tinyML® EMEA

Enabling Ultra-low Power Machine Learning at the Edge

June 26 - 28, 2023

TINY ML

www.tinyML.org

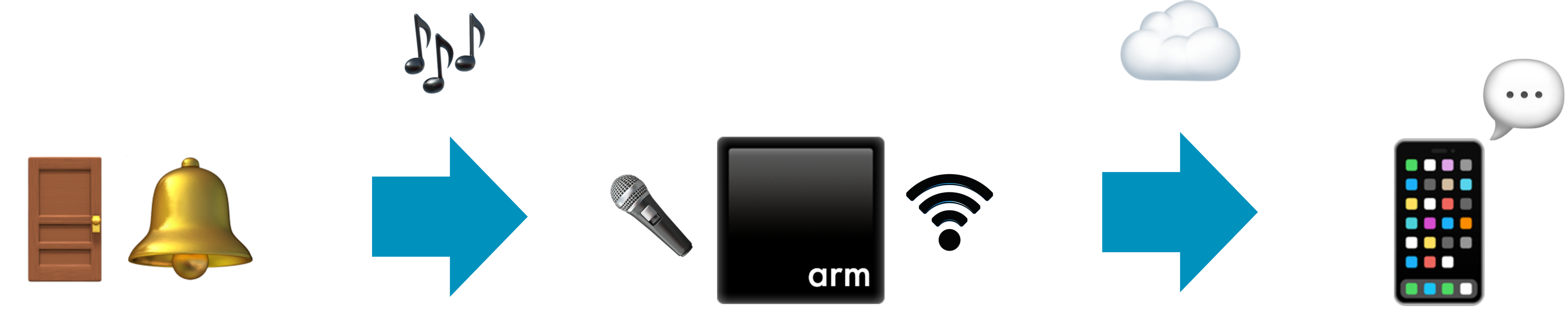
How to build an ML-powered doorbell notifier

Sandeep Mistry

June 26, 2023

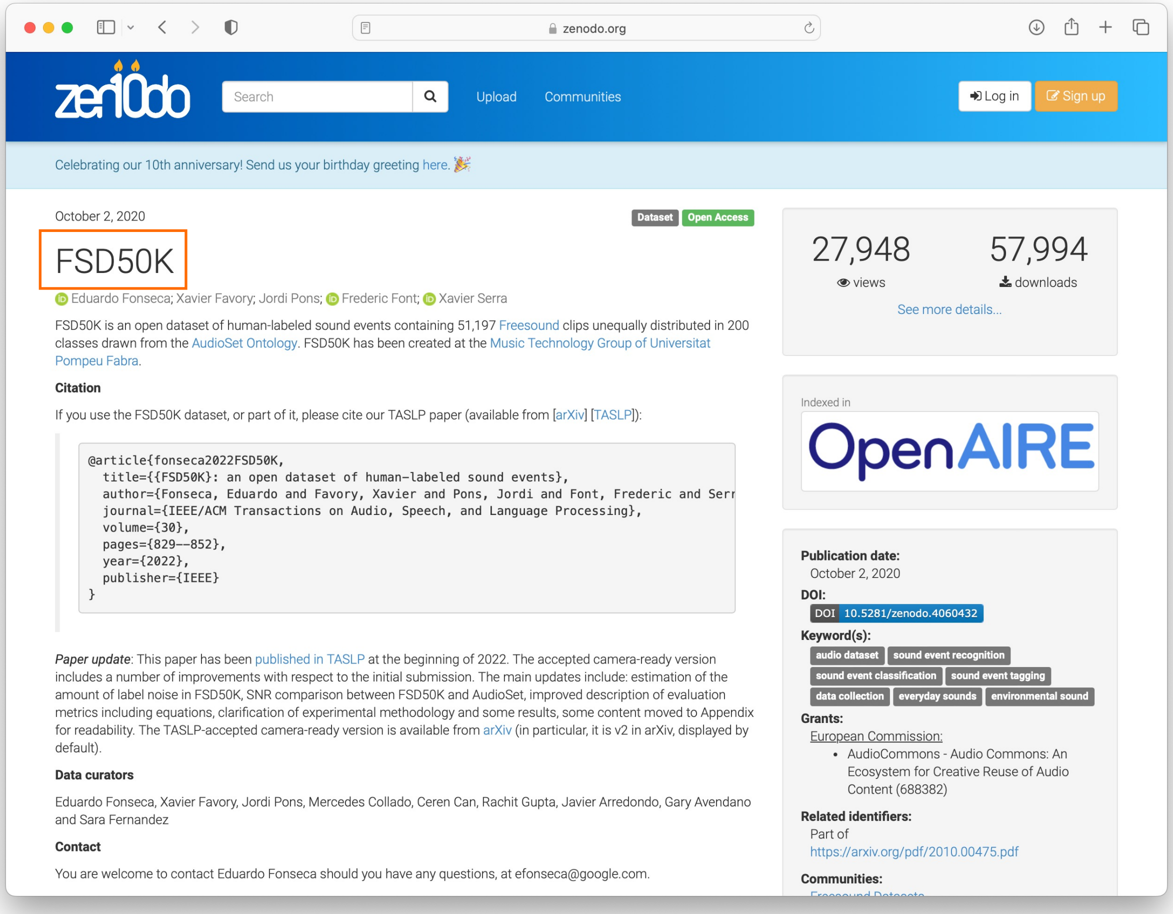
🏡 David Henry’s Garden Office

💡 Machine Learning + Microcontroller based solution



On-device ML Inferencing = Privacy Preserving

Dataset



Celebrating our 10th anniversary! Send us your birthday greeting here. 🎉

October 2, 2020

Dataset Open Access

FSD50K

Eduardo Fonseca; Xavier Favory; Jordi Pons; Frederic Font; Xavier Serra

FSD50K is an open dataset of human-labeled sound events containing 51,197 Freesound clips unequally distributed in 200 classes drawn from the AudioSet Ontology. FSD50K has been created at the Music Technology Group of Universitat Pompeu Fabra.

Citation

If you use the FSD50K dataset, or part of it, please cite our TASLP paper (available from [arXiv] [TASLP]):

```
@article{fonseca2022FSD50K,
  title={{FSD50K}: an open dataset of human-labeled sound events},
  author={Fonseca, Eduardo and Favory, Xavier and Pons, Jordi and Font, Frederic and Serr
  journal={IEEE/ACM Transactions on Audio, Speech, and Language Processing},
  volume={30},
  pages={829--852},
  year={2022},
  publisher={IEEE}
}
```

Paper update: This paper has been published in TASLP at the beginning of 2022. The accepted camera-ready version includes a number of improvements with respect to the initial submission. The main updates include: estimation of the amount of label noise in FSD50K, SNR comparison between FSD50K and AudioSet, improved description of evaluation metrics including equations, clarification of experimental methodology and some results, some content moved to Appendix for readability. The TASLP-accepted camera-ready version is available from arXiv (in particular, it is v2 in arXiv, displayed by default).

Data curators

Eduardo Fonseca, Xavier Favory, Jordi Pons, Mercedes Collado, Ceren Can, Rachit Gupta, Javier Arredondo, Gary Avendano and Sara Fernandez

Contact

You are welcome to contact Eduardo Fonseca should you have any questions, at efonseca@google.com.

27,948 views

57,994 downloads

See more details...

Indexed in

Publication date:
October 2, 2020

DOI:
DOI 10.5281/zenodo.4060432

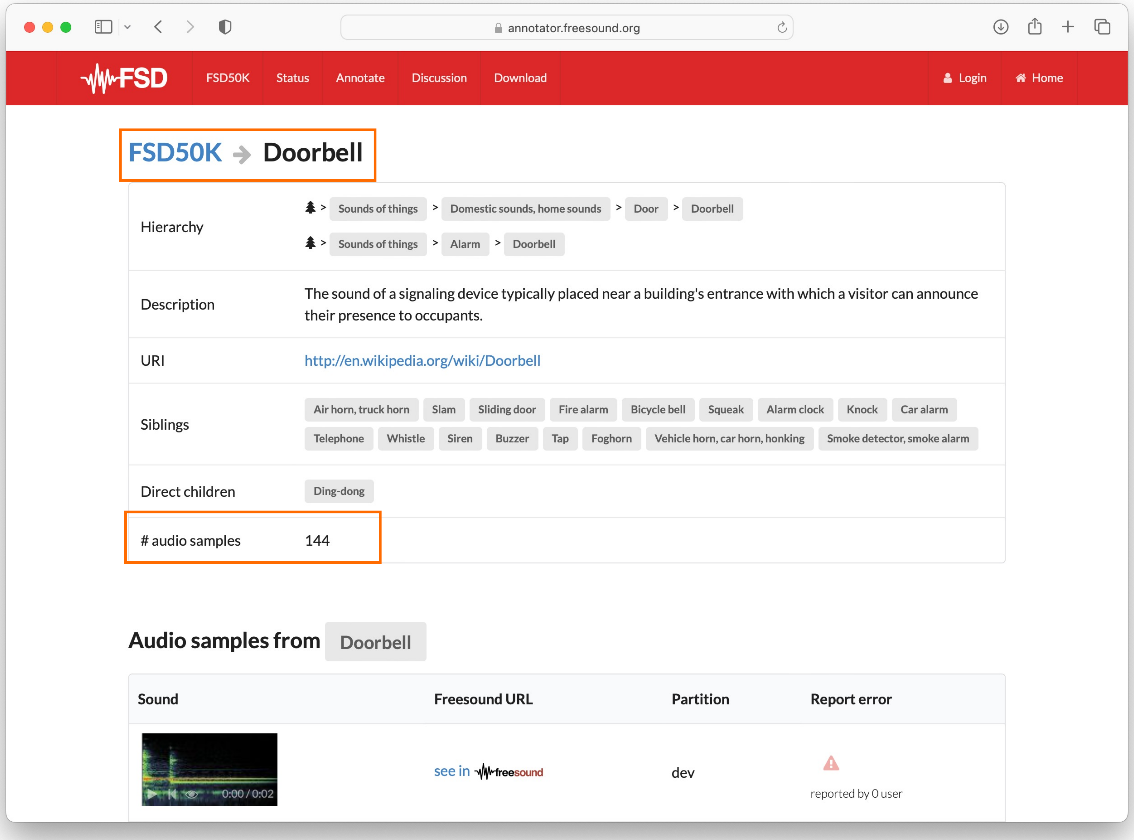
Keyword(s):
audio dataset · sound event recognition · sound event classification · sound event tagging · data collection · everyday sounds · environmental sound

Grants:
European Commission:

- AudioCommons - Audio Commons: An Ecosystem for Creative Reuse of Audio Content (688382)

Related identifiers:
Part of
https://arxiv.org/pdf/2010.00475.pdf

Communities:
Freesound Datasets



FSD50K → Doorbell

Hierarchy	> Sounds of things > Domestic sounds, home sounds > Door > Doorbell > Sounds of things > Alarm > Doorbell
Description	The sound of a signaling device typically placed near a building's entrance with which a visitor can announce their presence to occupants.
URI	http://en.wikipedia.org/wiki/Doorbell
Siblings	Air horn, truck horn · Slam · Sliding door · Fire alarm · Bicycle bell · Squeak · Alarm clock · Knock · Car alarm · Telephone · Whistle · Siren · Buzzer · Tap · Foghorn · Vehicle horn, car horn, honking · Smoke detector, smoke alarm
Direct children	Ding-dong
# audio samples	144

