

Real-Time Keyword Recognition on STM32 using TensorFlow Lite

Stefan Gloor

ETH Zürich

June 12, 2024

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Contents

- Project goals
- Hardware
- Dataset
- Model and framework
- Microcontroller Implementation
- Performance evaluation
- Limitations
- Demo

Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Motivation

- Simple speech recognition
 - Wide range of applications
 - Simple and cheap user interface
 - Hands-free operation of devices
- Why on the edge?
 - Very cheap
 - Low latency
 - Data privacy



Goals

- Achieve simple speech recognition completely “on the edge”, detect keywords such “left” and “right”.
- Familiarize with available frameworks
- Demo project as a base for future applications



Goals

- Achieve simple speech recognition completely “on the edge”, detect keywords such “left” and “right”.
- Familiarize with available frameworks
- Demo project as a base for future applications



Goals

- Achieve simple speech recognition completely “on the edge”, detect keywords such “left” and “right”.
- Familiarize with available frameworks
- Demo project as a base for future applications



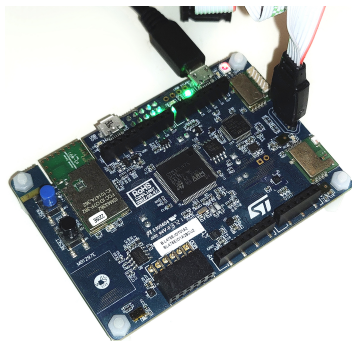
Goals

- Achieve simple speech recognition completely “on the edge”, detect keywords such “left” and “right”.
- Familiarize with available frameworks
- Demo project as a base for future applications



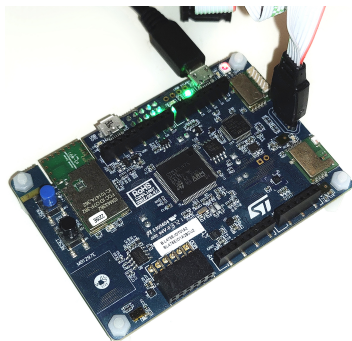
Hardware

- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
- 1 MB Flash
- 128 kB SRAM



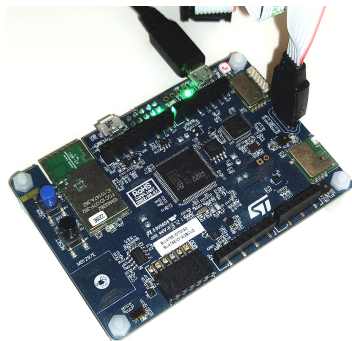
Hardware

- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
- 1 MB Flash
- 128 kB SRAM



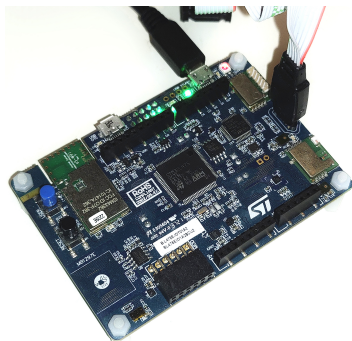
Hardware

- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
- 1 MB Flash
- 128 kB SRAM



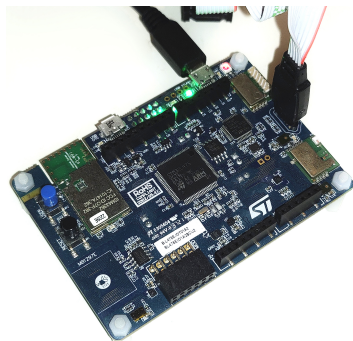
Hardware

- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
- 1 MB Flash
- 128 kB SRAM



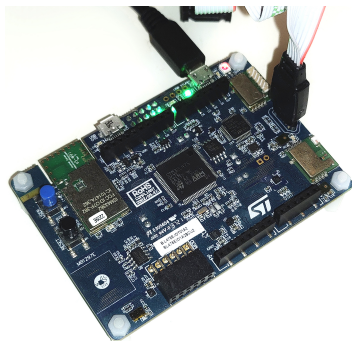
Hardware

- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
- 1 MB Flash
- 128 kB SRAM



