tinyML® Talks

Enabling Ultra-low Power Machine Learning at the Edge

"Unpacking the music genre recognition project from the TinyML Cookbook, second edition!"

Gian Marco Iodice –Tech Lead in the Machine Learning Group, Arm

February 13, 2024

www.tinyML.org

TINY ML TALKS webcast

Thank you, **tinyML Strategic Partners**, for committing to take tinyML to the next Level, together


TINY ML TALKS webcast

Executive Strategic Partners

Qualcomm
AI research

Advancing AI research to make efficient AI ubiquitous

Power efficiency

Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization

Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning

Robust learning through minimal data, unsupervised learning, on-device learning

A platform to scale AI across the industry

Perception

Object detection, speech recognition, contextual fusion

Reasoning

Scene understanding, language understanding, behavior prediction

Action

Reinforcement learning for decision making

Edge cloud

Cloud

IoT/IIoT

Automotive

Mobile

Accelerate Your Edge Compute

SYNTIANT

Making Edge AI A Reality

www.syntiant.com

TINY ML TALKS webcast

Platinum Strategic Partners

TINY ML TALKS webcast

AITRIOS

DEPLOY VISION AI AT THE EDGE AT SCALE

SONY

TINY ML TALKS webcast

Gold Strategic Partners


Build the
Future of tinyML
on arm

TINY ML TALKS webcast

EDGE IMPULSE

The Leading Development Platform for Edge ML

edgeimpulse.com

Driving decarbonization and digitalization. Together.

Infineon serving all target markets as
Leader in Power Systems and IoT

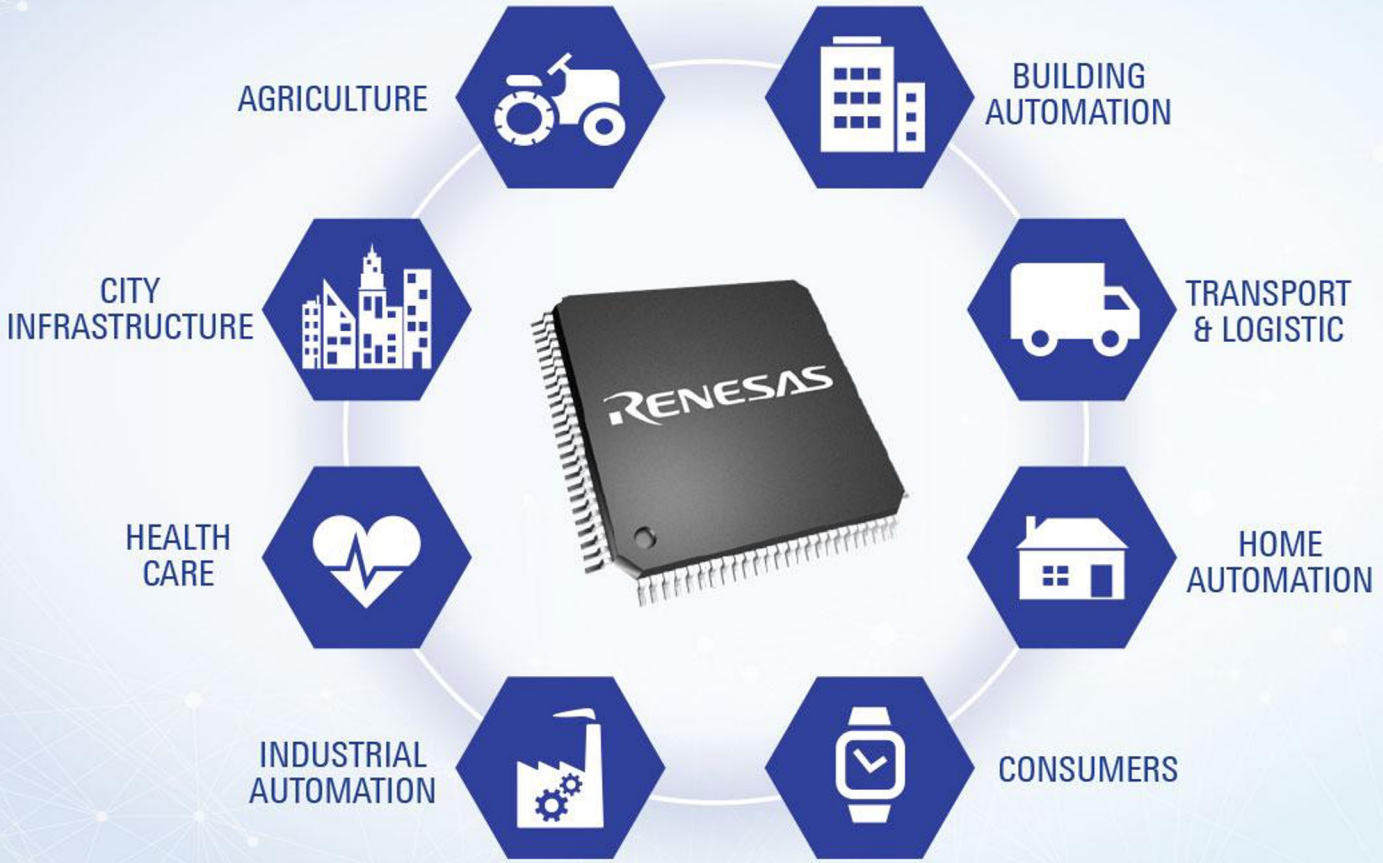
www.infineon.com

innatera

NEUROMORPHIC INTELLIGENCE FOR THE SENSOR-EDGE

www.innatera.com

Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.



renesas.com

STMicroelectronics provides extensive solutions to make tiny Machine Learning easy

life.augmented

www.st.com/ai

synaptics®

ENGINEERING EXCEPTIONAL EXPERIENCES

We engineer exceptional experiences for consumers in the home, at work, in the car, or on the go.

www.synaptics.com

TINY ML TALKS webcast

Silver Strategic Partners

Join Growing tinyML Communities:

19.3k members in
49 Groups in 41 Countries

tinyML - Enabling ultra-low Power ML at the Edge

https://www.meetup.com/tinyML-Enabling-ultra-low-Power-ML-at-the-Edge/

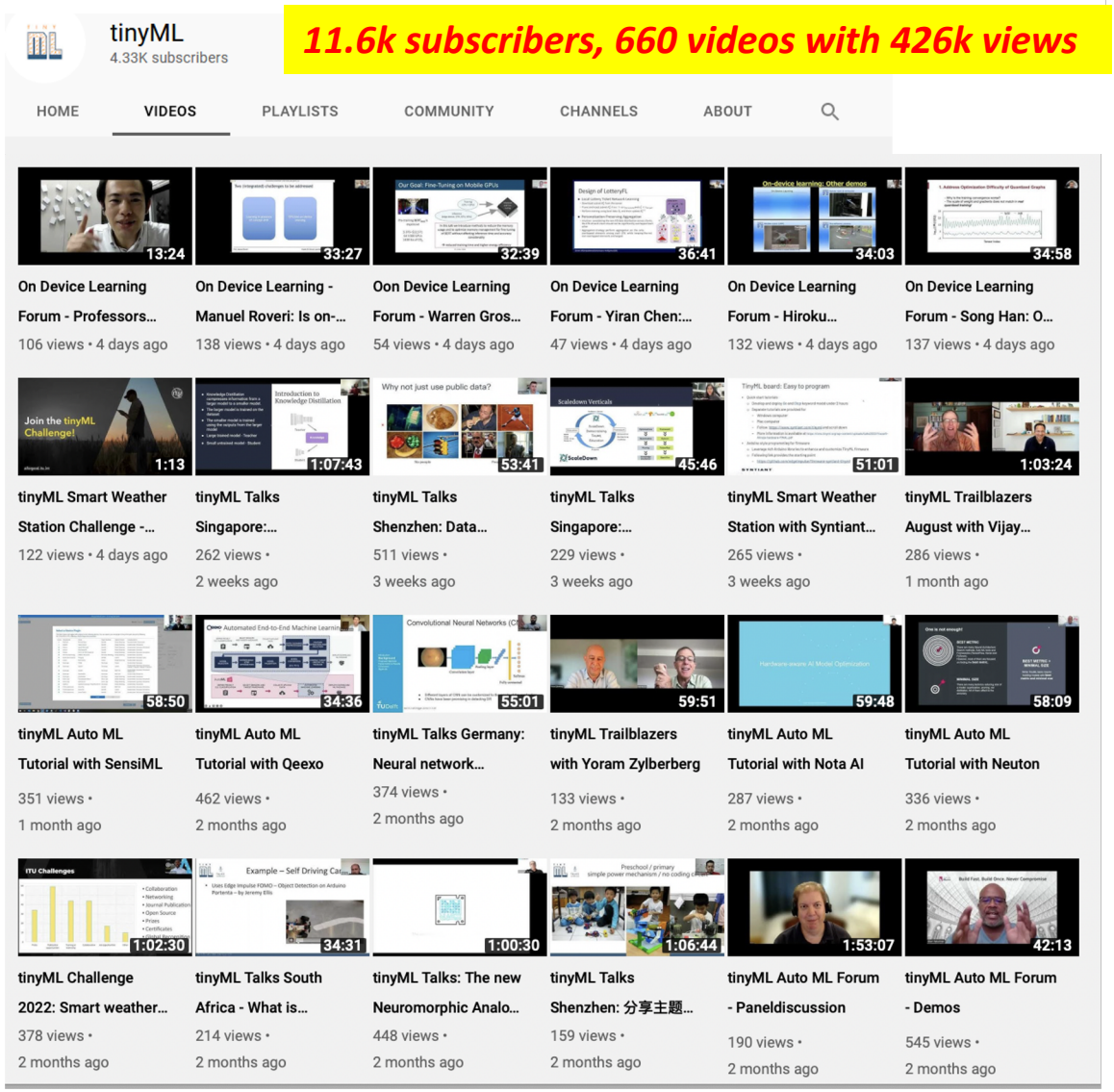
4.2k members
&
14.5k followers

The tinyML Community

https://www.linkedin.com/groups/13694488/

TINY ML TALKS webcast

YouTube



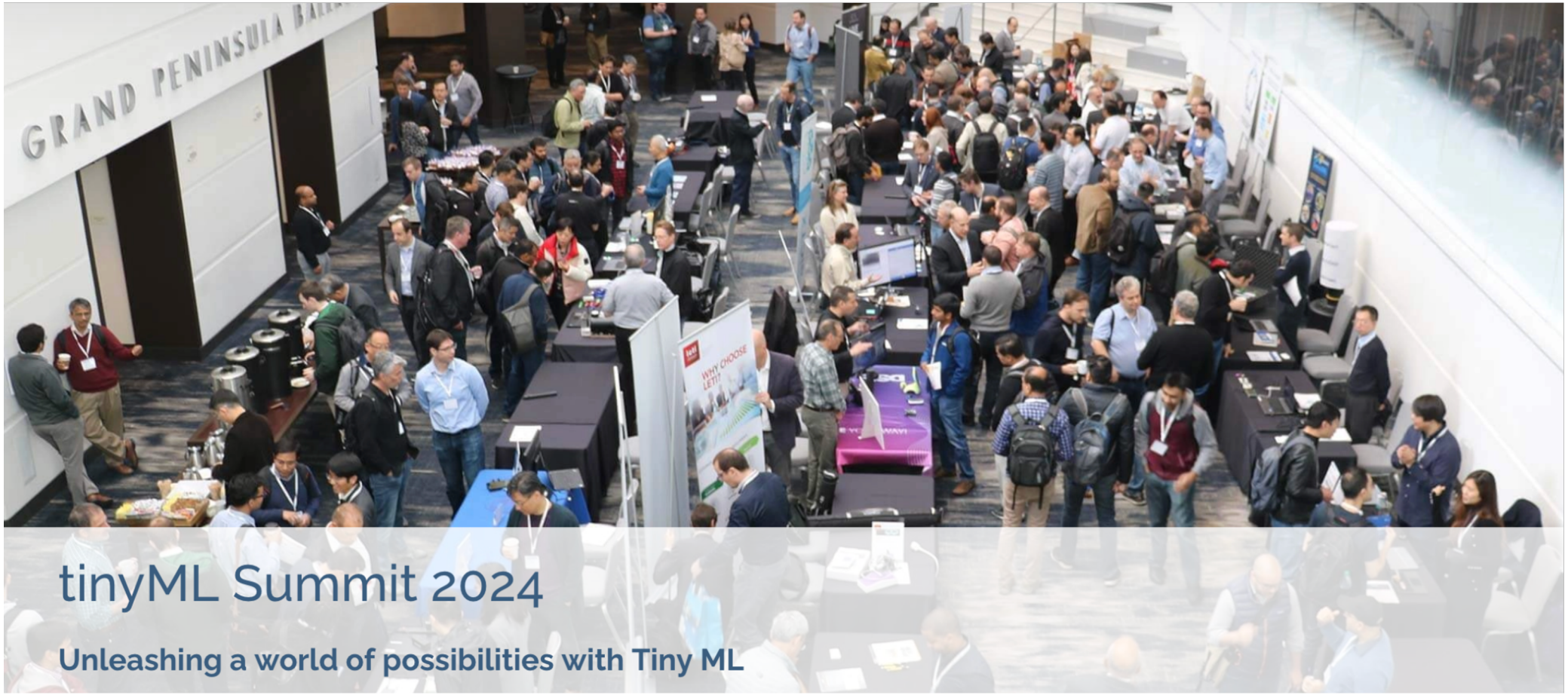
Subscribe to tinyML YouTube Channel for updates and notifications