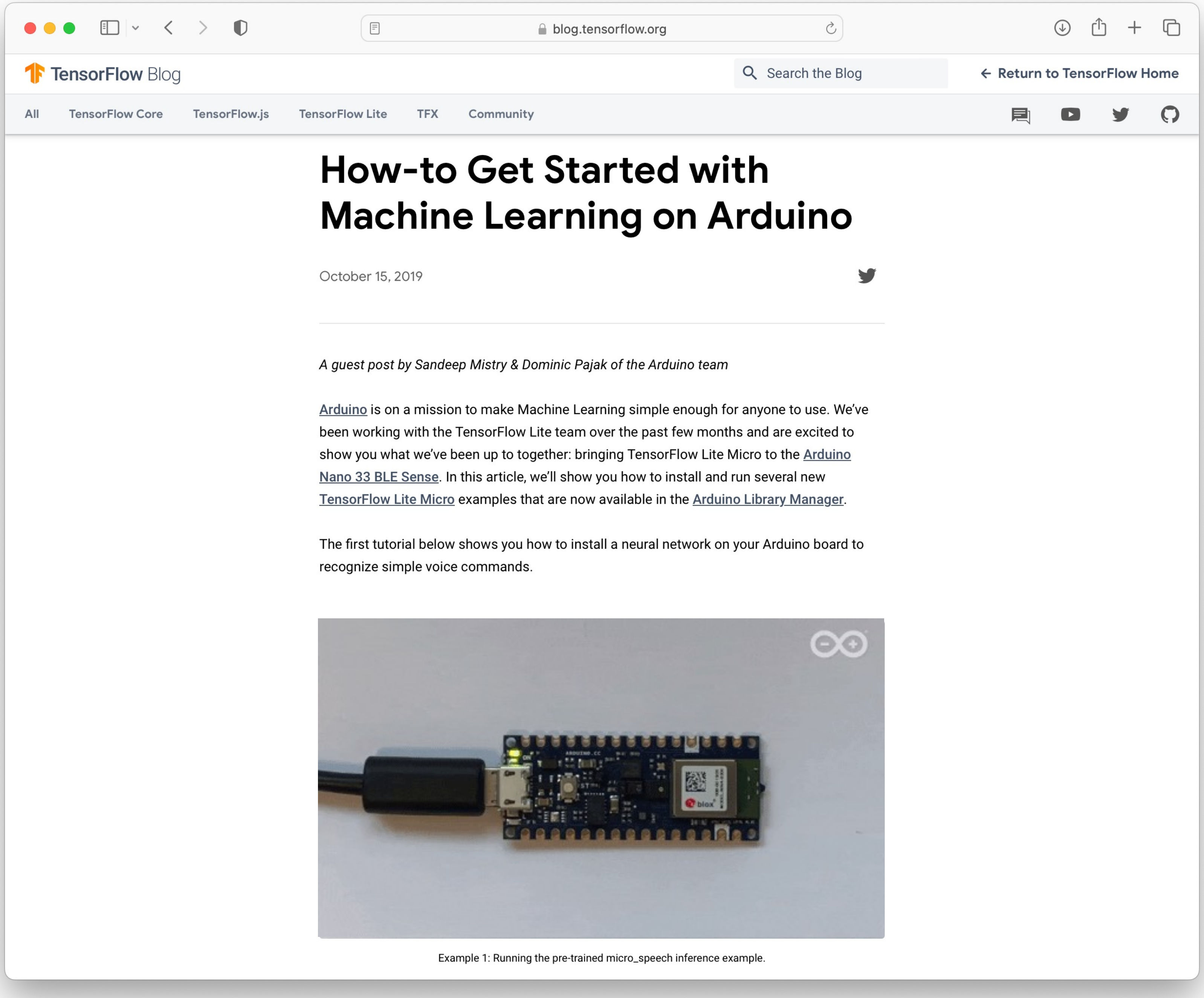
Audio samples from Doorbell

Sound	Freesound URL	Partition	Report error
0:00 / 0:02	see in freesound	dev	reported by 0 user

🔊 Audio Classification – FSD50K subset

- Doorbell - 🚪🔔
- Music - 🎶
- Domestic and home sounds - 🏠
- Human voice - 🗣️
- Hands (clapping, finger snapping) - 👏 🫰

ML Model



How-to Get Started with Machine Learning on Arduino

October 15, 2019

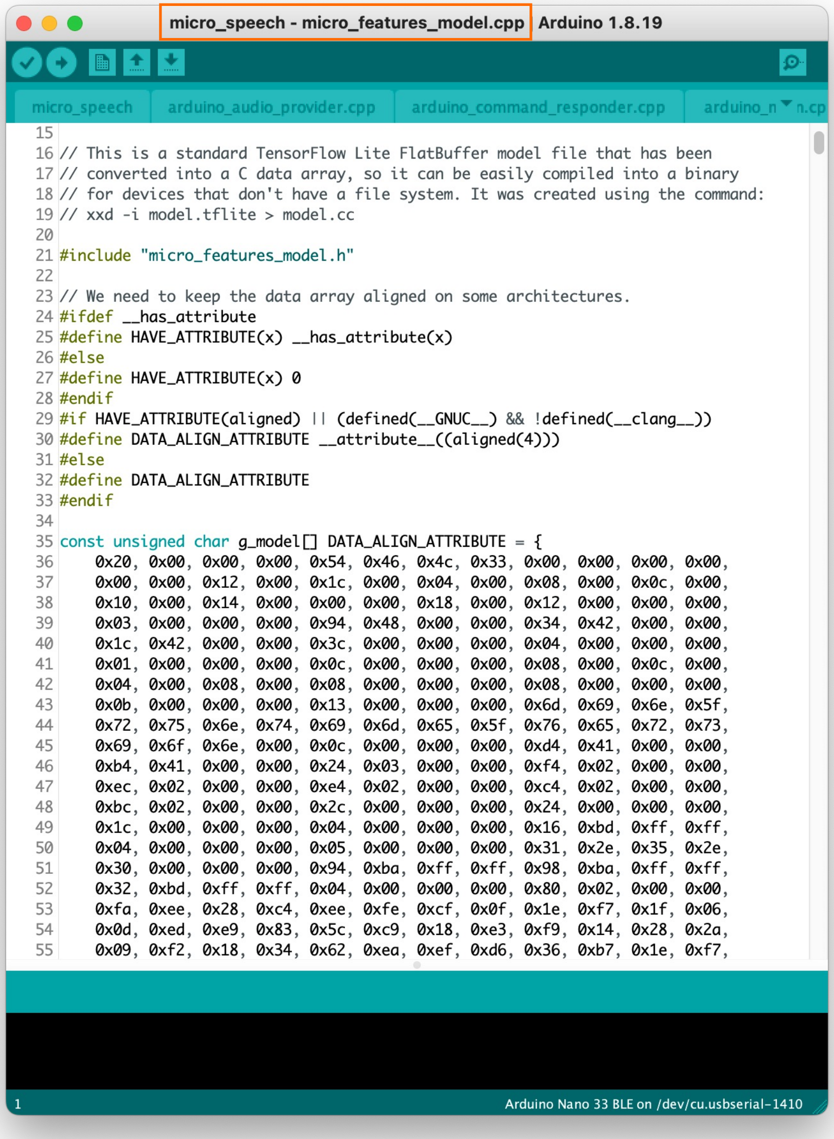
A guest post by Sandeep Mistry & Dominic Pajak of the Arduino team

Arduino is on a mission to make Machine Learning simple enough for anyone to use. We've been working with the TensorFlow Lite team over the past few months and are excited to show you what we've been up to together: bringing TensorFlow Lite Micro to the Arduino Nano 33 BLE Sense. In this article, we'll show you how to install and run several new TensorFlow Lite Micro examples that are now available in the Arduino Library Manager.

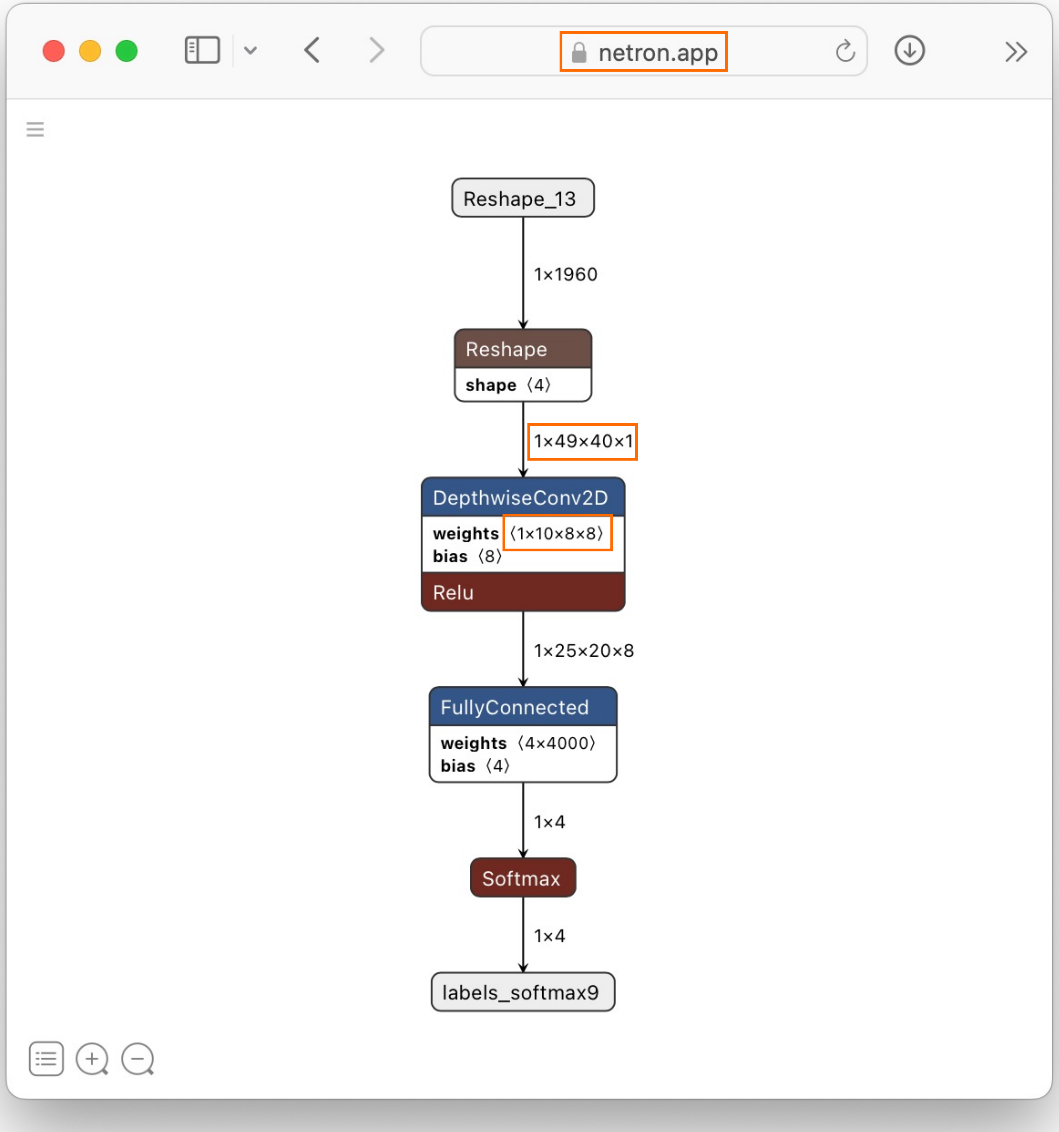
The first tutorial below shows you how to install a neural network on your Arduino board to recognize simple voice commands.

Example 1: Running the pre-trained micro_speech inference example.