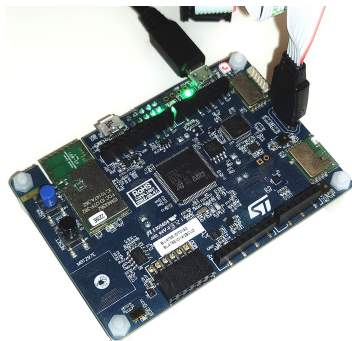
Hardware

- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
 - 1 MB Flash
 - 128 kB SRAM



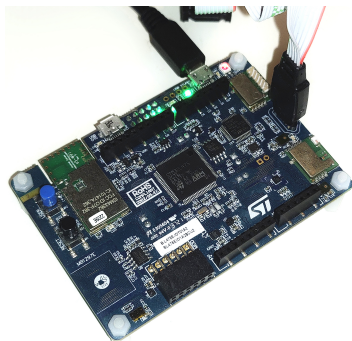
Hardware

- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
- 1 MB Flash
- 128 kB SRAM



Hardware

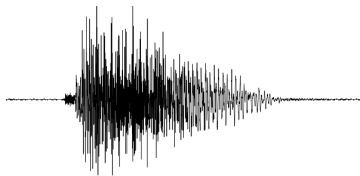
- STM32L475VGT
 - General-purpose MCU
 - Industry-proven
 - Good availability
- Arm Cortex M4F @ 80 MHz
- 1 MB Flash
- 128 kB SRAM



Dataset

speech_commands¹ by P. Warden at Google.

- Spoken keywords like “yes”, “no”, “up”, “down”, ...
- 1 second clips
- 16 kHz sampling rate
- 6:1:1 split

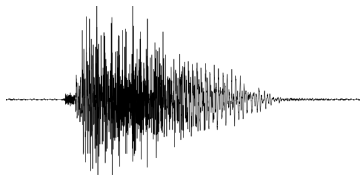


¹https://huggingface.co/datasets/google/speech_commands

Dataset

speech_commands¹ by P. Warden at Google.

- Spoken keywords like “yes”, “no”, “up”, “down”, ...
- 1 second clips
- 16 kHz sampling rate
- 6:1:1 split

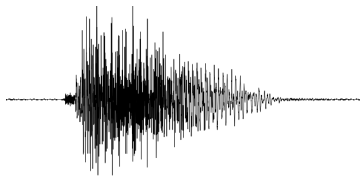


¹https://huggingface.co/datasets/google/speech_commands

Dataset

speech_commands¹ by P. Warden at Google.

- Spoken keywords like “yes”, “no”, “up”, “down”, ...
- 1 second clips
- 16 kHz sampling rate
- 6:1:1 split

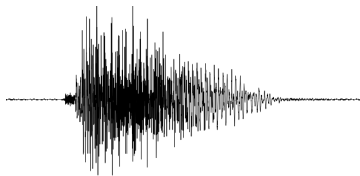


¹https://huggingface.co/datasets/google/speech_commands

Dataset

speech_commands¹ by P. Warden at Google.

- Spoken keywords like “yes”, “no”, “up”, “down”, ...
- 1 second clips
- 16 kHz sampling rate
- 6:1:1 split

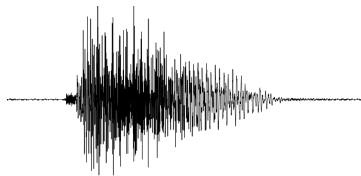


¹https://huggingface.co/datasets/google/speech_commands

Dataset

speech_commands¹ by P. Warden at Google.

- Spoken keywords like “yes”, “no”, “up”, “down”, ...
- 1 second clips
- 16 kHz sampling rate
- 6:1:1 split



¹https://huggingface.co/datasets/google/speech_commands

Framework

TensorFlow Lite for Microcontrollers

- Easy to use
- Fully customizable
- Platform-independent



TensorFlow

Framework

TensorFlow Lite for Microcontrollers

- Easy to use
- Fully customizable
- Platform-independent



TensorFlow

Framework

TensorFlow Lite for Microcontrollers

- Easy to use
- Fully customizable
- Platform-independent



TensorFlow

Framework

TensorFlow Lite for Microcontrollers

- Easy to use
- Fully customizable
- Platform-independent

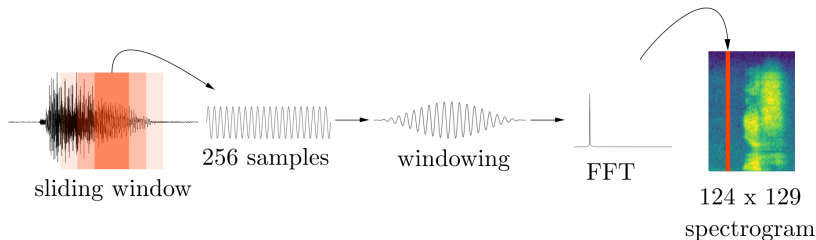


TensorFlow

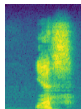
Preprocessing

Short-time fourier transform:

$$\mathbf{STFT}\{x[n]\}(m, \omega) \equiv X(m, \omega) = \sum_{n=-\infty}^{\infty} x[n]w[n - m]e^{-i\omega n}$$