(including this video)

www.youtube.com/tinyML

TINY ML TALKS webcast

tinyML Summit
April 22 - 24, 2024 - Register now!



tinyML Awards 2024

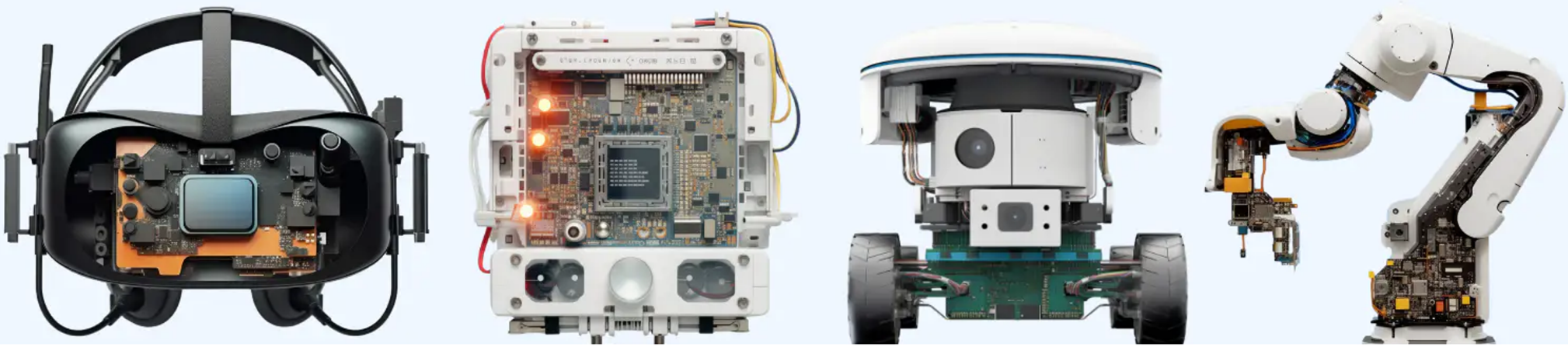
WEVOLVER

https://www.wevolver.com/article/2023-edge-ai-technology-report

TINY ML FOUNDATION

2023 Edge AI Technology Report

The guide to understanding the state of the art in hardware & software in Edge AI.



TINY ML TALKS webcast

Reminders

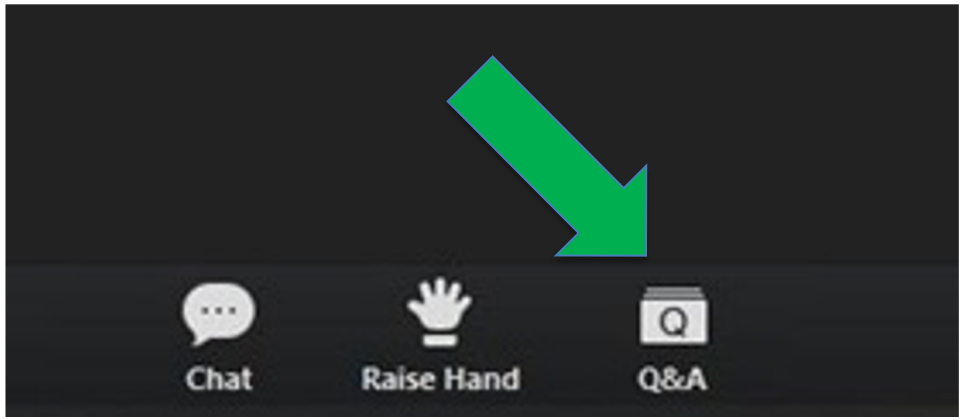
Slides & Videos will be posted tomorrow

PDF

tinyml.org/forums

youtube.com/tinyml

Please use the Q&A window for your questions



Gian Marco Iodice

Gian Marco Iodice is an experienced edge and mobile computing specialist at Arm for machine learning (ML). He is also chair of the global meetups for the tinyML foundation since 2022.

He received the MSc with honors in electronic engineering from the University of Pisa (Italy), where he specialized in HW/SW co-design for embedded systems.

Within Arm, he leads the engineering developments for Generative AI and the Arm Compute Library, which he co-created in 2017 to run ML workloads as efficiently as possible on Arm Cortex-A CPUs and Arm Mali GPUs.

In 2023, he collaborated with the University of Cambridge to integrate ML functionalities on an Arm Cortex-M microcontroller powered by algae.

Still, in 2023, Gian Marco contributed to developing the EdTech for Good Curation Framework. This framework, developed by UNICEF in collaboration with Arm and the Asian Development Bank (ADB), represents a significant step forward in the responsible use of technology in education.

EXPERT INSIGHT

TinyML Cookbook

Combine machine learning with small embedded devices to solve real-world problems

Second edition

packt

Gian Marco Iodice

5 and 6!

Recognizing Music Genres with TensorFlow and the Raspberry Pi Pico – Part 1

The project we will develop together holds a special place in my heart because it brings back memories of my first programming experience on an **Arm Cortex-M** microcontroller.

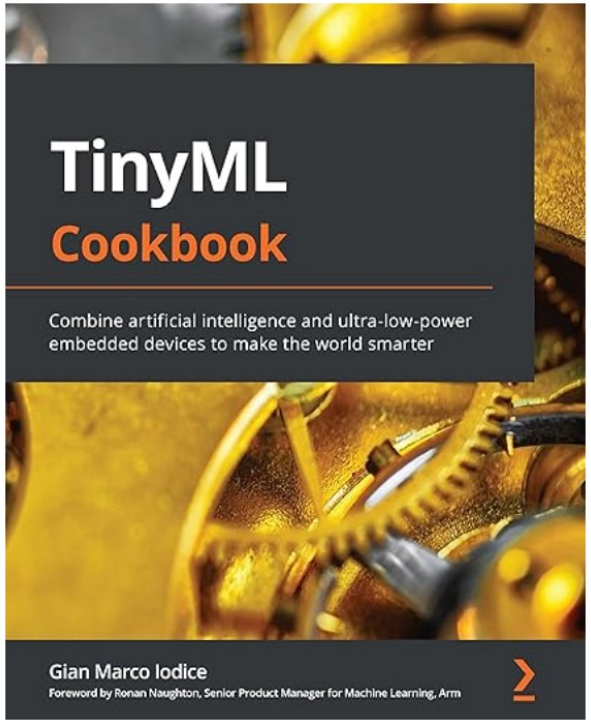
Back in my university days, portable MP3 audio players were definitely one of the coolest things to have. I was fascinated by this technology that allowed me to carry thousands of high-quality songs in my pocket to enjoy anywhere. As a technology and music enthusiast, I wanted to learn more about it. Therefore, I undertook the challenge of building an audio player from scratch using a microcontroller and a touch screen (`https://youtu.be/LXm6-LuMmUU`).

Completing that project was fun and gave me valuable hands-on experience. One feature I hoped to include was a **machine learning** (**ML**) algorithm for recognizing the music genre to auto-equalize the sound and get the best listening experience.

However, **deep learning** (**DL**) was not advanced enough then, especially on edge devices, so I had to abandon the idea. Today, with the recent advances in **tinyML**, I am excited to demonstrate that it is feasible to bring that algorithm to life on microcontrollers.

This project aims to recognize three music genres from recordings obtained with a microphone connected to the Raspberry Pi Pico. The music genres we will classify are disco, jazz, and metal. Since the project offers many learning opportunities, it is split into two parts to give as much exposure to the technical aspects as possible.

TinyML
Cookbook, 2Ed



TinyML Cookbook: Combine artificial intelligence and ultra-low-power embedded devices to make the world smarter 1st Edition, Kindle Edition

by Gian Marco Iodice (Author), Ronan Naughton (Foreword) | Format: Kindle Edition

4.8 ★★★★★ 30 ratings See all formats and ed

Book description | Editorial reviews

Work through over 50 recipes to develop smart applications on Arduino Nano 33 BLE Sense and Raspberry Pico using the power of machine learning

Key Features

- Train and deploy ML models on Arduino Nano 33 BLE Sense and Raspberry Pi Pico
- Work with different ML frameworks such as TensorFlow Lite for Microcontrollers and Edge Impulse
- Explore cutting-edge technologies such as microTVM and Arm Ethos-U55 microNPU

Book Description

This book explores TinyML, a fast-growing field at the unique intersection of machine learning and embedded

Thank you all!

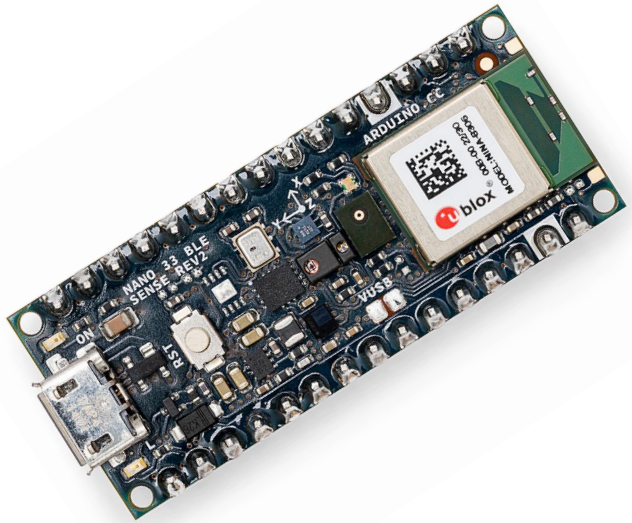
TinyML
Cookbook, 2Ed

The second edition: an incremental edition

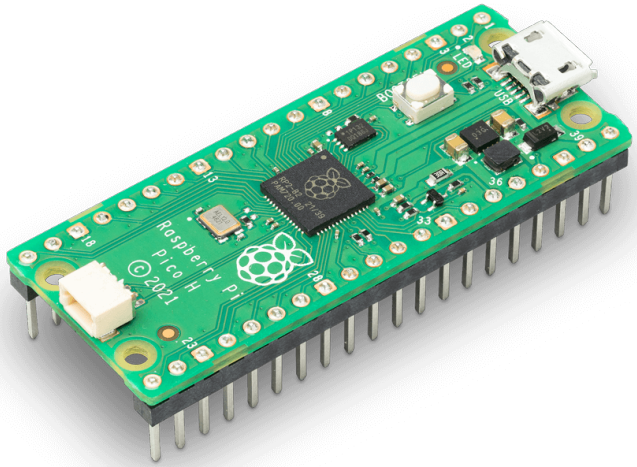
Introduction to tinyML	Intro to tinyML concepts (ML, microcontroller, power,..)
Embedded programming basics	A chapters for those new to embedded programming
Project with temperature and humidity	Weather station with TensorFlow
2 x projects with audio sensor	Keyword spotting with Edge Impulse and Music Genre Recognition with TensorFlow
2 x projects with vision sensor	Object detection with Edge Impulse and image classification with TensorFlow
1 x projects with inertial sensor	Gesture-based interface
Tips for building tiny models	Cifar-10 using less than 64 KB of RAM
1x project on a microNPU with TVM	Cifar-10 on the Arm Ethos-u55 uNPU
On-device learning & scikit-learn	Backpropagation on microcontrollers and scikit-learn deployment

12 chapters

On three different microcontrollers



Arduino Nano 33 BLE Sense



Raspberry Pi Pico

SparkFun Redboard Artemis Nano

TinyML
Cookbook, 2Ed

Where I can find the code...on GitHub!

https://github.com/PacktPublishing/TinyML-Cookbook_2E

Each chapter has its own folder.

