. Power management
10. Materials and costs

These steps have been explained in detail in the further sections.

Step 1: Defining the problem

This step involves delineating the current situation, essential requirements, and metrics for success while remaining open to various potential solutions. You should be able to answer the following ten questions before starting any project. The document with the answers will guide the design decision-making in the following steps and validate the final implementation:

First, we define the problem:

- What is the current state?
- What is the desired state?
- What obstacles are preventing the transition from the current to the desired state?
- In what manner will this solution be utilized?

Next, we identify the data available for addressing the problem and establish a metric for tracking your progress:

- What data is accessible to tackle this issue?
- What output would you like the system to produce to demonstrate its effectiveness?
- Does the solution require any changes to the environment?

Finally, we specify the constraints of the solution:

- Where must the solution be implemented?
- Is it necessary for the solution to operate on an independent power source?
- Must the solution offer real-time updates?
- What is the allocated time and money budget for this solution?

Recognizing that this document is dynamic and will evolve over time is crucial. As the project advances through its various phases, you may realize that your initial goals could be more ambitious and require refinement or that the data you intended to use is challenging to obtain, needing an alternative approach. Additionally, changes in the project's location or shifts in environmental conditions may occur. View these developments as part of an iterative process where adjustments are acceptable and expected. The key is to remain focused on addressing the problem at hand.

Step 2: Concept development

In this step, we sketch a preliminary diagram showcasing a potential solution leveraging TinyML technology. This phase is inherently iterative, often requiring multiple iterations to align with expected metrics, functionality, and operational costs.

The core idea behind concept development is to generate multiple options to reduce the risk of choosing a suboptimal solution, which is achieved by exploring the solution space. To do this, we draft at least three different ways to solve the same problem. You need at least three, not only one, because usually, the first concept is the recipient of all our biases and assumptions of a solution, which is not necessarily the optimal one but, in most cases, based on personal preferences. The second concept will allow you to explore new approaches to solve the same problem. The third concept will be a good mix between the first and second concepts.

Creating the concept

The best way to create a concept is to represent its functionality in blocks. Each block should do one and one thing only. Each block can be represented as a black box with an input and an output.

Methodology

You can translate an idea like this one, there should be a people counting sensor on each one of the three doors communicating wirelessly to an LED panel showing the total people count, into a diagram.

Draw a diagram with as many boxes (or blocks) as components, name each, and list its main characteristics (color, type, voltage, etc.). Indicate what data is moving between blocks by writing it using the arrows that link the blocks (temperature signal, inference results, image, etc.). Try to estimate the cost of each component. Write the number by the blocks.