ML Model

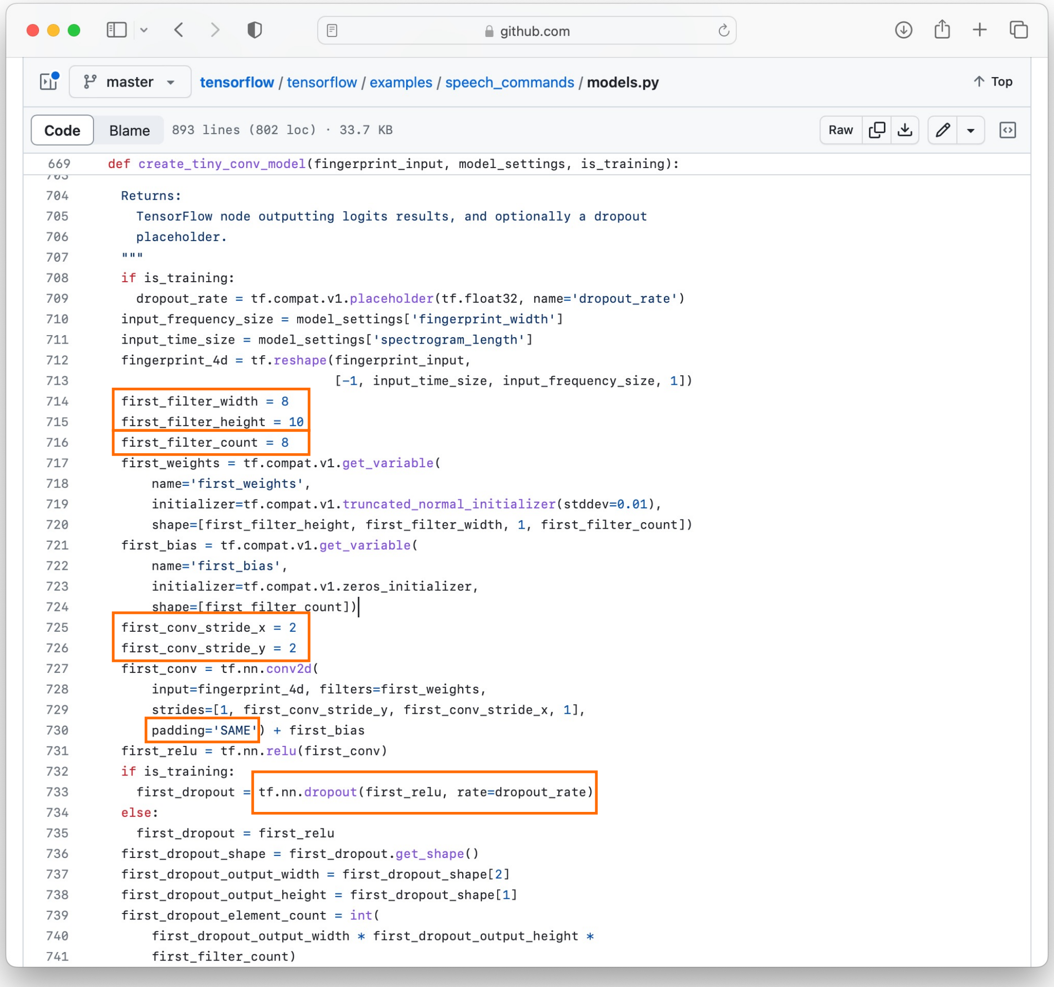

micro_speech - micro_features_model.cpp | Arduino 1.8.19
micro_speech
arduino_audio_provider.cpp
arduino_command_responder.cpp
arduino_n n.cp
1
Arduino Nano 33 BLE on /dev/cu.usbserial-1410

```
// This is a standard TensorFlow Lite FlatBuffer model file that has been
// converted into a C data array, so it can be easily compiled into a binary
// for devices that don't have a file system. It was created using the command:
// xxd -i model.tflite > model.cc

#include "micro_features_model.h"

// We need to keep the data array aligned on some architectures.
#ifdef __has_attribute
#define HAVE_ATTRIBUTE(x) __has_attribute(x)
#else
#define HAVE_ATTRIBUTE(x) 0
#endif
#if HAVE_ATTRIBUTE(aligned) || (defined(__GNUC__) && !defined(__clang__))
#define DATA_ALIGN_ATTRIBUTE __attribute__((aligned(4)))
#else
#define DATA_ALIGN_ATTRIBUTE
#endif

const unsigned char g_model[] DATA_ALIGN_ATTRIBUTE = {
    0x20, 0x00, 0x00, 0x00, 0x54, 0x46, 0x4c, 0x33, 0x00, 0x00, 0x00, 0x00,
    0x00, 0x00, 0x12, 0x00, 0x1c, 0x00, 0x04, 0x00, 0x08, 0x00, 0x0c, 0x00,
    0x10, 0x00, 0x14, 0x00, 0x00, 0x00, 0x18, 0x00, 0x12, 0x00, 0x00, 0x00,
    0x03, 0x00, 0x00, 0x00, 0x94, 0x48, 0x00, 0x00, 0x34, 0x42, 0x00, 0x00,
    0x1c, 0x42, 0x00, 0x00, 0x3c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00,
    0x01, 0x00, 0x00, 0x00, 0x0c, 0x00, 0x00, 0x00, 0x08, 0x00, 0x0c, 0x00,
    0x04, 0x00, 0x08, 0x00, 0x08, 0x00, 0x00, 0x00, 0x08, 0x00, 0x00, 0x00,
    0x0b, 0x00, 0x00, 0x00, 0x13, 0x00, 0x00, 0x00, 0x6d, 0x69, 0x6e, 0x5f,
    0x72, 0x75, 0x6e, 0x74, 0x69, 0x6d, 0x65, 0x5f, 0x76, 0x65, 0x72, 0x73,
    0x69, 0x6f, 0x6e, 0x00, 0x0c, 0x00, 0x00, 0x00, 0xd4, 0x41, 0x00, 0x00,
    0xb4, 0x41, 0x00, 0x00, 0x24, 0x03, 0x00, 0x00, 0xf4, 0x02, 0x00, 0x00,
    0xec, 0x02, 0x00, 0x00, 0xe4, 0x02, 0x00, 0x00, 0xc4, 0x02, 0x00, 0x00,
    0xbc, 0x02, 0x00, 0x00, 0x2c, 0x00, 0x00, 0x00, 0x24, 0x00, 0x00, 0x00,
    0x1c, 0x00, 0x00, 0x00, 0x04, 0x00, 0x00, 0x00, 0x16, 0xbd, 0xff, 0xff,
    0x04, 0x00, 0x00, 0x00, 0x05, 0x00, 0x00, 0x00, 0x31, 0x2e, 0x35, 0x2e,
    0x30, 0x00, 0x00, 0x00, 0x94, 0xba, 0xff, 0xff, 0x98, 0xba, 0xff, 0xff,
    0x32, 0xbd, 0xff, 0xff, 0x04, 0x00, 0x00, 0x00, 0x80, 0x02, 0x00, 0x00,
    0xfa, 0xee, 0x28, 0xc4, 0xee, 0xfe, 0xcf, 0x0f, 0x1e, 0xf7, 0x1f, 0x06,
    0x0d, 0xed, 0xe9, 0x83, 0x5c, 0xc9, 0x18, 0xe3, 0xf9, 0x14, 0x28, 0x2a,
    0x09, 0xf2, 0x18, 0x34, 0x62, 0xea, 0xef, 0xd6, 0x36, 0xb7, 0x1e, 0xf7,
```

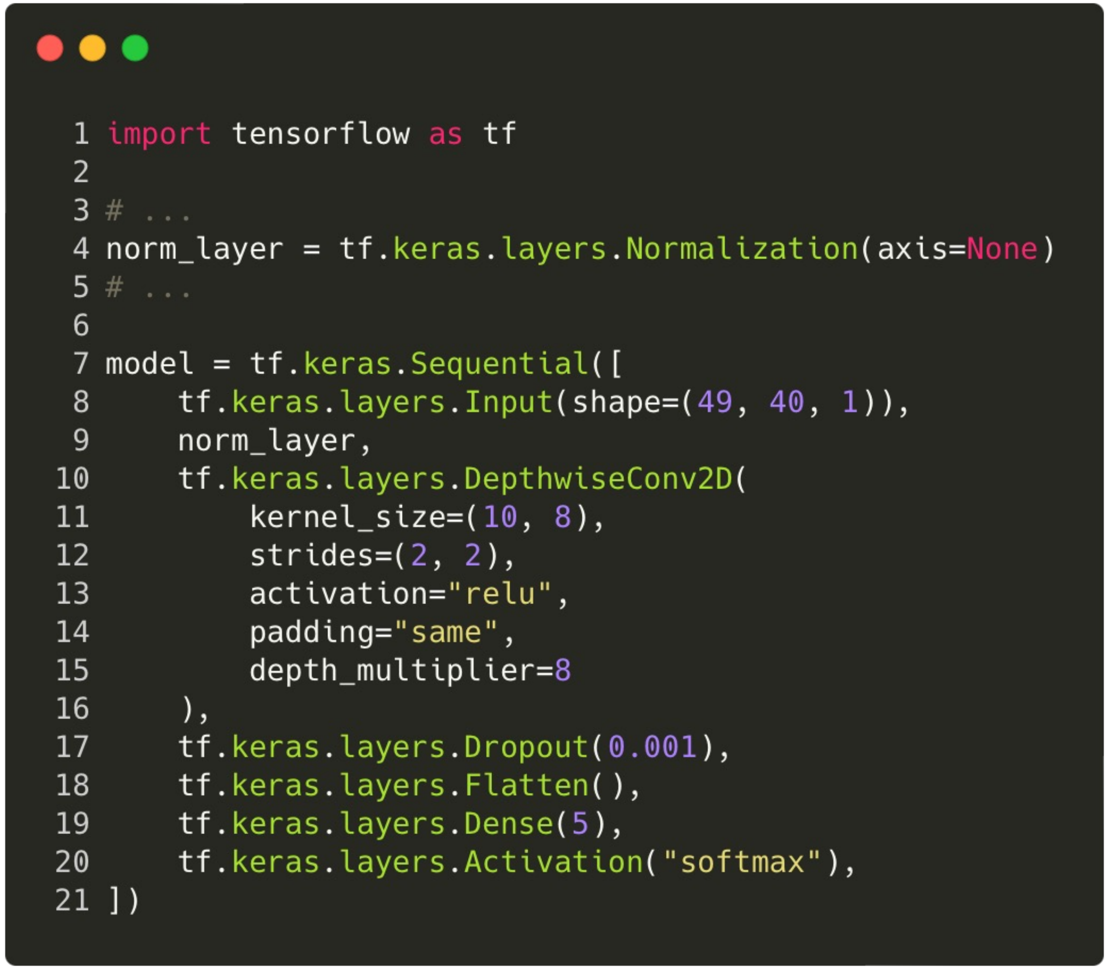


“tiny_conv” ML model

TensorFlow v1 vs TensorFlow v2 with Keras


github.com
master
tensorflow / tensorflow / examples / speech_commands / models.py
Top
Code
Blame
893 lines (802 loc) · 33.7 KB
Raw

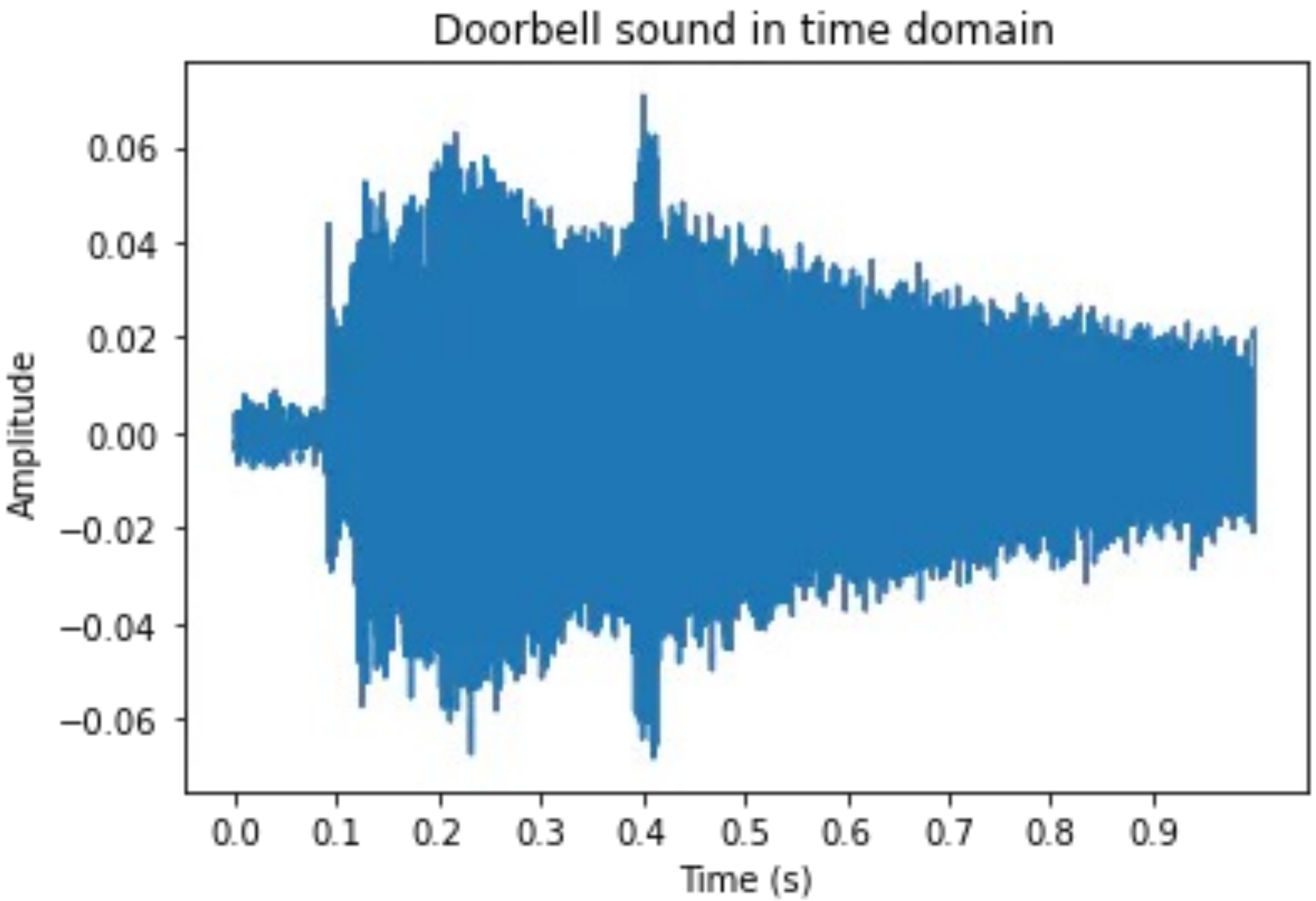
```
def create_tiny_conv_model(fingerprint_input, model_settings, is_training):
  Returns:
    TensorFlow node outputting logits results, and optionally a dropout
    placeholder.
  """
  if is_training:
    dropout_rate = tf.compat.v1.placeholder(tf.float32, name='dropout_rate')
  input_frequency_size = model_settings['fingerprint_width']
  input_time_size = model_settings['spectrogram_length']
  fingerprint_4d = tf.reshape(fingerprint_input,
                              [-1, input_time_size, input_frequency_size, 1])
  first_filter_width = 8
  first_filter_height = 10
  first_filter_count = 8
  first_weights = tf.compat.v1.get_variable(
      name='first_weights',
      initializer=tf.compat.v1.truncated_normal_initializer(stddev=0.01),
      shape=[first_filter_height, first_filter_width, 1, first_filter_count])
  first_bias = tf.compat.v1.get_variable(
      name='first_bias',
      initializer=tf.compat.v1.zeros_initializer,
      shape=[first_filter_count])
  first_conv_stride_x = 2
  first_conv_stride_y = 2
  first_conv = tf.nn.conv2d(
      input=fingerprint_4d, filters=first_weights,
      strides=[1, first_conv_stride_y, first_conv_stride_x, 1],
      padding='SAME') + first_bias
  first_relu = tf.nn.relu(first_conv)
  if is_training:
    first_dropout = tf.nn.dropout(first_relu, rate=dropout_rate)
  else:
    first_dropout = first_relu
  first_dropout_shape = first_dropout.get_shape()
  first_dropout_output_width = first_dropout_shape[2]
  first_dropout_output_height = first_dropout_shape[1]
  first_dropout_element_count = int(
      first_dropout_output_width * first_dropout_output_height *
      first_filter_count)
```