Chapter02	Add README.md for Chapter02
Chapter03	Add README.md for Chapter02
Chapter04	Add README.md for Chapter04
Chapter05_06	Add instructions to use the local Arduino IDE
Chapter07	Add README.md for Chapter02
Chapter08	Add README.md for Chapter08
Chapter09	Add README.md for Chapter07
Chapter10	Add README.md for Chapter11
Chapter11	Update README.md
Chapter12	Add instructions to use the local Arduino IDE
Docs	Add guide to generate the Arduino tflite-micro libra
Imgs	Add guides
LICENSE	Initial commit
README.md	Update README.md

Each chapter contains a summary of the project

gmiodice

Name	Last commit mess
..	
ArduinoSketches	Remove static ke
Artifacts	Rename Chapter
ColabNotebooks	Add README.mc
PythonScripts	Rename Chapter
README.md	Add instructions

README.md

Chapter 5 and Chapter 6

Recognizing Music Genres with Te

About these chapters

This project aims to recognize three music genres from recordings
music genres we will classify are disco, jazz, and metal. Since the p
give as much exposure to the technical aspects as possible

Each chapter contains the shopping list and direct links to the source code and notebooks

Hardware comp

Sensor	board	Jumper wires	LEDs	Resistors	Extra
1x electret microphone amplifier - MAX9814	1x half-size	8x	-	-	5-pins press-fit hea (optional)
					1x push-button

Software (SW) Environments

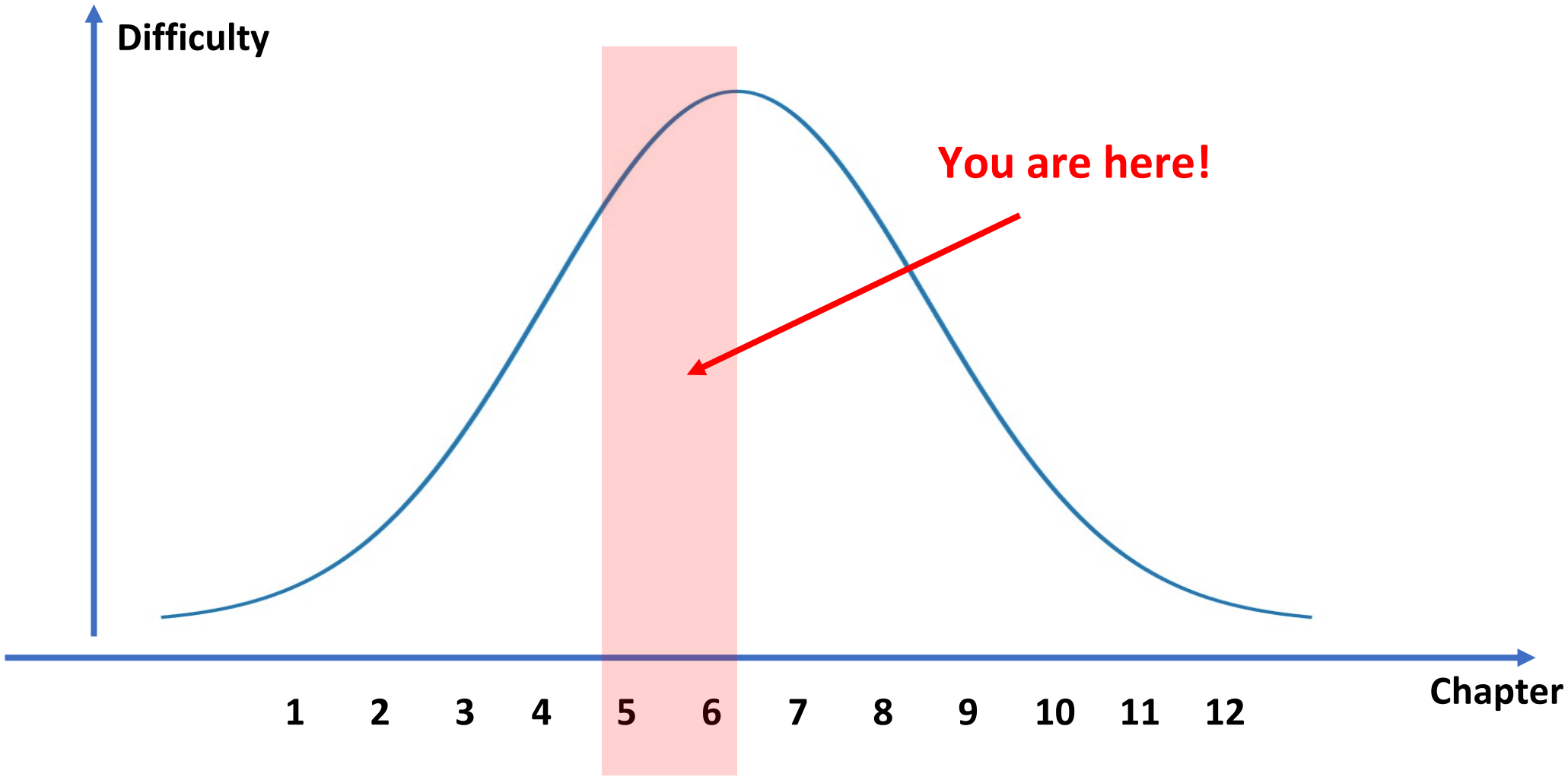
ML	Embedded programming
Google Colaboratory	Arduino IDE (or Arduino Web Editor)

Chapter 5 source code

Recipe	Source code	There's more
Connecting the microphone to the Raspberry Pi Pico		
Recording audio samples with the Raspberry Pi Pico	.ino	
Generating audio files from samples transmitted over the serial	.py	
Building the dataset for classifying music genres	.py	
Extracting MFCCs from audio samples with TensorFlow	.ipynb	.ipynb

TinyML
Cookbook, 2Ed

Learning experience point in the TinyML Cookbook



What you will learn in the book

Chapter 5

1. How to connect an external microphone to the microcontroller (Raspberry Pi Pico)
2. How to record audio samples with the microcontroller
3. How to upload audio samples to Google Drive
4. How to build a dataset for music genre recognition
5. How to extract the Mel Frequency Cepstral Coefficients (MFCCs) with TensorFlow

Chapter 6

1. How to use the CMSIS-DSP library in Python
2. How to implement the MFCCs algorithm with fixed-point arithmetic
3. How to design and train an LSTM RNN
4. How to evaluate the accuracy of the quantized model
5. How to implement the MFCCs algorithm in C using the CMSIS-DSP library
6. How to implement an app on the microcontroller to recognize three music genres (disco, jazz, and metal)

What you will learn in the book

Chapter 5

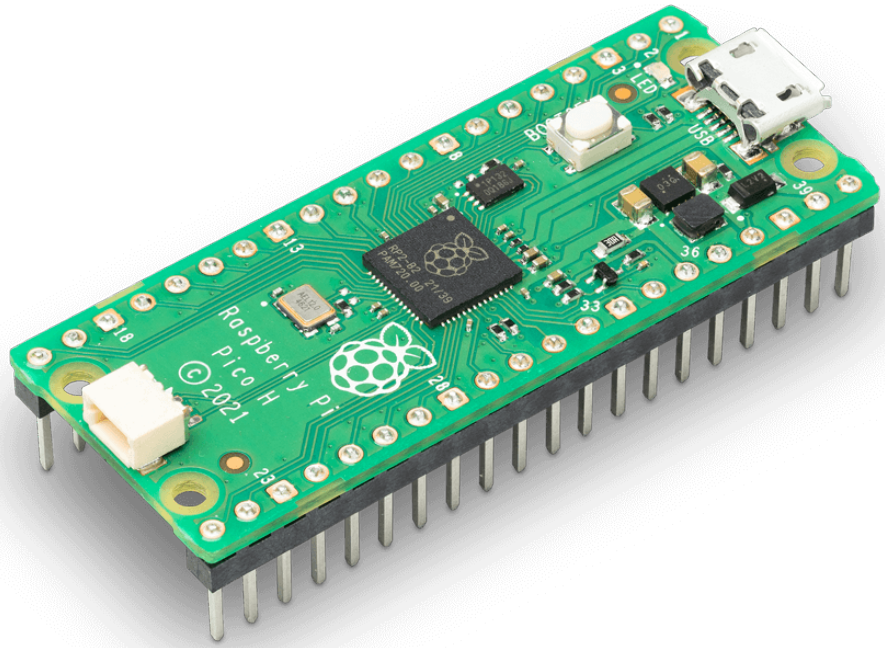
1. How to connect an external microphone to the microcontroller (Raspberry Pi Pico)
2. How to record audio samples with the microcontroller
3. How to upload audio samples to Google Drive
4. How to build a dataset for music genre recognition
5. How to extract the Mel Frequency Cepstral Coefficients (MFCCs) with TensorFlow

Chapter 6

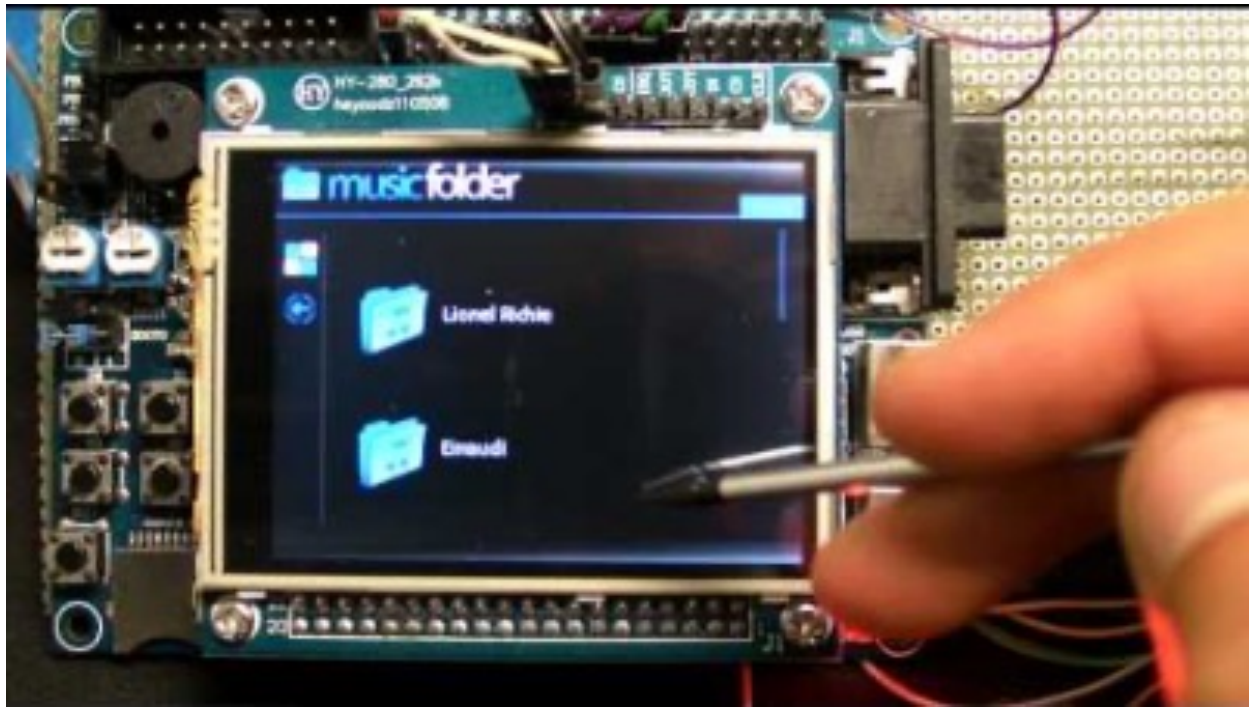
1. How to use the CMSIS-DSP library in Python
2. How to implement the MFCCs algorithm with fixed-point arithmetic
3. How to design and train an LSTM RNN
4. How to evaluate the accuracy of the quantized model
5. How to implement the MFCCs algorithm in C using the CMSIS-DSP library
6. How to implement an app on the microcontroller to recognize three music genres (disco, jazz, and metal)