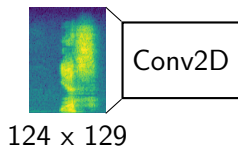


Model

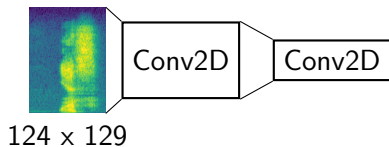


124 × 129

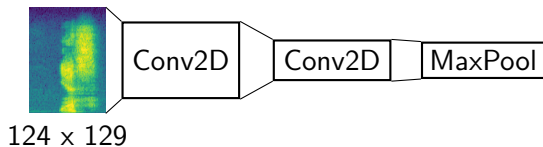
Model



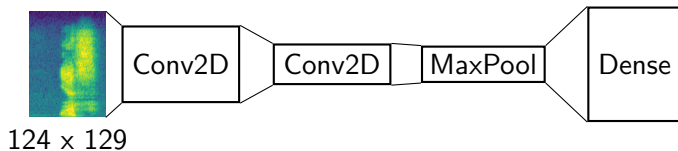
Model



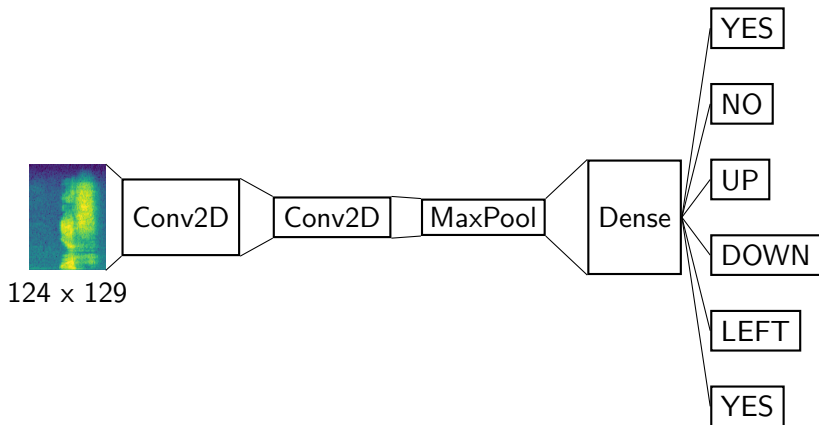
Model



Model



Model



Microcontroller Implementation

- Full integer quantization of the model
- Implemented UART protocol with CRC32 checksum
- Preprocessing using CMSIS-DSP

Microcontroller Implementation

- Full integer quantization of the model
- Implemented UART protocol with CRC32 checksum
- Preprocessing using CMSIS-DSP

Microcontroller Implementation

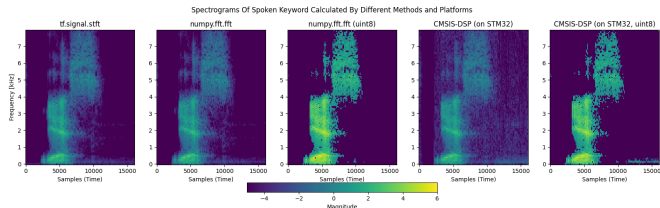
- Full integer quantization of the model
- Implemented UART protocol with CRC32 checksum
- Preprocessing using CMSIS-DSP

Microcontroller Implementation

- Full integer quantization of the model
- Implemented UART protocol with CRC32 checksum
- Preprocessing using CMSIS-DSP