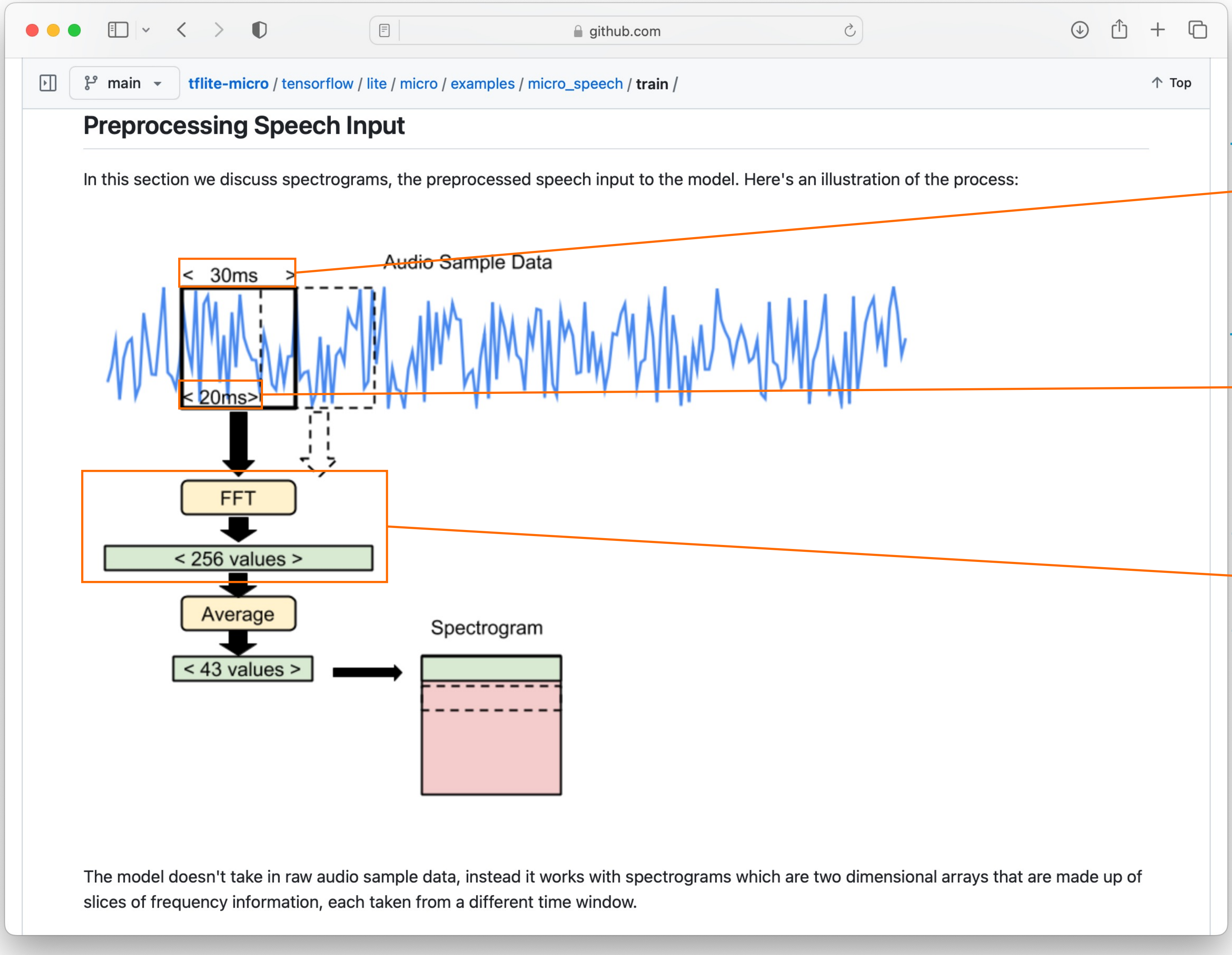
```
import tensorflow as tf

# ...
norm_layer = tf.keras.layers.Normalization(axis=None)
# ...

model = tf.keras.Sequential([
    tf.keras.layers.Input(shape=(49, 40, 1)),
    norm_layer,
    tf.keras.layers.DepthwiseConv2D(
        kernel_size=(10, 8),
        strides=(2, 2),
        activation="relu",
        padding="same",
        depth_multiplier=8
    ),
    tf.keras.layers.Dropout(0.001),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(5),
    tf.keras.layers.Activation("softmax"),
])
```

🎶 Input Signal

1 second of audio @ 16 kHz = 16,000 samples



Preprocessing



github.com
main
tflite-micro / tensorflow / lite / micro / examples / micro_speech / train /
Top

Preprocessing Speech Input

In this section we discuss spectrograms, the preprocessed speech input to the model. Here's an illustration of the process:

The model doesn't take in raw audio sample data, instead it works with spectrograms which are two dimensional arrays that are made up of slices of frequency information, each taken from a different time window.

Sample Rate = 16 kHz

Frame Length = 30 ms

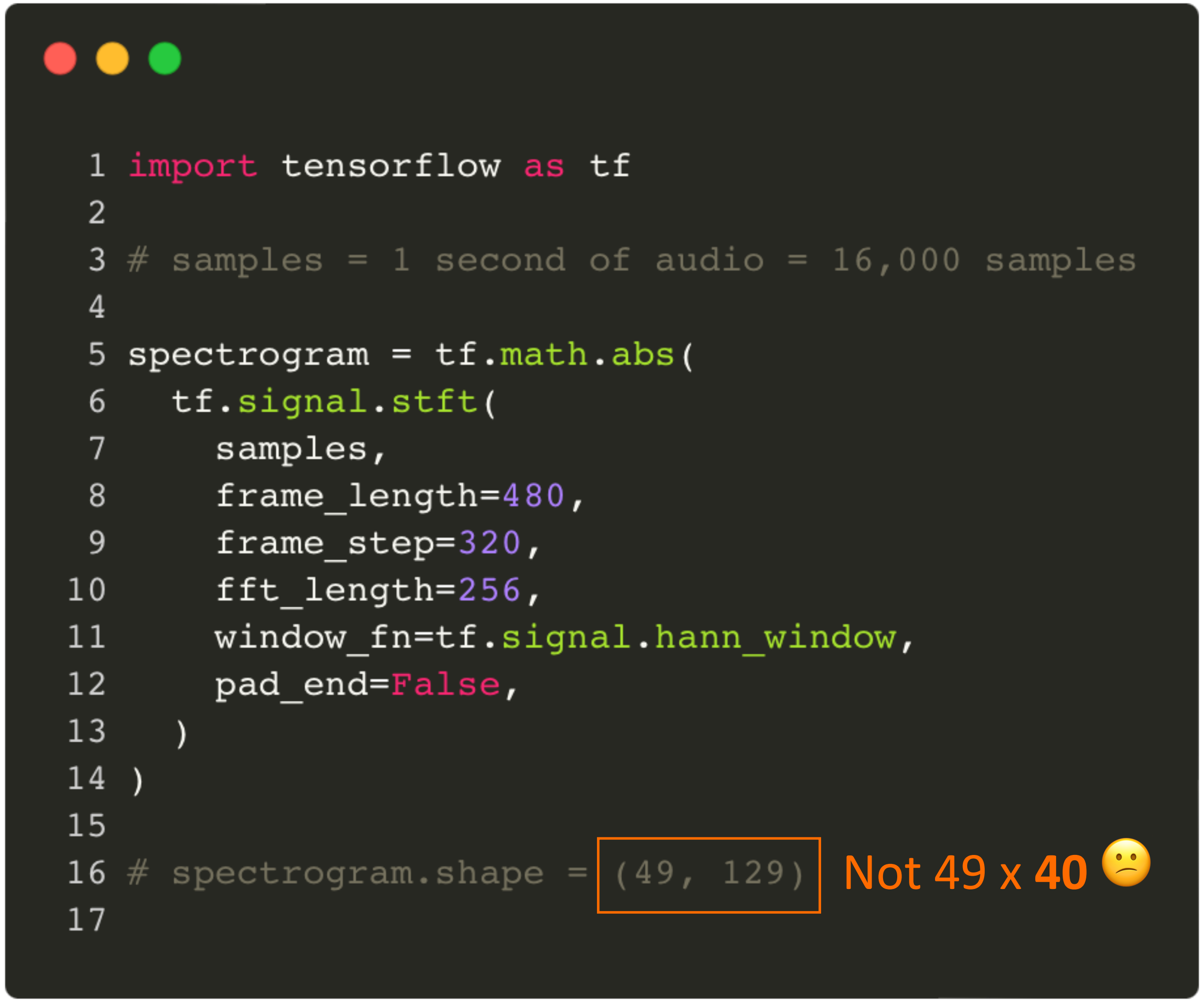
$$\frac{30}{1000} \times 16000 = 480 \text{ samples}$$

Frame Step = 20 ms

$$\frac{20}{1000} \times 16000 = 320 \text{ samples}$$

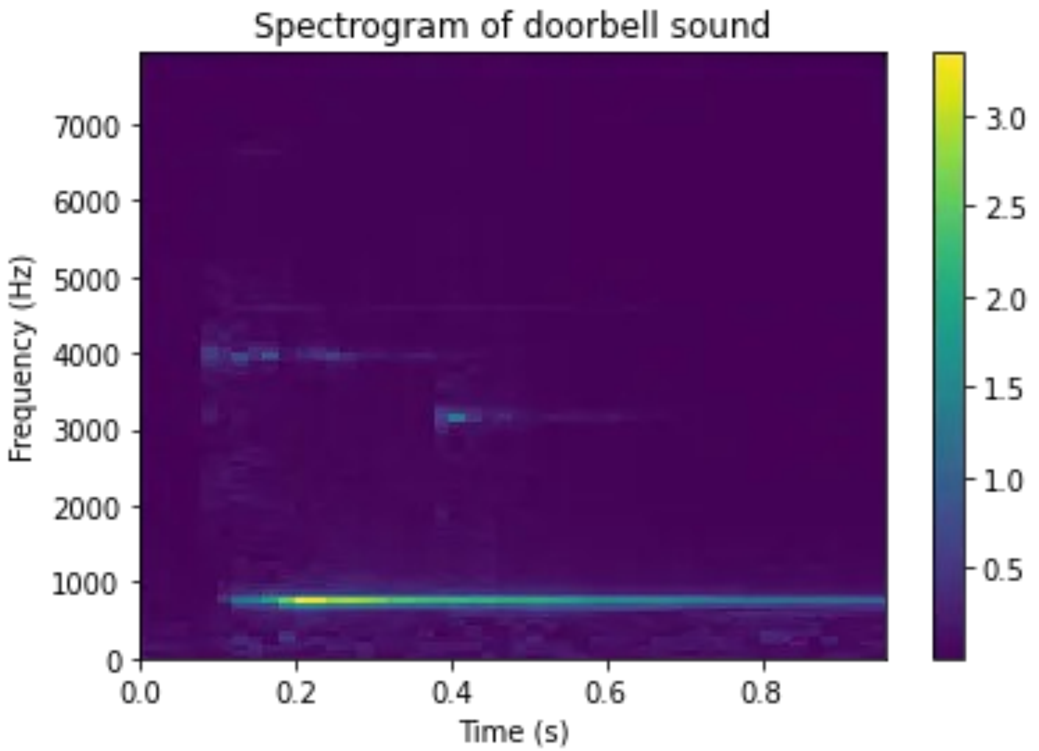
FFT Size = 256

tf.signal - spectrogram



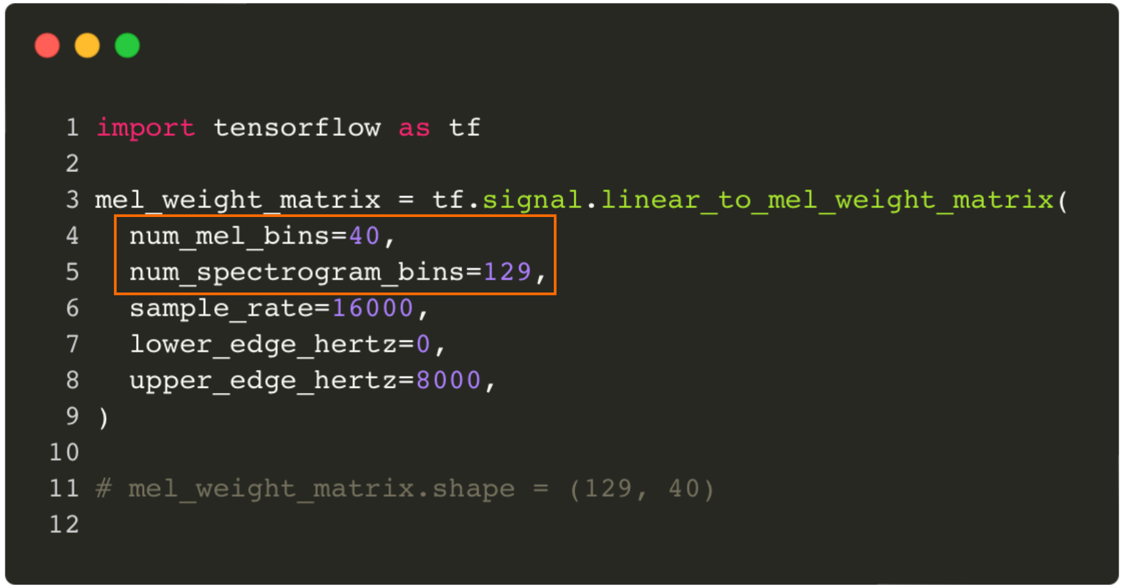
```
import tensorflow as tf

# samples = 1 second of audio = 16,000 samples

spectrogram = tf.math.abs(
  tf.signal.stft(
    samples,
    frame_length=480,
    frame_step=320,
    fft_length=256,
    window_fn=tf.signal.hann_window,
    pad_end=False,
  )
)

# spectrogram.shape = (49, 129)