What you will need

- A Raspberry Pi Pico
- A SparkFun RedBoard Artemis Nano (optional)
- A micro-USB data cable
- A USB-C data cable (optional)
- 1 x electret microphone amplifier
- 1 x half-size solderless breadboard
- 6 x jumper wires
- A laptop/PC with either Linux, macOS, or Windows
- A Google Drive account



Why this project?…everything started 12 years ago…



https://www.youtube.com/watch?v=LXm6-LuMmUU

Personal project to build an MP3 player **from scratch in C** on an Arm Cortex-M3 microcontroller (STM32F1x)

- Program memory: 512 KBytes
- SRAM: 64 Kbytes
- Songs stored on SD card
- Responsive touchscreen user interface (UI)
- Everything runs on the Arm Cortex-M CPU without O.S.
- No auto-equalization based on the music genre

At that time, there weren't SW libraries for the UI!

☹

The User Interface

Algorithms for sophisticated UI with screen overlay and anti-aliasing effects are generally very computationally expensive.

Therefore, how can the interface be so responsive?

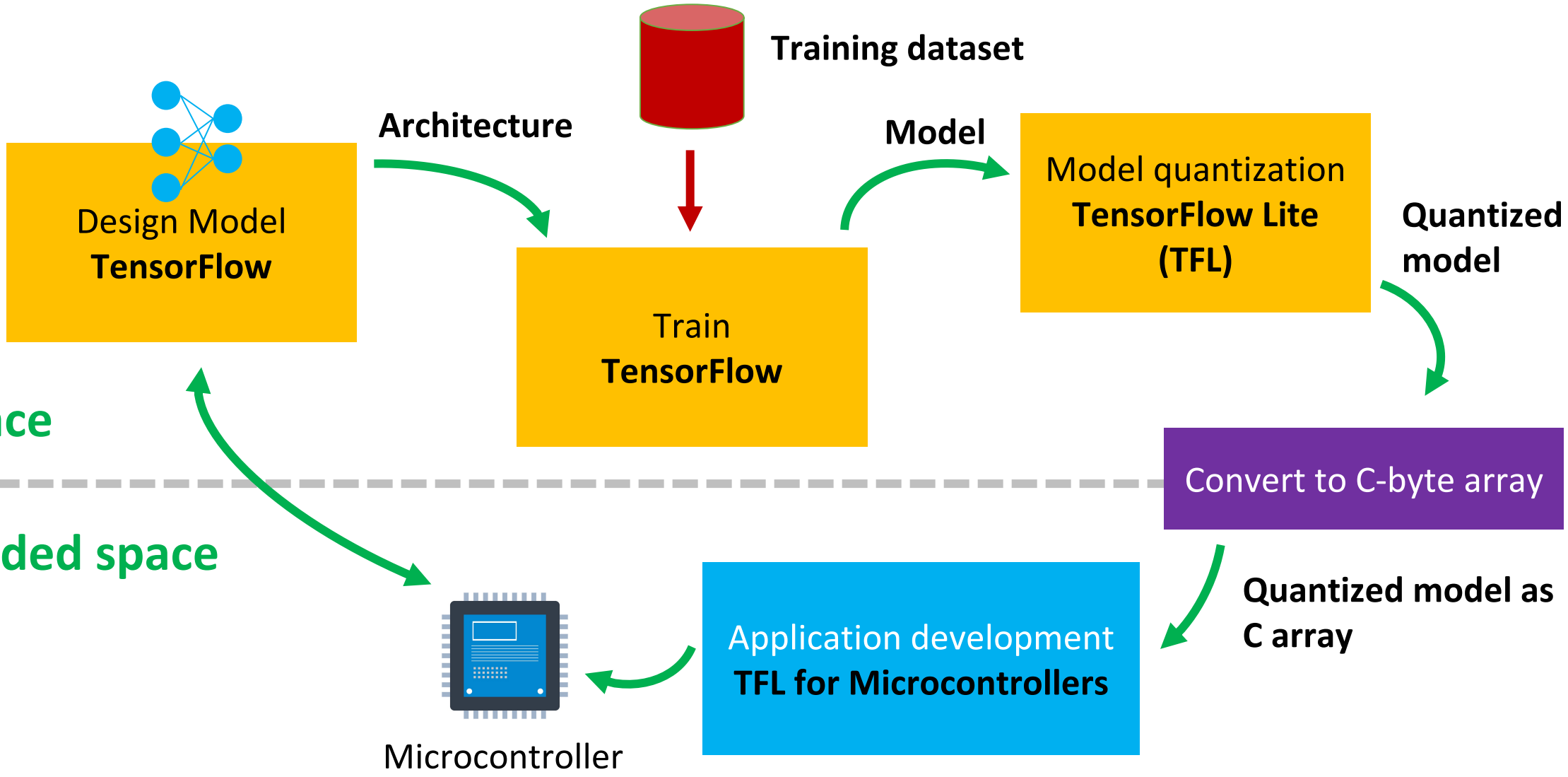
All the software algorithms have been accelerated on the Arm Cortex-M CPU. The tricks were:

- Perform the computation only when strictly necessary
- Pre-compute portion of the computation ahead of time and store these values in

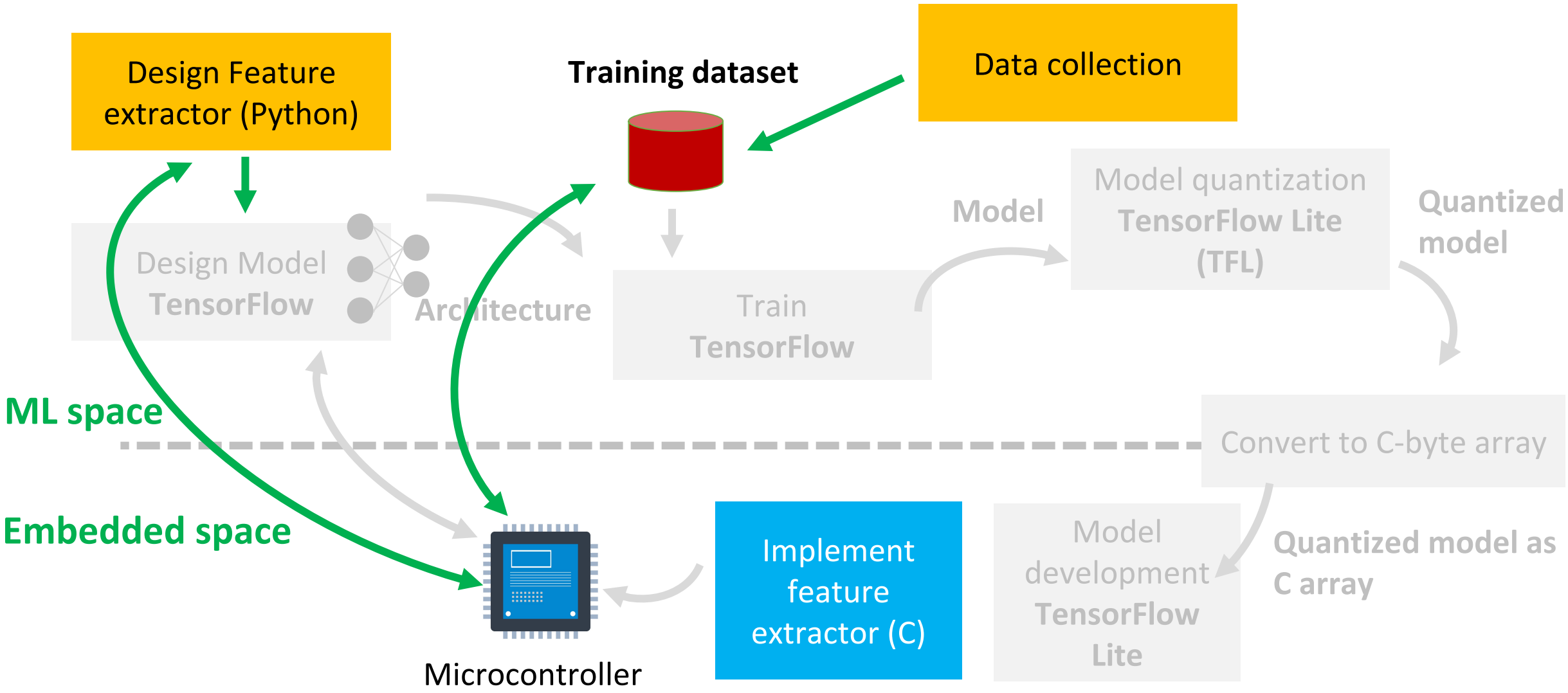
the program memory

From this project, I learned some of the software optimizations we will discuss today for the MFCCs feature extraction.

The simple view of tinyML



The complex view of tinyML



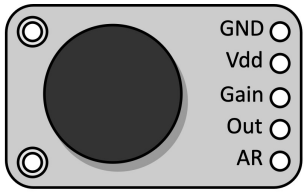
The additional ingredients

- Collect data samples with the target device to improve the robustness of the trained model in a real scenario
- Select the suitable ML input data for the target device
- Implement a feature extractor in Python suitable for the target device
- Implement a feature extractor in C that numerically matches what is implemented in Python

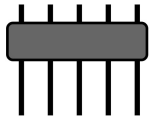
Data Collection

Connecting the microphone to the Raspberry Pi Pico 1of2

- Our application requires a microphone to record songs and classify their music genre.
- Since the Raspberry Pi Pico does not have a built-in microphone, we need to employ an external one and figure out the appropriate way to connect it to the microcontroller.
- We have opted for an electret microphone with the MAX9814 amplifier.
- This microphone was also chosen to demonstrate how to sample signals with the ADC peripheral.