```

Not 49 x **40** 😕

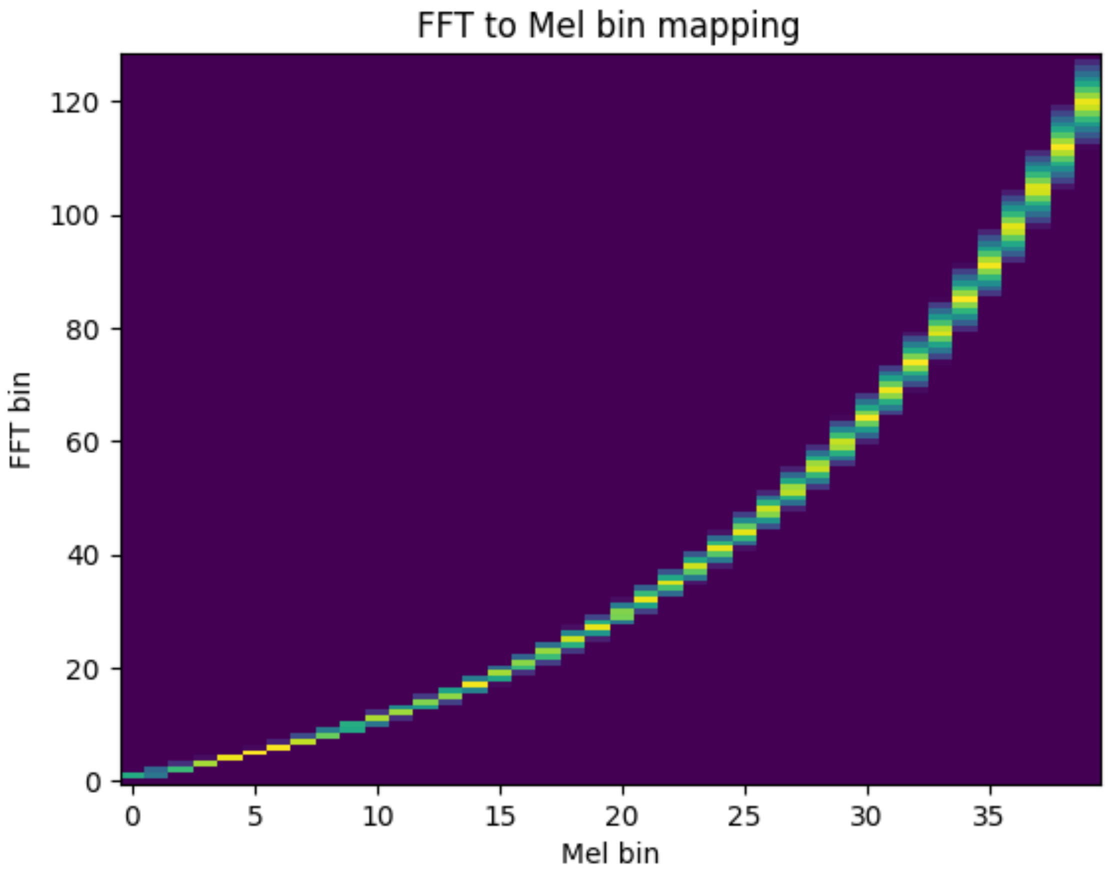


tf.signal - Mel Weight Matrix

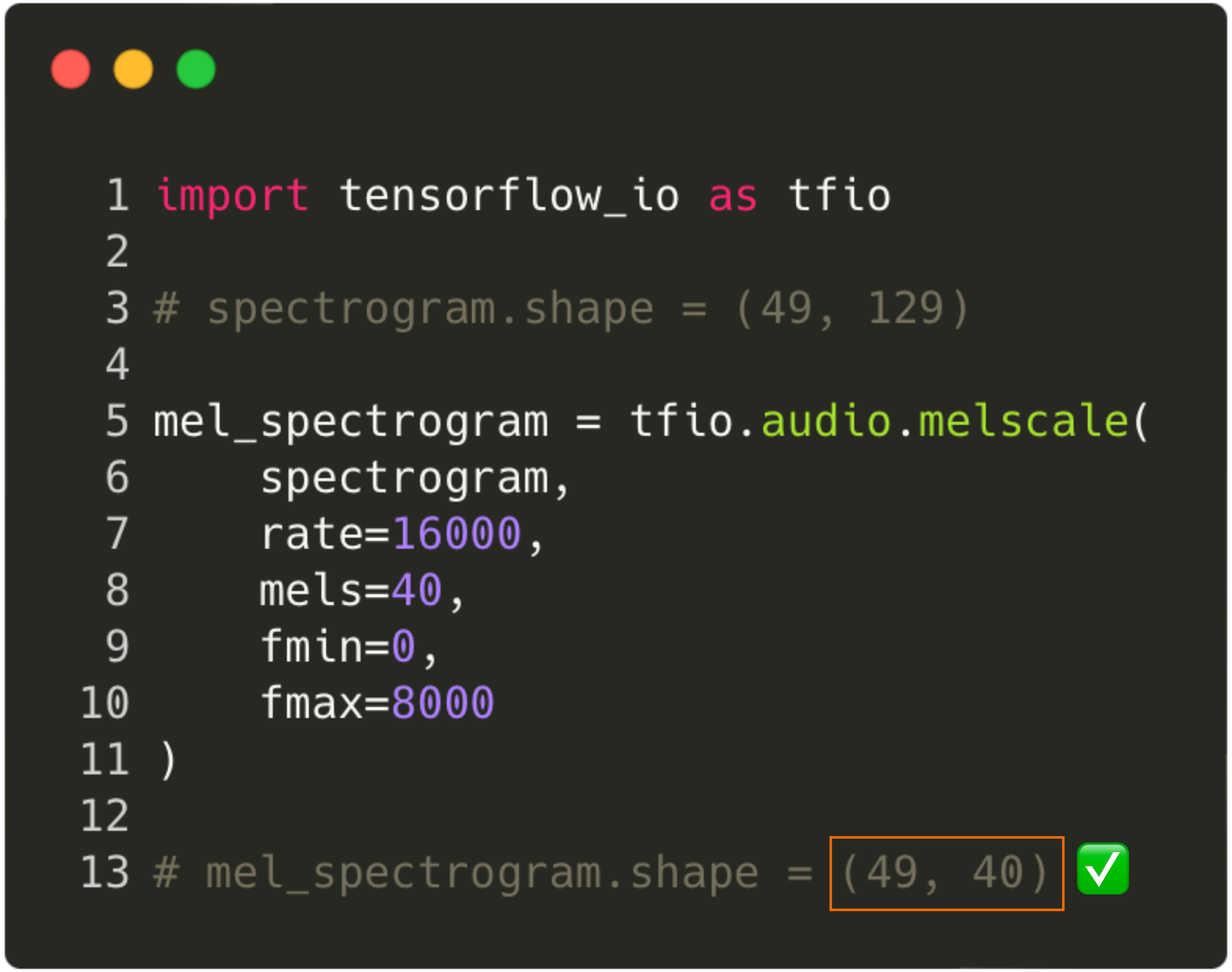
Human perception of audio frequencies



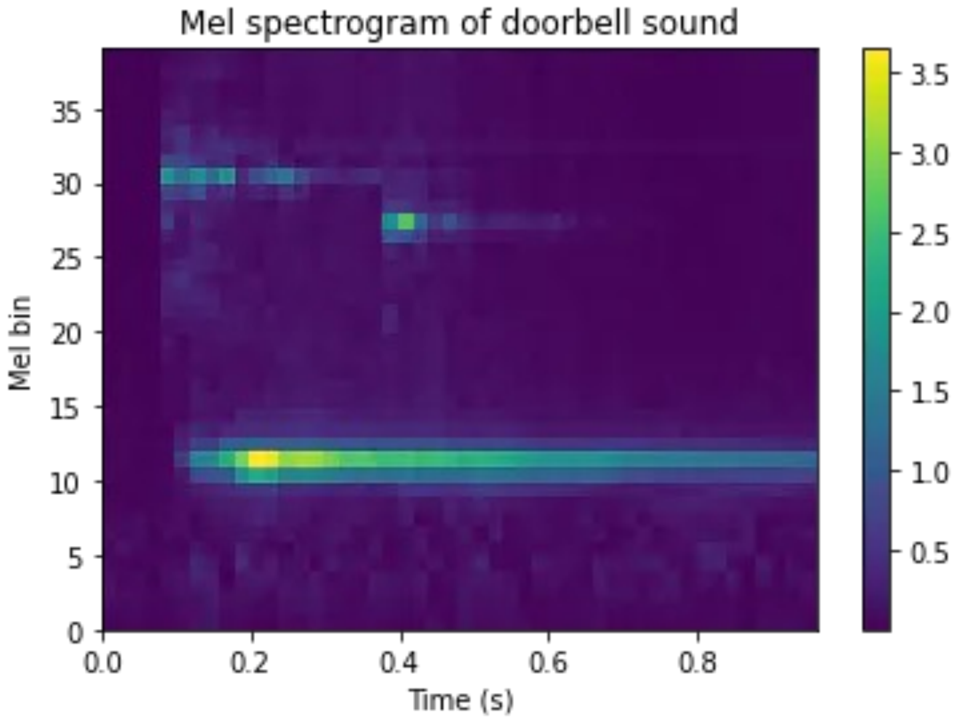
```
import tensorflow as tf

mel_weight_matrix = tf.signal.linear_to_mel_weight_matrix(
  num_mel_bins=40,
  num_spectrogram_bins=129,
  sample_rate=16000,
  lower_edge_hertz=0,
  upper_edge_hertz=8000,
)

# mel_weight_matrix.shape = (129, 40)

```



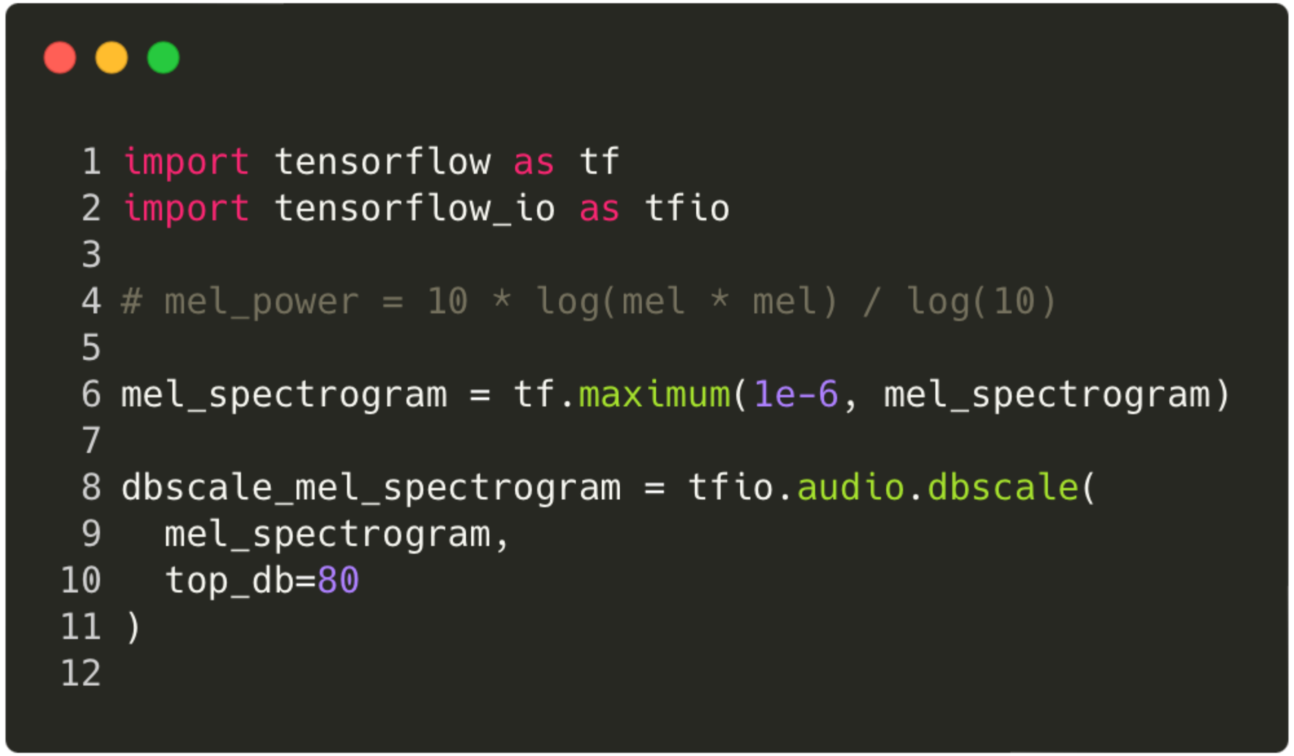
tf.io - Mel spectrogram



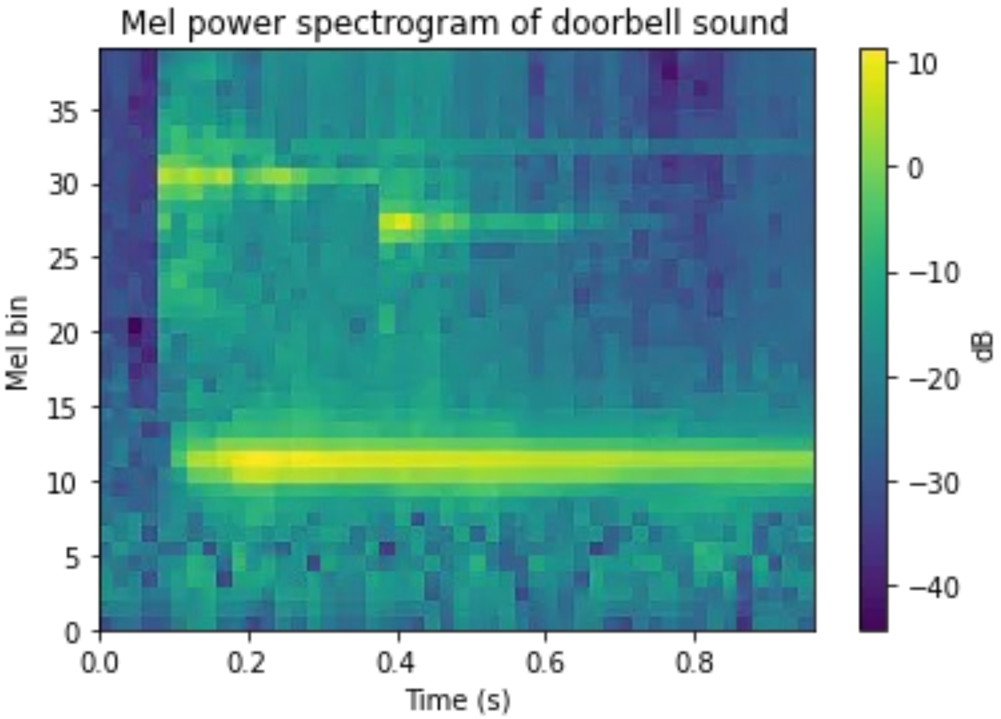
```python
import tensorflow_io as tfio

# spectrogram.shape = (49, 129)

mel_spectrogram = tfio.audio.melscale(
      spectrogram,
      rate=16000,
      mels=40,
      fmin=0,
      fmax=8000
)

# mel_spectrogram.shape = (49, 40) ✅
```



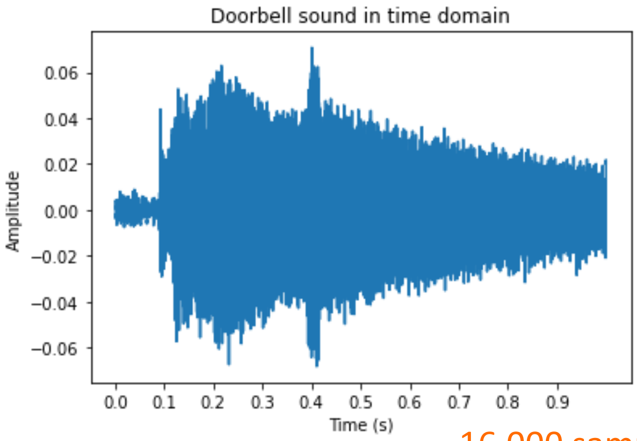
tf.io - Mel power spectrogram (dB)



```
import tensorflow as tf
import tensorflow_io as tfio

# mel_power = 10 * log(mel * mel) / log(10)

mel_spectrogram = tf.maximum(1e-6, mel_spectrogram)

dbscale_mel_spectrogram = tfio.audio.dbscale(
  mel_spectrogram,
  top_db=80
)

```

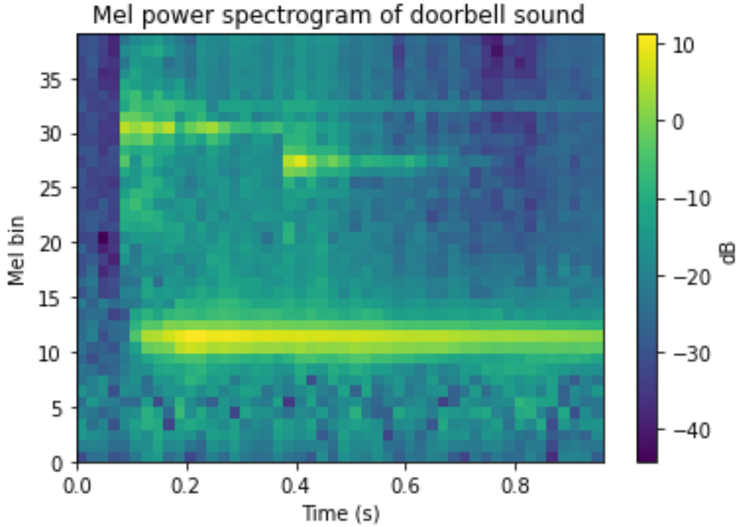


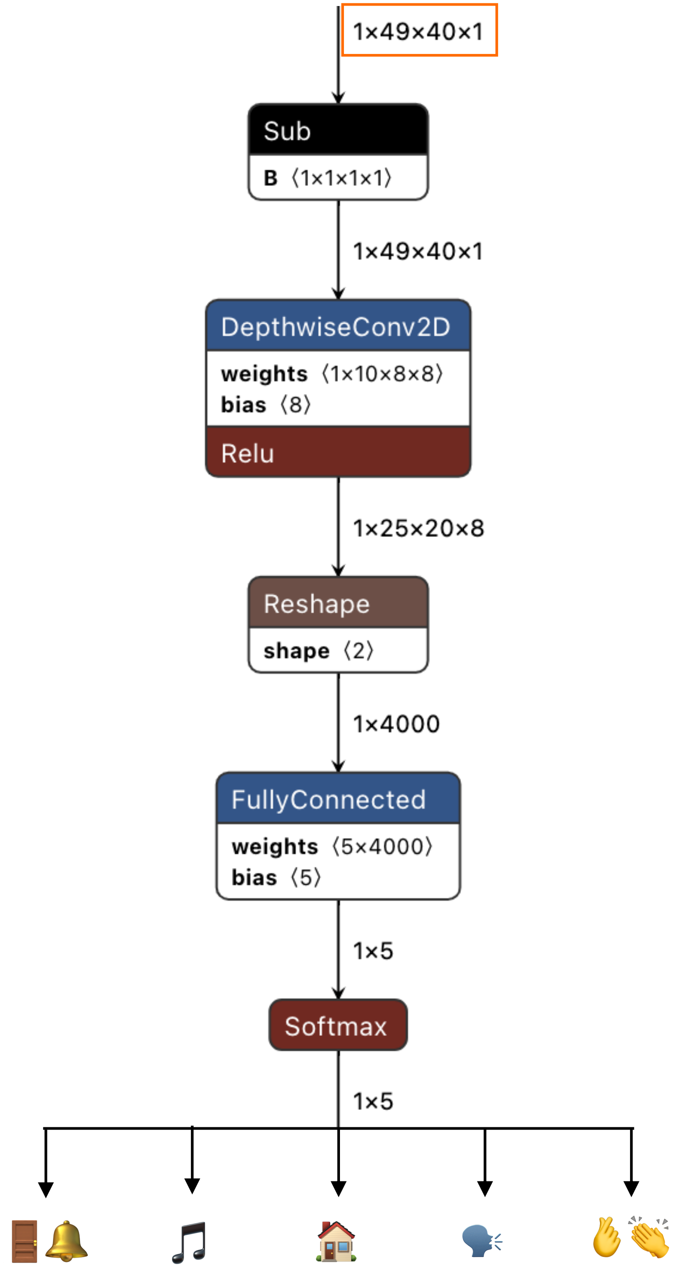
DSP + ML Model



16,000 samples

49 x 40 “2d image”





1×49×40×1

Sub

B ⟨1×1×1×1⟩

1×49×40×1

DepthwiseConv2D

weights ⟨1×10×8×8⟩
bias ⟨8⟩

Relu

1×25×20×8

Reshape

shape ⟨2⟩

1×4000

FullyConnected

weights ⟨5×4000⟩
bias ⟨5⟩

1×5

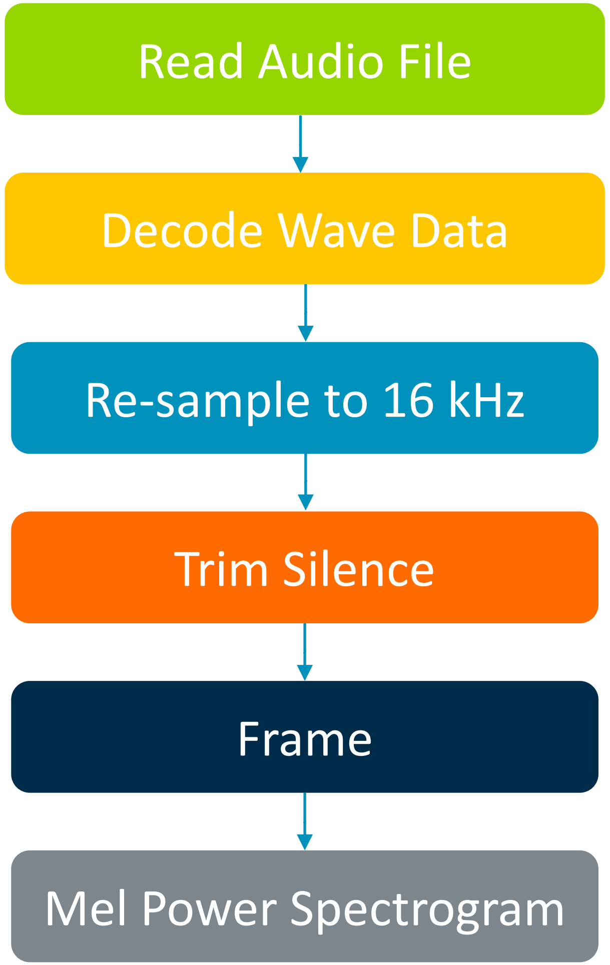
Softmax

1×5

🚪🔔 🎵 🏠 🗣️ 🤞👏

Model Training Flow

`tf.data.Dataset` pipeline



Read Audio File

Decode Wave Data

Re-sample to 16 kHz

Trim Silence

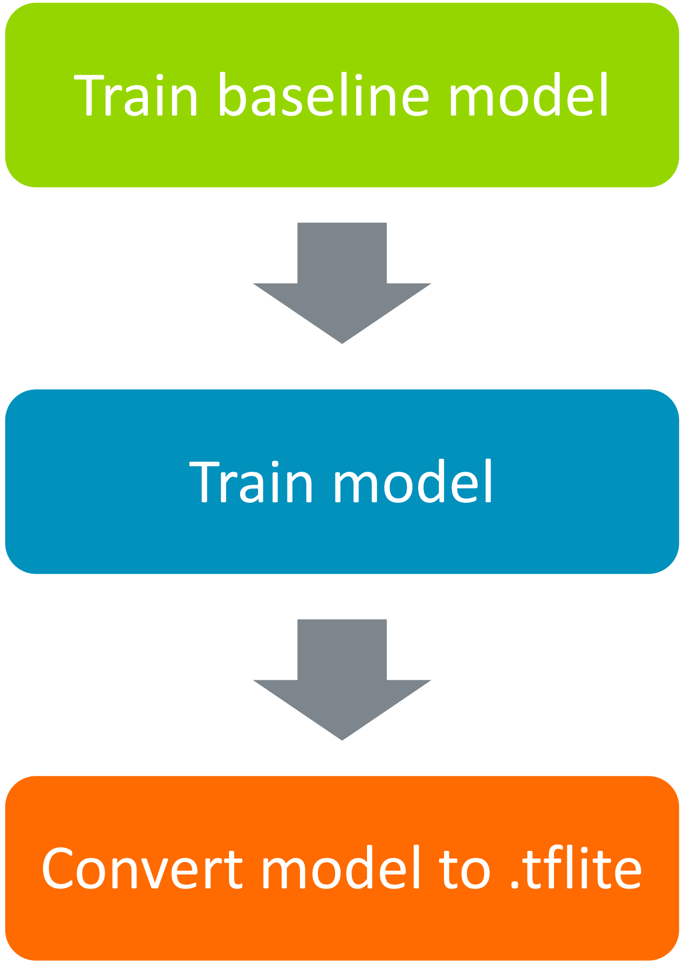
Frame

Mel Power Spectrogram

```
tf.io.read_file(…)

tf.audio.decode_wav(…)

tfio.audio.resample(…)

tfio.audio.trim(…)

# create 1 s slices with 0.1 s of overlap
tf.signal.frame(…)

# steps from previous slides
```