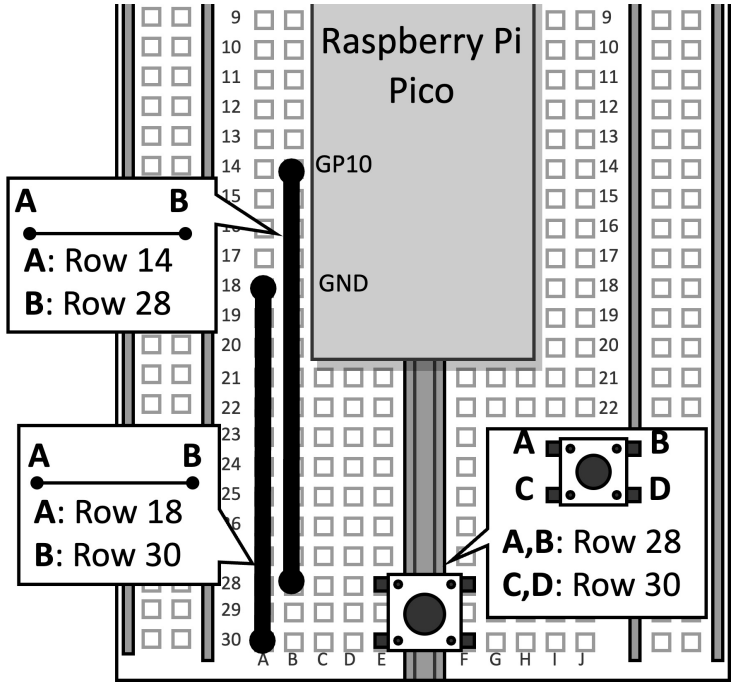


Electret microphone with MAX9814

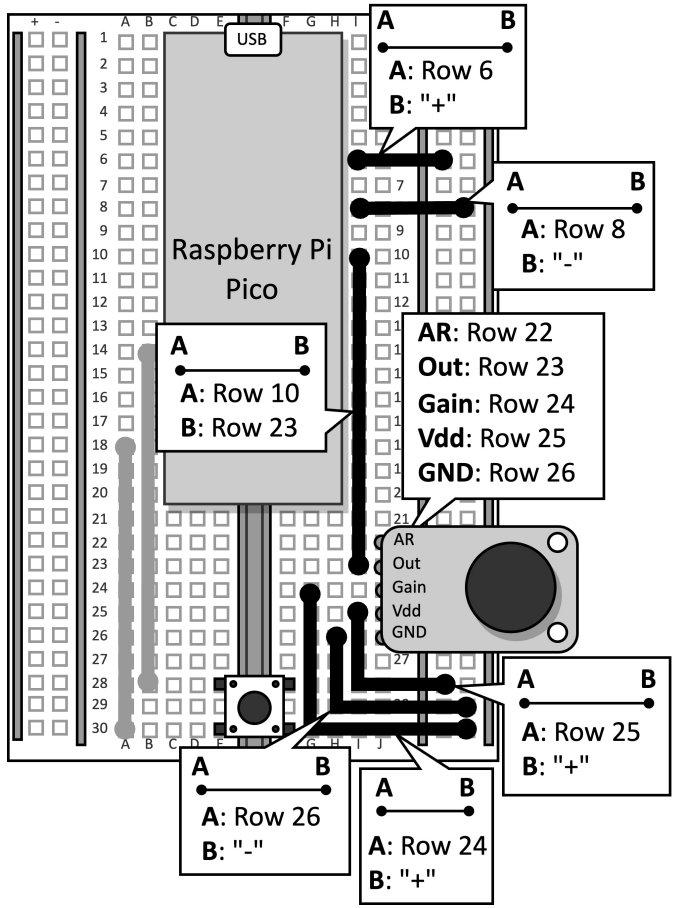

Five pins header strip

TinyML
Cookbook, 2Ed

Connecting the microphone to the Raspberry Pi Pico 2of2



Step 1



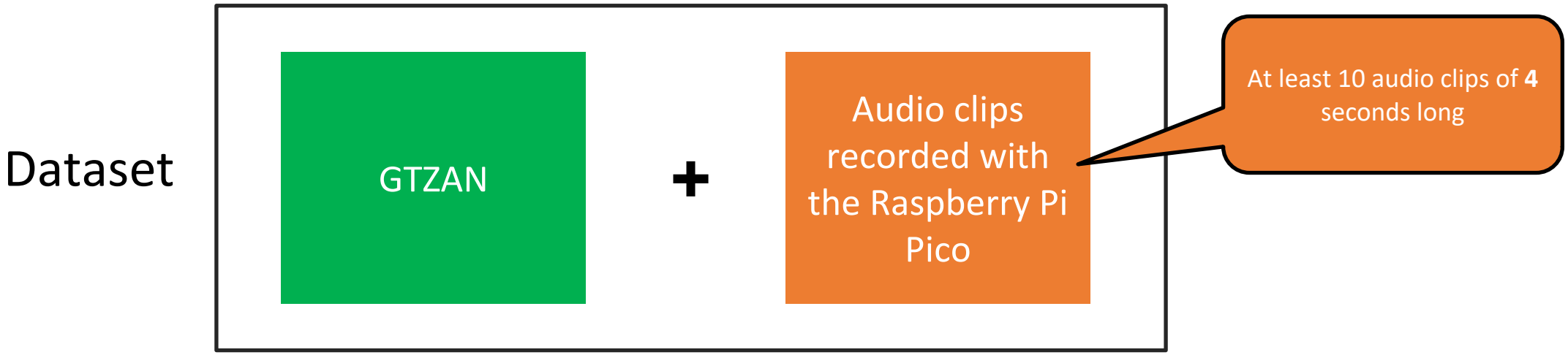
Step 2

Uploading audio clips to Google Drive

- Implemented an Arduino sketch to sample the audio signal with ADC and transmit the samples over the serial.
- Implement a Python script to acquire the audio samples transmitted over the serial, create the audio file, and upload it into Google Drive for access in the Colab notebook.
- The audio clips are recorded every time the user presses the push-button connected to the Raspberry Pi Pico.

Preparing the Dataset

Preparing the dataset for music genre recognition



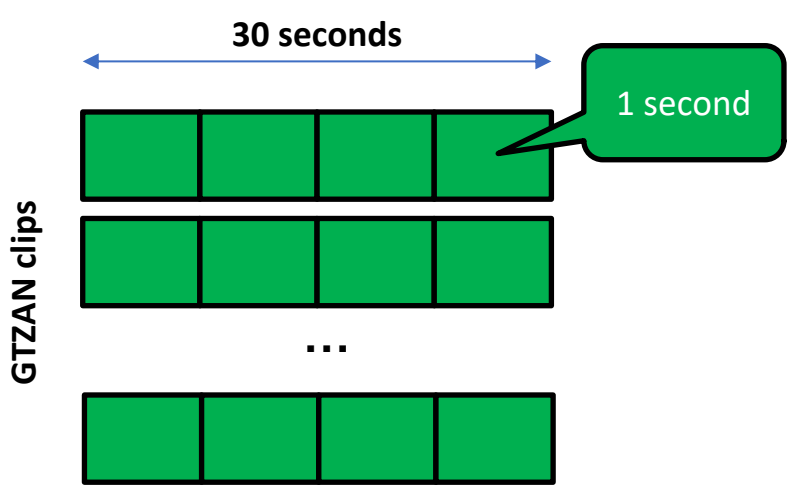
- GTZAN has 10 music genres: blues, classical, country, disco, hip hop, jazz, metal, pop, reggae, and rock
- Each genre has 100 audio clips that are 30 seconds long
- Each audio clip has the following:
  - 22050 Hz sample rate
  - 16 bit integer samples
  - Mono channel

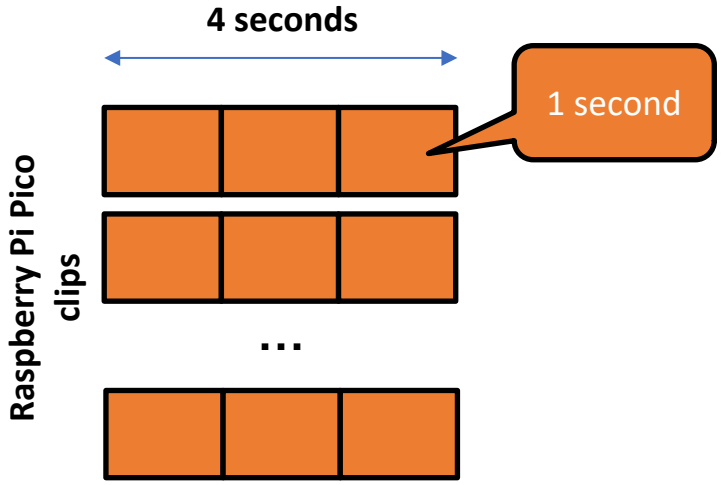
Choosing the suitable model input length

- Given that the Raspberry Pi Pico has 264 KB of SRAM, it is impractical to consider both 30-second and 4-second audio inputs. In fact, the former requires over 1 MByte of memory, while the latter would demand 176.4 KB.

- As a result, we considered audio clips of 1 second long. This choice will help reduce the memory required to 44.1 KB

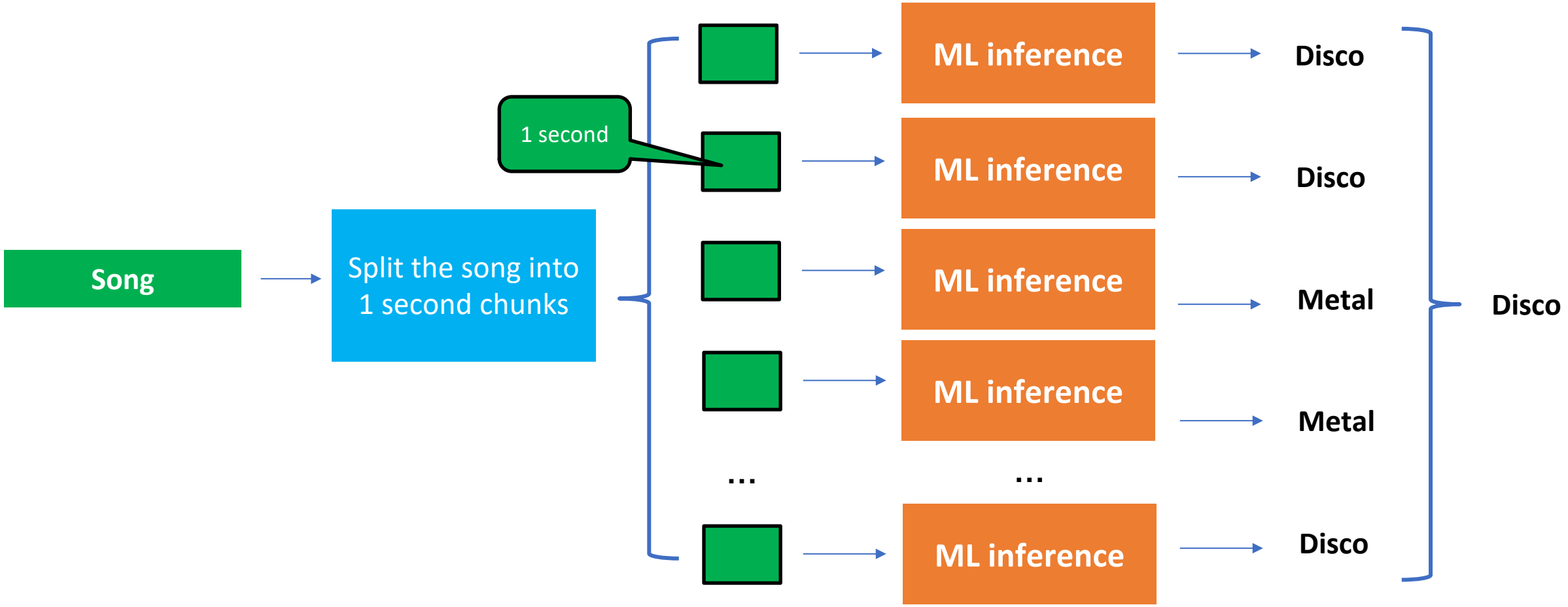
- The 1-second audio clips have been extracted from the GTZAN, and audio clips recorded with the microphone





Is 1-sec model input length enough?

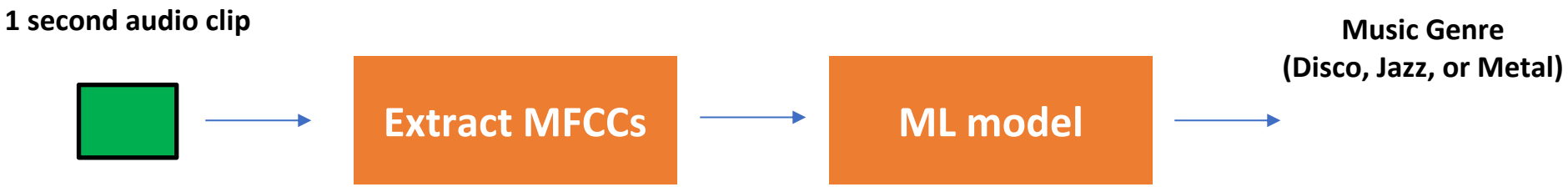
- Recognizing the music genre from just 1 second of audio can be challenging.
- However, we can analyze multiple chunks of a song to classify its genre.



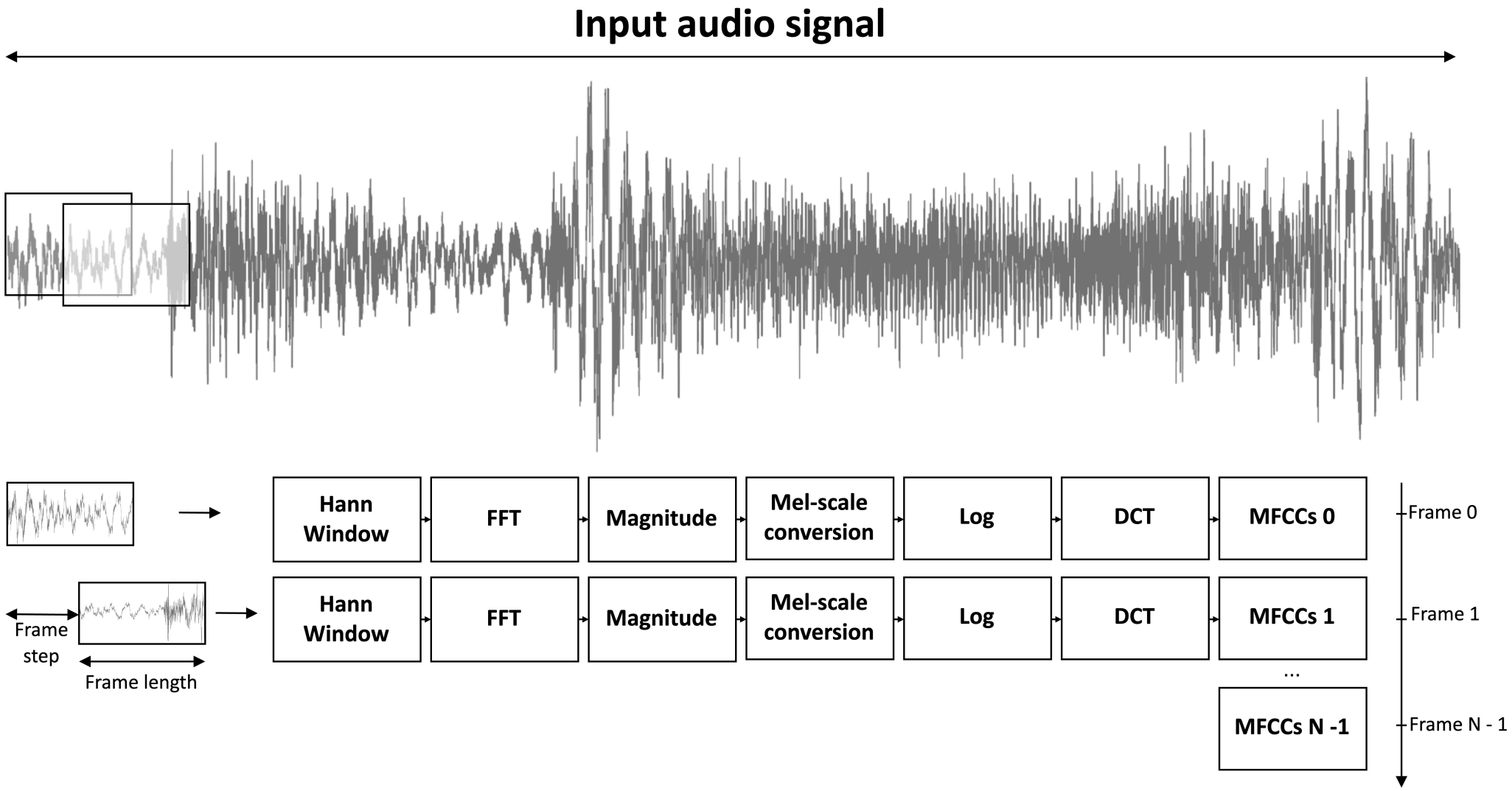
Feature Extraction

Extracting MFCCs 1of3

- Many acoustic models rely on hand-crafted engineering input features to achieve high accuracy and keep the ML model small.
- Mel Frequency Cepstral Coefficients (MFCCs) are extensively utilized in audio applications and have demonstrated remarkable success in various use cases, including music genre classification.



Extracting MFCCs 2of3



Extracting MFCCs 3of3

The extracted features by MFCCs depend on some hyperparameters, which are:

- **Frame length**: This is the duration of each frame extracted from the audio signal.
- **Frame step**: This is the distance between two consecutive frames
- **FFT length**: This is the number of samples used in the FFT to compute the spectrum. Typically, the FFT length is the same as the frame length.
- **Mel-spectrogram attributes**: These are the necessary attributes to compute the Mel-spectrogram, such as the minimum and maximum frequency to represent and the number of output Mel frequencies.
- **Number of MFCCs to extract**: This is the number of Discrete Cosine Transform coefficients to keep.

To determine the appropriate value for the hyperparameters, we need to get familiar with all the operations performed by MFCCs.

Applying the Hann Window

- Filter applied in the time domain to minimize the spectral leakage with the Fourier Transform.
- To apply this filter, we multiply the input frame element-wise with the Hann window coefficients:

$$W_{Hann}(i) = 0.5 \cdot (1 - \cos\left(\frac{2 \cdot \pi \cdot i}{N - 1}\right))$$

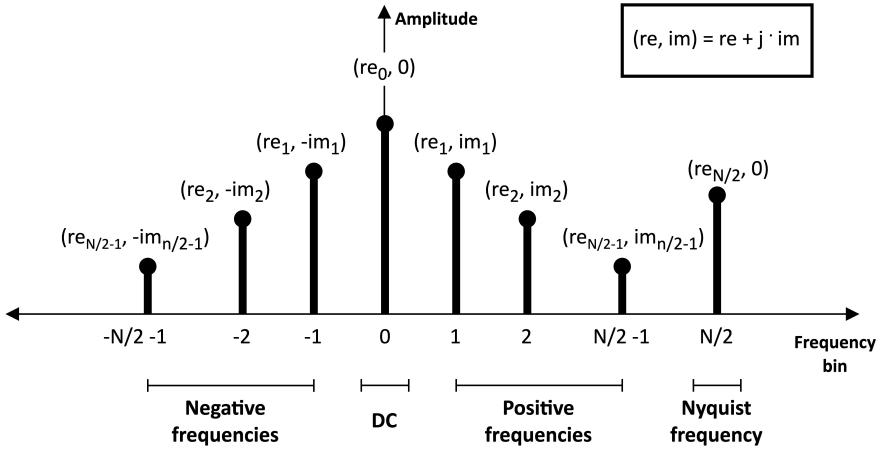
Index of the sample

Number of total samples in 1-sec audio clip

- Quite expensive computation, isn't it?
- However, the Hann coefficients can be precomputed…let's keep it in mind it ;)

Calculating the FFT

- The FFT is used to bring the signal into the frequency domain



The FFT on a real signal produces negative frequencies symmetric to the positive ones

- The distance between two consecutive frequencies is the frequency resolution, *depending on the input frame length*.
- *The shorter the frame length, the lower the frequency resolution*. For applications like ours, where fine-grained spectral details are crucial to capture harmonics and pitch, frame lengths ranging from 90 ms to 100 ms are generally needed.

Calculating the FFT magnitude

- The magnitude of the FFT conveys information about energy distribution and tells us the relevance of each frequency component

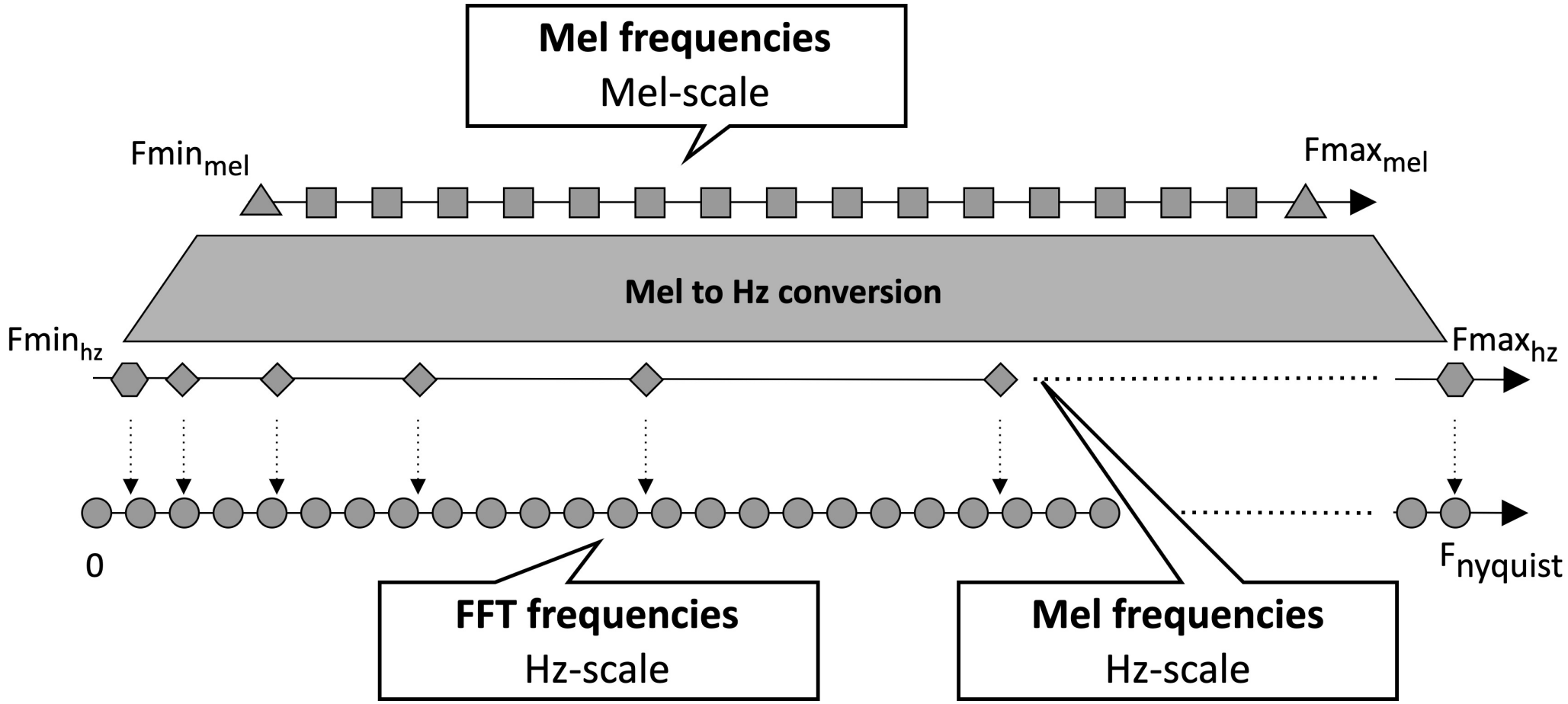
$$abs(x) = |x| = \sqrt{re^2 + im^2}$$

The Mel scale conversion 1of2

- The Mel frequency scale is a type of compression technique.
- Inspired by how the human auditory system perceives the frequencies, the Mel scale was introduced to apply a logarithmic transformation to the FFT frequencies.
- To apply this frequency conversion, we must first define the following:
    - **Fmin**: This is the lower frequency of the Mel scale.
    - **Fmax**: This is the higher frequency of the Mel scale.
    - **Number of Mel frequencies**: This is the number of evenly spaced frequencies between Fmin and Fmax on the Mel scale.