Model Training Flow

Train ”tiny_conv” model and convert



Train baseline model

Using ESC-50K data, 50 classes

Train model

Using subset of FSD50K data and Transfer Learning

--> same `DepthWiseConv2D` weights as baseline model

--> new classification head, 5 classes

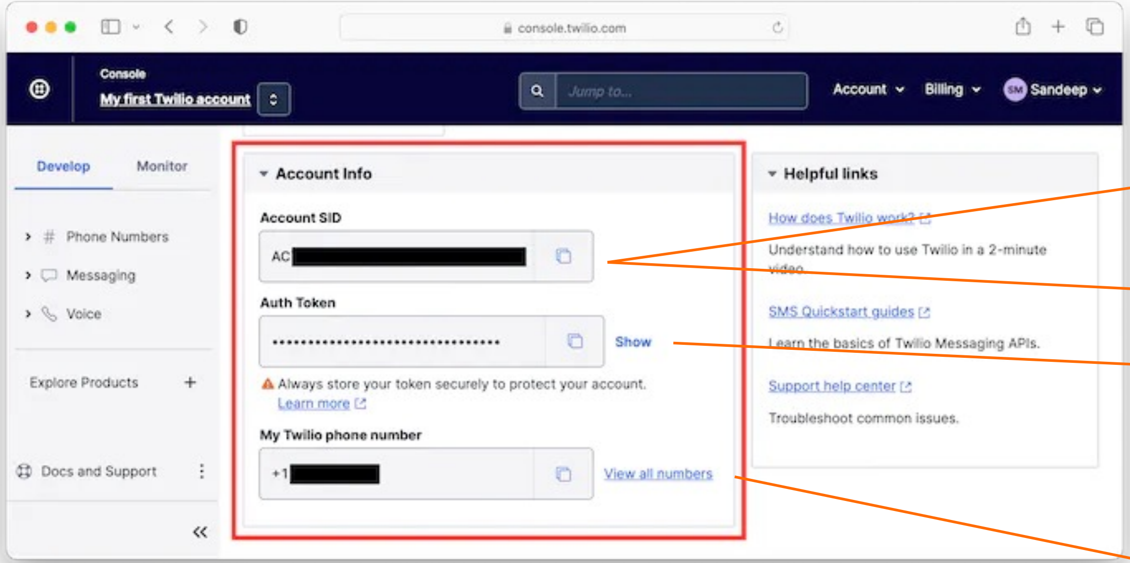
Convert model to .tflite

8-bit inputs and outputs, quantized weights

Sending SMS messages 💬

Twilio REST API

Twilio Console



Account Info

Account SID

AC[redacted]

Auth Token

••••••••••••••••••••••••••••••••

⚠ Always store your token securely to protect your account.

My Twilio phone number

+1[redacted]

Understand how to use Twilio in a 2-minute video.

Learn the basics of Twilio Messaging APIs.

Troubleshoot common issues.

HTTP POST

- URL

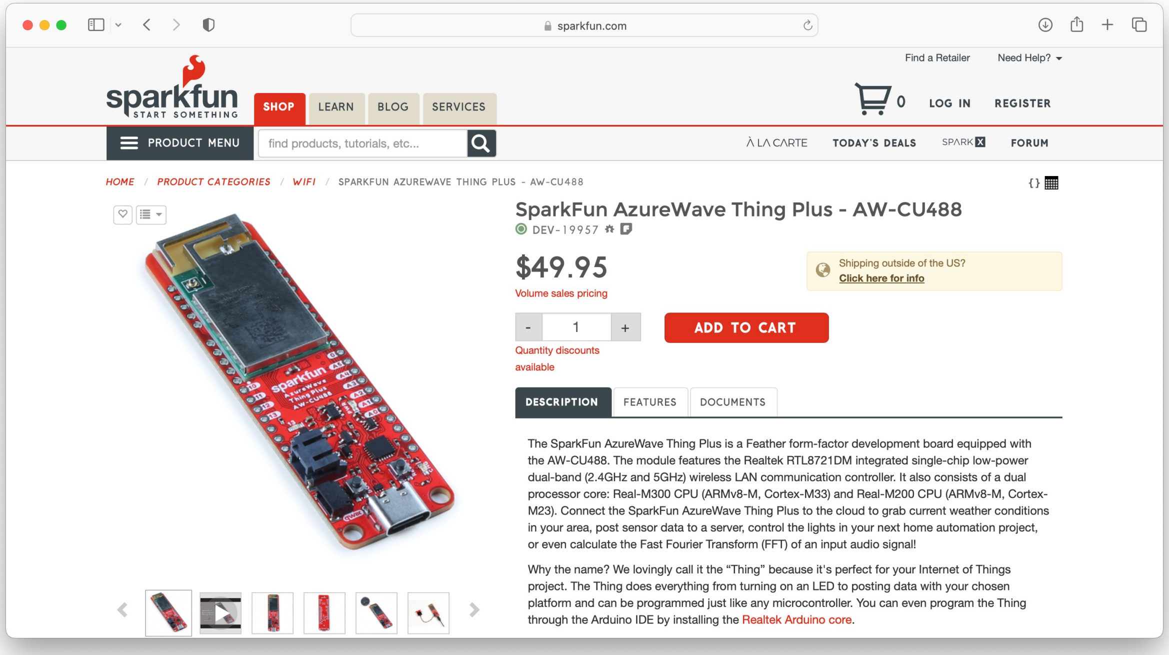
  `https://api.twilio.com/2010-04-01/Accounts/<Account SID>/Messages.json`

- Auth Header = HTTP Basic Auth
  - `Username` = Account SID
  - `Password` = Auth Token

- Body = `application/x-www-form-urlencoded`
  - `To` = the phone # to send the message to
  - `From` = the Twilio # the message is from
  - `Body` = the message text

SparkFun AzureWave ThingPlus

Realtek RTL8721DM SoC - compute, connectivity, and audio



SparkFun AzureWave Thing Plus - AW-CU488

DEV-19957

$49.95

The SparkFun AzureWave Thing Plus is a Feather form-factor development board equipped with the AW-CU488. The module features the Realtek RTL8721DM integrated single-chip low-power dual-band (2.4GHz and 5GHz) wireless LAN communication controller. It also consists of a dual processor core: Real-M300 CPU (ARMv8-M, Cortex-M33) and Real-M200 CPU (ARMv8-M, Cortex-M23). Connect the SparkFun AzureWave Thing Plus to the cloud to grab current weather conditions in your area, post sensor data to a server, control the lights in your next home automation project, or even calculate the Fast Fourier Transform (FFT) of an input audio signal!

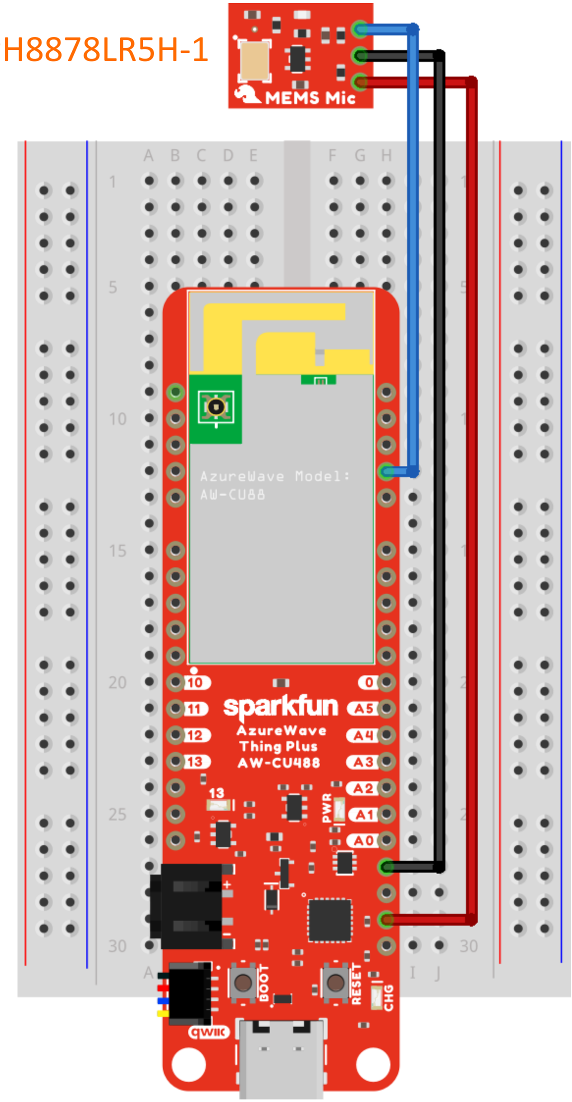
Why the name? We lovingly call it the “Thing” because it's perfect for your Internet of Things project. The Thing does everything from turning on an LED to posting data with your chosen platform and can be programmed just like any microcontroller. You can even program the Thing through the Arduino IDE by installing the Realtek Arduino core.

- Arm Cortex-M33 compatible Real-M300 CPU @ 200 MHz
- 512 KB of SRAM and 4 MB of PSRAM
- 4 MB of flash
- Built-in 2.4 GHz and 5 GHz Wi-Fi connectivity
- Built-in audio codec with two analog inputs

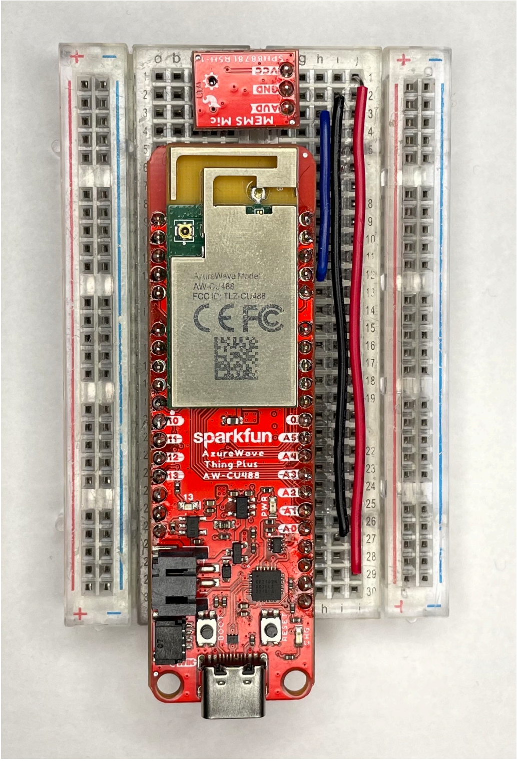
Arduino IDE support with the Realtek Arduino core

MEMS Microphone Input

SparkFun SPH8878LR5H-1



SparkFun AzureWave Thing+	Analog MEMS Microphone
3V3	VCC
GND	GND
22 (PA4)	AUD

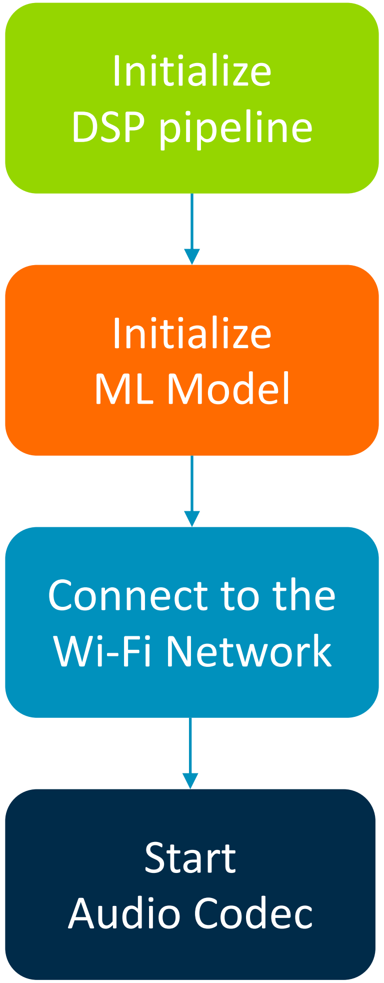


📚 Arduino Libraries

Name	Provider	Description
AudioCodec	Realtek	Used to control and manage the hardware Audio Codec • Provides audio data 1024 bytes at a time = 512 samples @ 16-bit • With sample rate = 16,000 Hz have **32 ms** to process audio in real-time • 32 ms = 0.320 s = 512 samples / 16,000 samples per second
WiFi	Realtek	Used to control and manage the Wi-Fi interface and UDP or TCP sockets
Ameba_TensorFlowLite	Realtek	Provides TensorFlow Lite for Microcontroller (TFLM) support for the board • Includes Arm's **CMSIS-NN** library, which provides optimized Neural Network compute kernels for Arm Cortex-M processors
CMSIS-DSP	Arm	Optimized Digital Signal Processing on Arm Cortex-M
ArduinoHttpClient	Arduino	Used to interact with HTTP + REST API’s