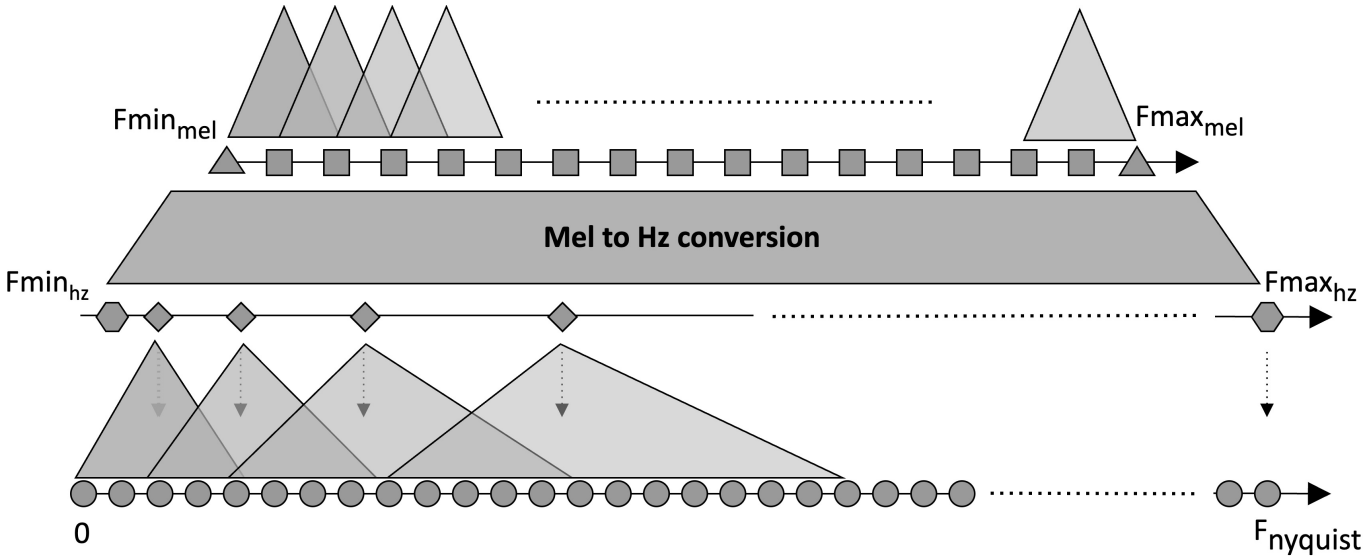
Due to the logarithmic nature of this conversion, we can use fewer Mel frequencies than FFT frequencies because many lower Mel frequencies correspond to the same Hz frequency.

The Mel scale conversion 2of2



How to apply the Mel scale conversion 1of2

- The magnitude of each Mel frequency is obtained *by applying a set of triangular filters (**filter banks**) in the Hz scale to capture the relevant FFT magnitudes that contribute to it*.

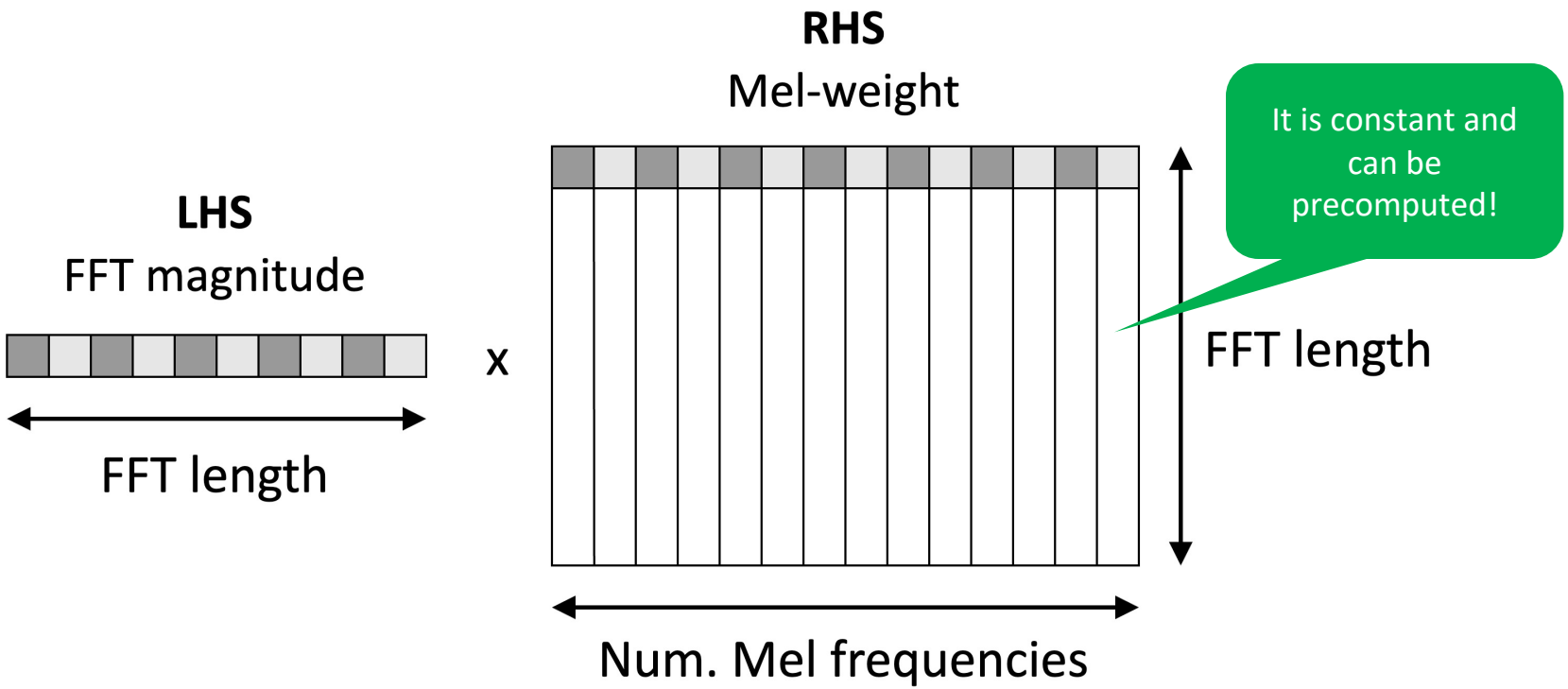


These triangular filters are for each Mel frequency and have been designed to:

- Be equally spaced on the Mel scale
- Have the apex of the triangle on each Mel frequency
- Have the two neighboring Mel frequencies that are adjacent to the center frequency of the filter as the vertices of the triangle's base

How to apply the Mel scale conversion 2of2

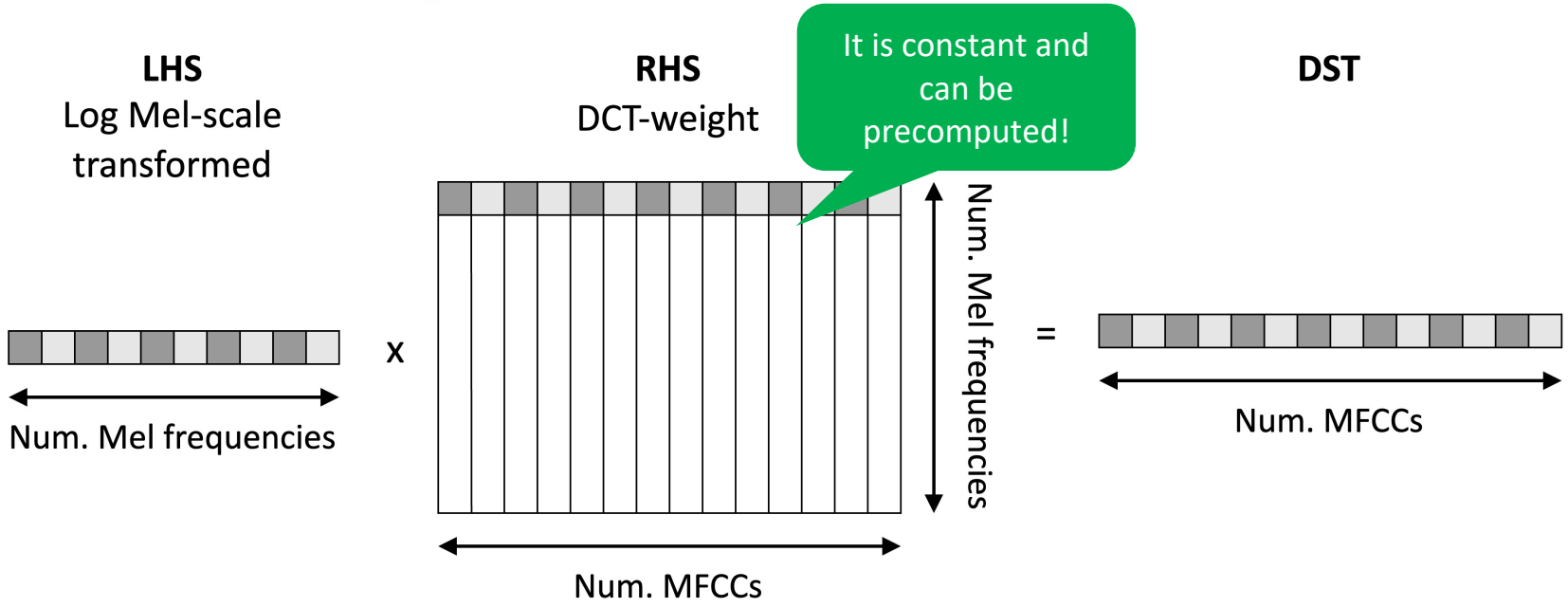
- The filter aims to assign a weight (Mel-weight) to each FFT frequency.
- The triangle's apex corresponds to the highest Mel-weight, set to 1.
- The base vertices of the triangle correspond to the lowest Mel-weight, set to 0.



Computing the DCT coefficients

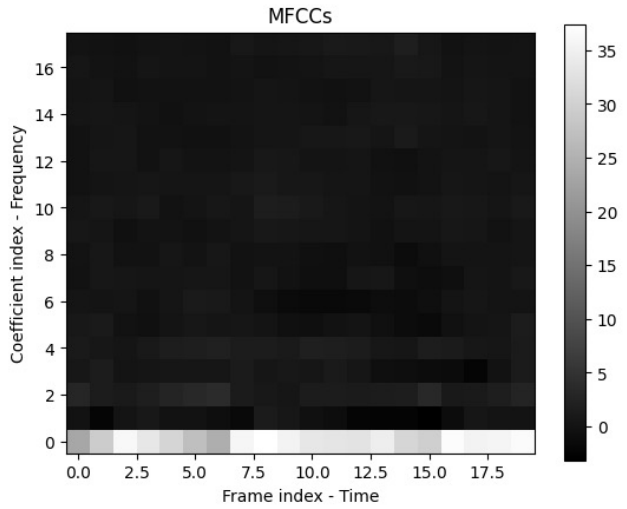
- MFCCs are the first few DCT coefficients that describe the spectrum's shape.
- The first DCT coefficient reflects the average power in the spectrum.
- The higher-order coefficients provide more detailed spectral information, such as the pitch

$$W_{DCT}(k) = \sqrt{\frac{2}{N}} \cdot \sum_{n=0}^{N-1} x(n) \cdot \cos\left(\frac{\pi}{N} \cdot (n + 0.5)\right) \cdot k$$

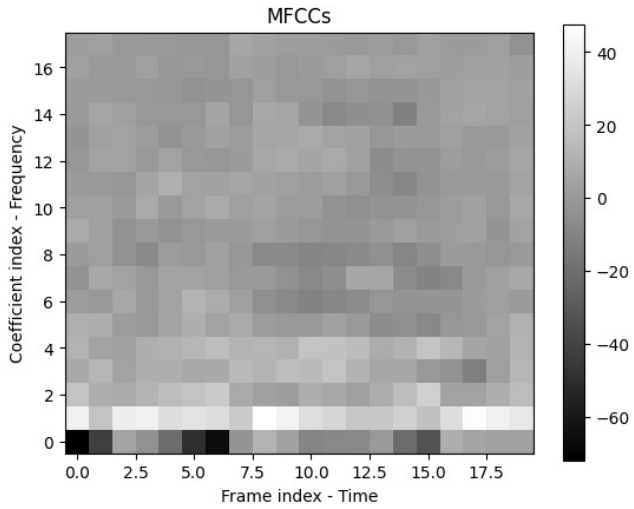


MFCCs: TensorFlow Vs Librosa 1of2

- For this project, we extracted the MFCCs features using TensorFlow.
- However, TensorFlow is not the only option to extract MFCCs from a signal. For example, we can use Librosa
- In the GitHub repository, we have included the code to demonstrate how to extract the MFCCs with Librosa.
- However, the resulting MFCCs obtained with Librosa differ from the ones obtained with TensorFlow due to their different underlying implementations.



TensorFlow



Librosa

MFCCs: TensorFlow Vs Librosa 2of2

Although we might have various Python libraries for extracting MFCCs features, we might have different options when deploying the model on the microcontroller.

Therefore, ***always consider how to leverage the computation on the target device*** because the same algorithm implemented in Python should be deployed on the microcontroller to avoid accuracy loss.

Optimize the Feature Extraction for the target device

Implement MFCCs with fixed-point arithmetic

- The TensorFlow implementation of the MFCCs feature extraction is done in floating-point.
- However, this data format is computationally inefficient for the Raspberry Pi Pico because it does not have floating-point hardware acceleration as many other microcontrollers.
- Therefore, the idea is to accelerate the blocks of the MFCCs using fixed-point arithmetic (Q15) with integer data types.
- These blocks are element-wise multiplication, FFT, Complex magnitude, and vector-by-matrix.

Interesting idea…but how can we do it in Python and C?

The CMSIS-DSP Python library

- For Arm-based microcontrollers, such as the Raspberry Pi Pico, implementing an algorithm can be made simple and efficient using the C functions available in the **CMSIS-DSP library** (https://github.com/ARM-software/CMSIS-DSP)
- This library provides optimized functions for various applications and Arm Cortex-M CPUs.

- Some examples of these routines include math functions, matrix functions, and domain transformation (like FFT) functions. Therefore, exactly what we need!

- The CMSIS-DSP library is available in C and as a Python library, enabling us to test and develop an algorithm in a Python environment that closely resembles the final implementation on the microcontroller.

In the book, there is a detailed explanation of fixed-point arithmetic and the MFCCs implementation.

CMSIS-DSP Example

Python

```
import cmsidsp

for i in range(NUM_FRAMES):
   ….

   # Apply the Hann Window.
  hann_q15 = dsp.arm_mult_q15(frame_q15, hann_lut_q15)

   # Apply the RFFT.
   inst = dsp.arm_rfft_instance_q15()
   stat = dsp.arm_rfft_init_q15(inst, FFT_LENGTH, 0, 1)
   fft_q = dsp.arm_rfft_q15(hann_q15)
```

C

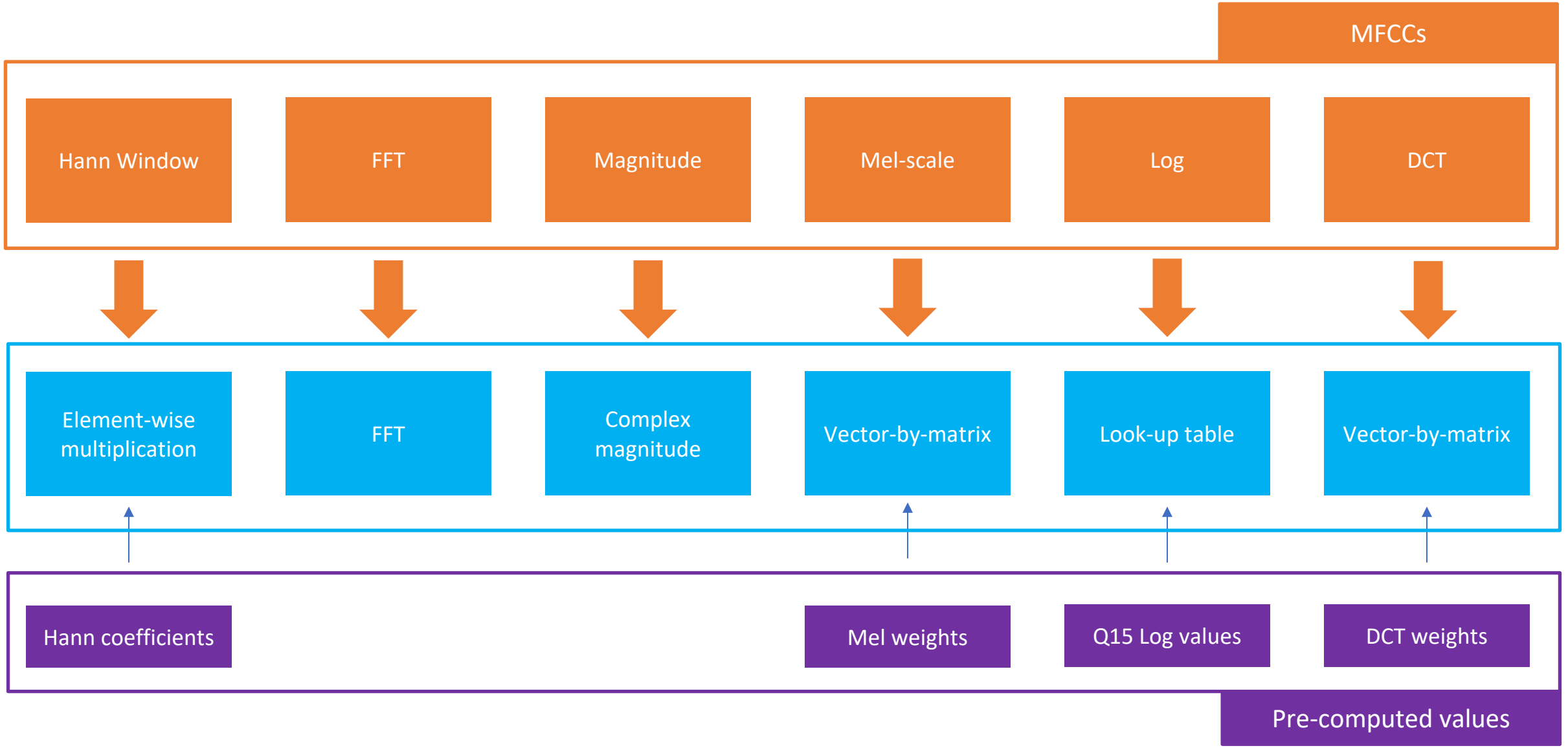
```
#include <arm_math.h>

for(int i = 0; i < NUM_FRAMES; ++i ) {
      ….

    // Apply the Hann Window.
    arm_mult_q15((q15_t*)&src[i*FRAME_STEP],
                                  (q15_t*)hann_lut_q15_data,
                                   _bufA, FRAME_LENGTH);

    // Apply the RFFT.
    arm_rfft_instance_q15   _rfft_inst;
    arm_rfft_init_q15(&_rfft_inst, FFT_LENGTH, 0, 1);
    arm_rfft_q15(&_rfft_inst, _bufA, _bufB);
```

The implementation 1of2



The implementation 2of2

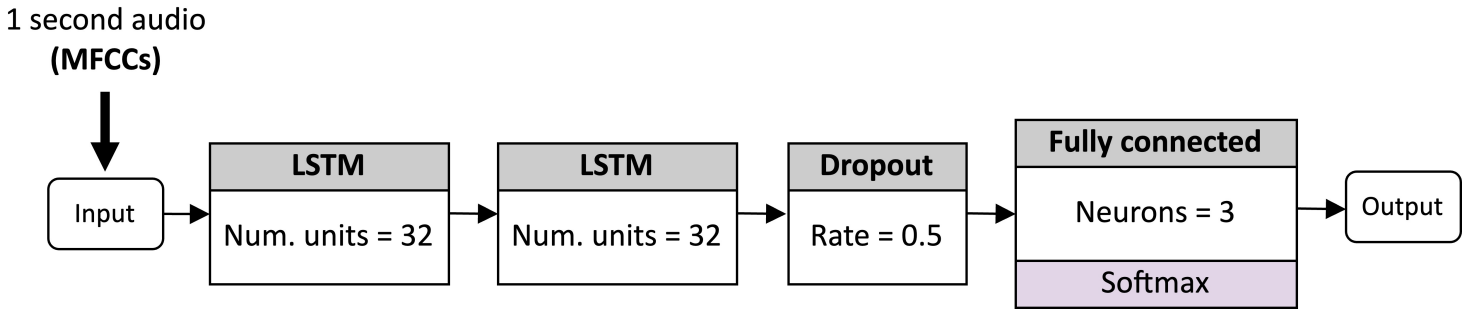
- The pre-computed values are stored in Flash (**153** Kbytes, 8% of total program memory available)
- Even if the Q15 implementation does not match the floating-point implementation numerically, it is not an issue, as the network will be trained on the Q15 MFCCs.
- By implementing the feature extraction in Python that numerically matches what will be employed on the target device, we minimize the risk of accuracy drop when deploying the model

The remaining part of Chapter 6

TinyML
Cookbook, 2Ed

The other recipes of Chapter 6

- Designed and trained an LSTM RNN (many-to-one)



Total trainable parameters: **~15K**
Accuracy (Float): 90%
Loss (Float): 0.09%

- Quantized the model to Int8 using the TensorFlow Lite Converter
- Evaluated the accuracy of the quantized model on the test dataset
- Implemented the MFCCs feature extraction in C in the Arduino IDE
- Built an application to recognize music genres (disco, jazz, and metal)

Model size: **23.5 KBytes**
Accuracy (Quantized): 90%

The Arduino tflite-micro library

- Unfortunately, an official pre-built Arduino tflite-micro library that supports all Arduino microcontrollers, including the Arduino Nano 33 BLE Sense, Raspberry Pi Pico, and the SparkFun RedBoard Artemis Nano, is not available.

- As a solution, we have included a pre-built tflite-micro Arduino library on GitHub, designed to work on any Arduino-compatible platform with an Arm Cortex-M CPU.

- The library derived from the Arduino tflite-micro library for the Arduino Nano 33 BLE Sense (credit to Pete Warden!)

- For those eager to know the necessary modifications to apply to the original Arduino tflite-micro library to achieve compatibility with all Arduino-compatible platforms, we have included a Colab notebook on GitHub.

Conclusion

- TinyML is a unique technology where hardware and software must know each other at best.

- Although these devices look “tiny,” they offer enough computational power and memory for many real-world use cases.

- The best friend of tinyML is the open-source community. If this technology is where it is now, it is because of the contributions of many developers worldwide.

Copyright Notice

www.tinyml.org

Copyright Notice

www.tinyml.org