Arduino Sketch pseudo code

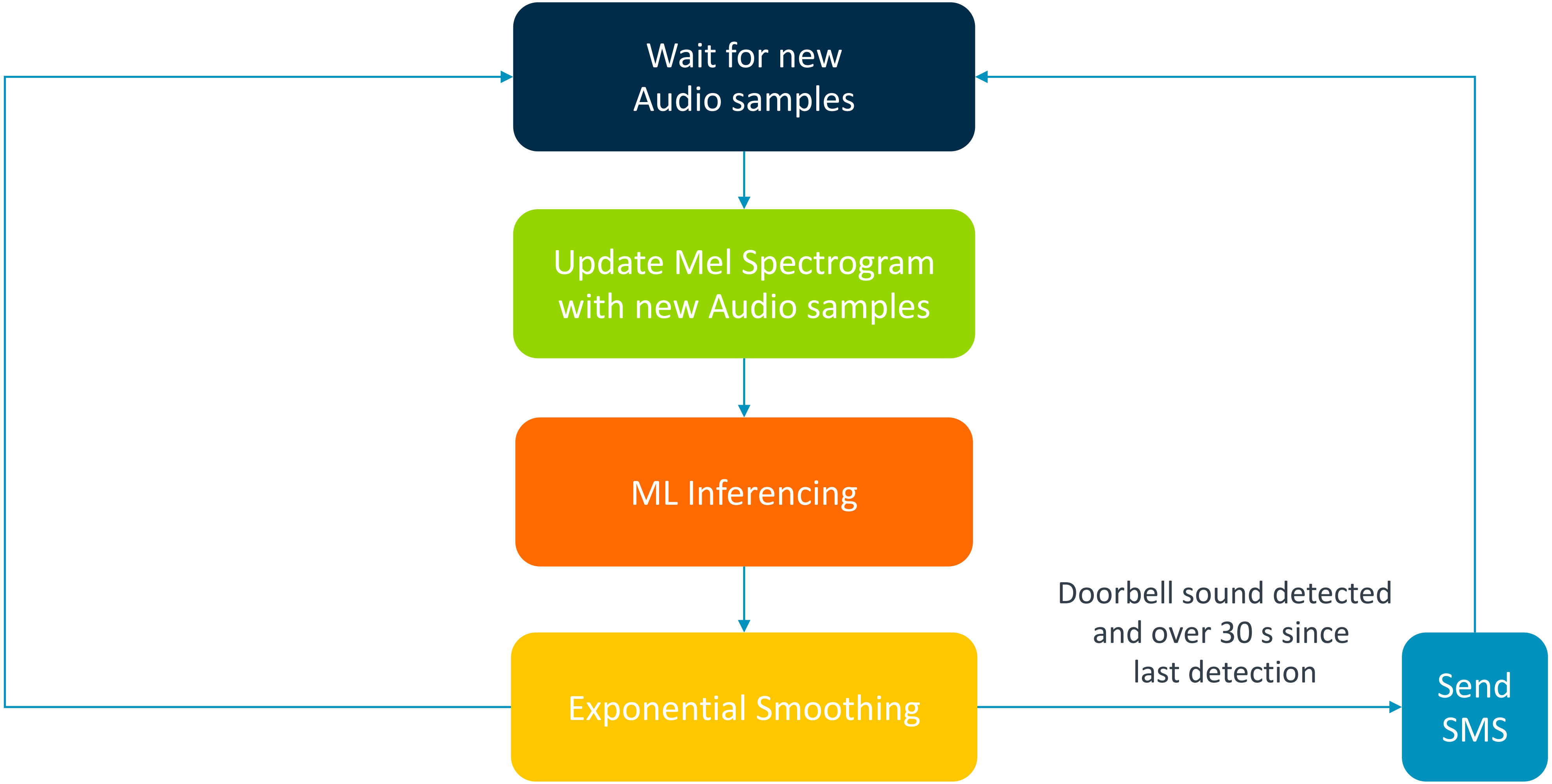
`setup()`



Mono 16-bit @ 16 kHz

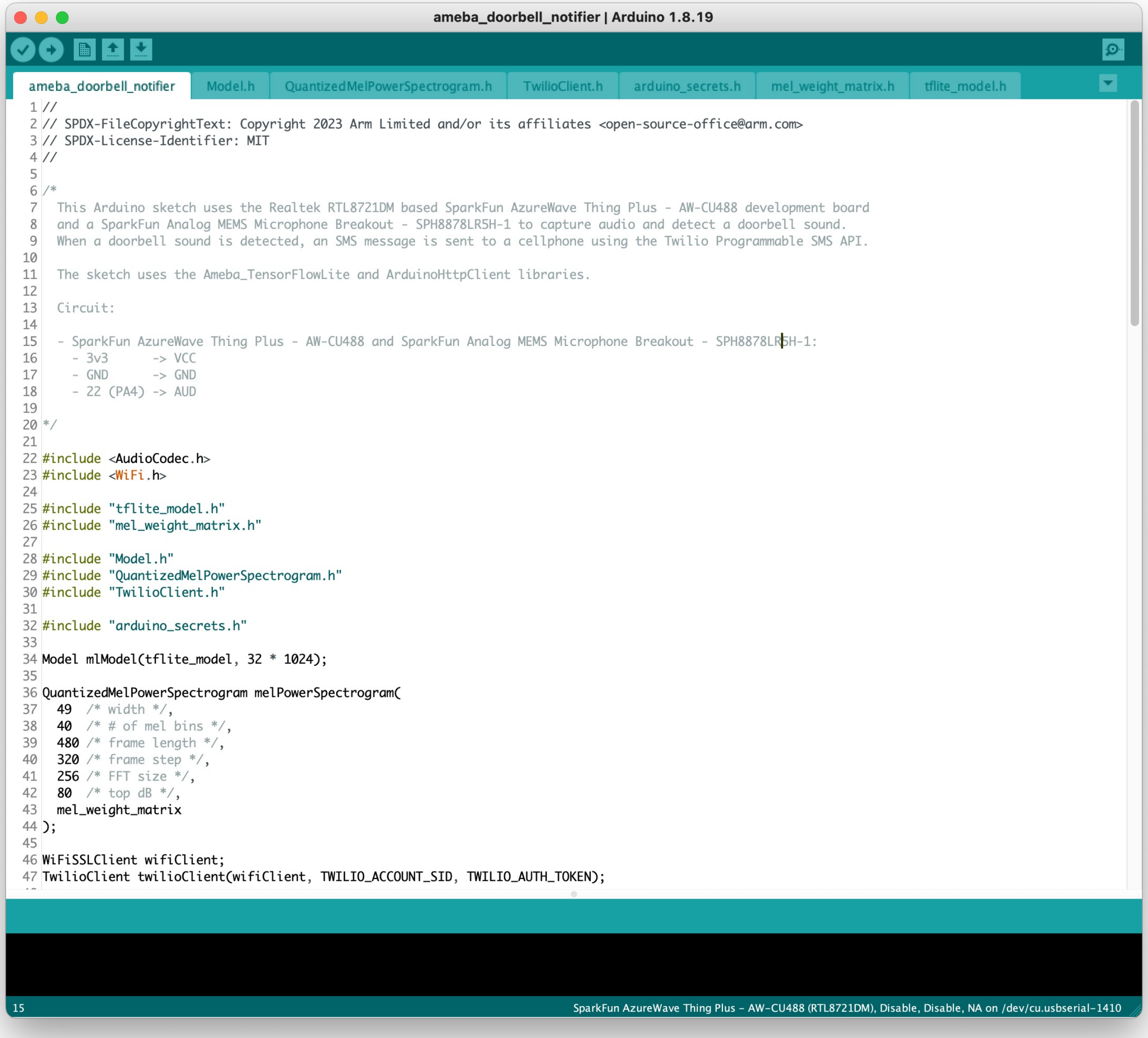
Arduino Sketch pseudo code

`loop()`


Wait for new Audio samples
Update Mel Spectrogram with new Audio samples
ML Inferencing
Exponential Smoothing
Doorbell sound detected and over 30 s since last detection
Send SMS

$$s_t = \alpha x_t + (1 - \alpha) s_{t-1}$$

Arduino Sketch



ameba_doorbell_notifier | Arduino 1.8.19
ameba_doorbell_notifier
Model.h
QuantizedMelPowerSpectrogram.h
TwilioClient.h
arduino_secrets.h
mel_weight_matrix.h
tflite_model.h
15
SparkFun AzureWave Thing Plus - AW-CU488 (RTL8721DM), Disable, Disable, NA on /dev/cu.usbserial-1410

```
//
// SPDX-FileCopyrightText: Copyright 2023 Arm Limited and/or its affiliates <open-source-office@arm.com>
// SPDX-License-Identifier: MIT
//

/*
  This Arduino sketch uses the Realtek RTL8721DM based SparkFun AzureWave Thing Plus - AW-CU488 development board
  and a SparkFun Analog MEMS Microphone Breakout - SPH8878LR5H-1 to capture audio and detect a doorbell sound.
  When a doorbell sound is detected, an SMS message is sent to a cellphone using the Twilio Programmable SMS API.

  The sketch uses the Ameba_TensorFlowLite and ArduinoHttpClient libraries.

  Circuit:

  - SparkFun AzureWave Thing Plus - AW-CU488 and SparkFun Analog MEMS Microphone Breakout - SPH8878LR5H-1:
    - 3v3      -> VCC
    - GND      -> GND
    - 22 (PA4) -> AUD

*/

#include <AudioCodec.h>
#include <WiFi.h>

#include "tflite_model.h"
#include "mel_weight_matrix.h"

#include "Model.h"
#include "QuantizedMelPowerSpectrogram.h"
#include "TwilioClient.h"

#include "arduino_secrets.h"

Model mlModel(tflite_model, 32 * 1024);

QuantizedMelPowerSpectrogram melPowerSpectrogram(
  49  /* width */,
  40  /* # of mel bins */,
  480 /* frame length */,
  320 /* frame step */,
  256 /* FFT size */,
  80  /* top dB */,
  mel_weight_matrix
);

WiFiSSLClient wifiClient;
TwilioClient twilioClient(wifiClient, TWILIO_ACCOUNT_SID, TWILIO_AUTH_TOKEN);
```

Processing Step	Time
Mel Power Spectrogram	~3.4 ms
Model Inferencing	~14.0 ms
Total	**~17.4 ms**

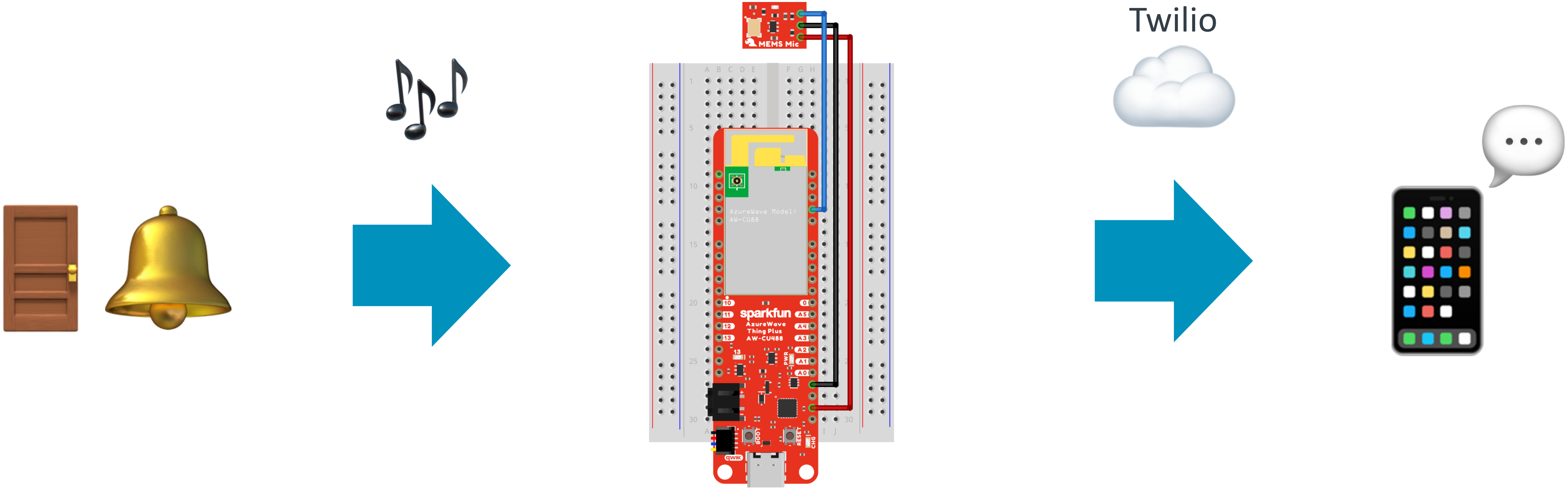
Under the 32 ms goal for real-time processing !

Recap

Audio

Compute

Connectivity



Twilio

Learn More …

- Demo at Arm booth

- Hackster.io
  - https://www.hackster.io/sandeep-mistry/how-to-build-an-ml-powered-doorbell-notifier-0a781e

- GitHub
  - https://github.com/ArmDeveloperEcosystem/aiot-doorbell-notifier-example-for-ameba

Thank You

Danke

Gracias

Grazie

谢谢

ありがとう

Asante

Merci

감사합니다

धन्यवाद

Kiitos

شكرًا

ধন্যবাদ

תודה

Copyright Notice

TINY ML EMEA
AMSTERDAM, NL
June 26-28,2023

www.tinyml.org