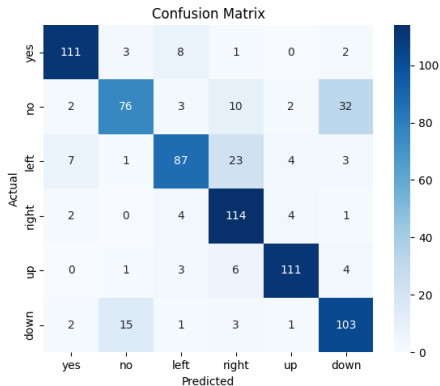
Microcontroller Implementation

- Full integer quantization of the model
- Implemented UART protocol with CRC32 checksum
- Preprocessing using CMSIS-DSP



Performance Evaluation

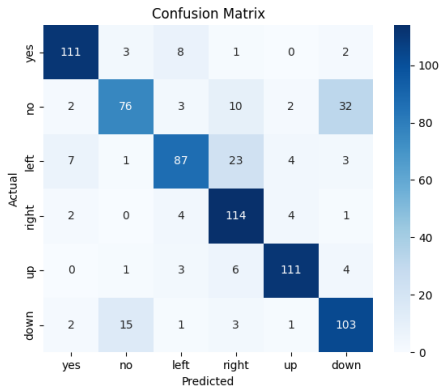
- 80.27 % overall accuracy
- 180 ms per inference



Performance Evaluation

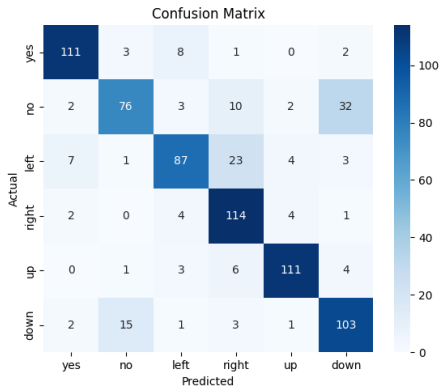
■ 80.27 % overall accuracy

■ 180 ms per inference



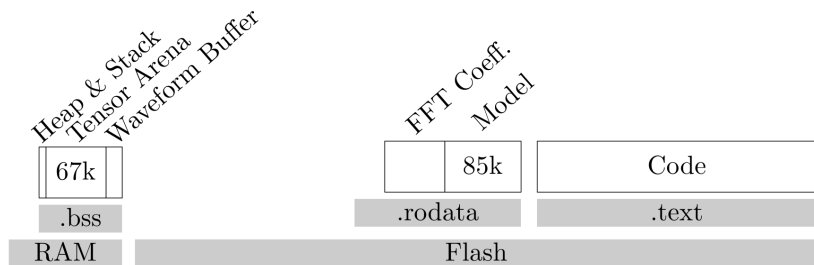
Performance Evaluation

- 80.27 % overall accuracy
- 180 ms per inference



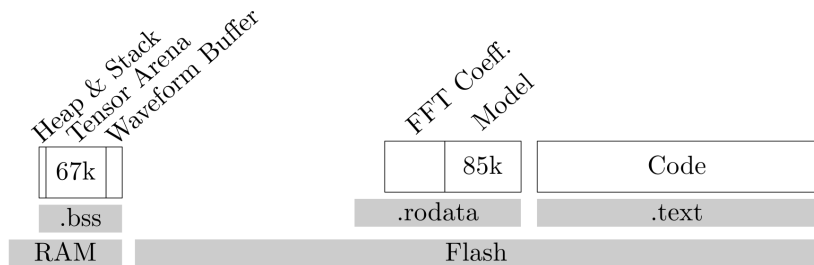
Memory and Storage Footprint

- Limiting factor is RAM for tensor arena
- 74.5 % SRAM used
- 45.9 % Flash used



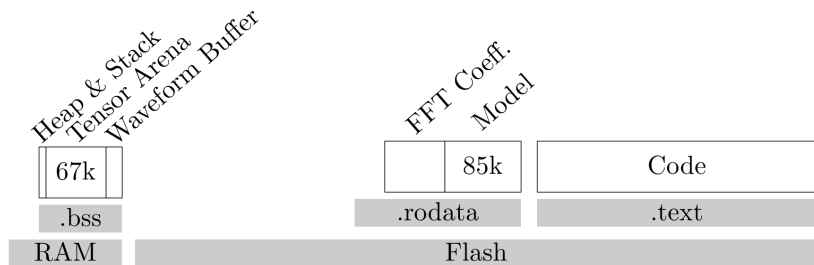
Memory and Storage Footprint

- Limiting factor is RAM for tensor arena
- 74.5 % SRAM used
- 45.9 % Flash used



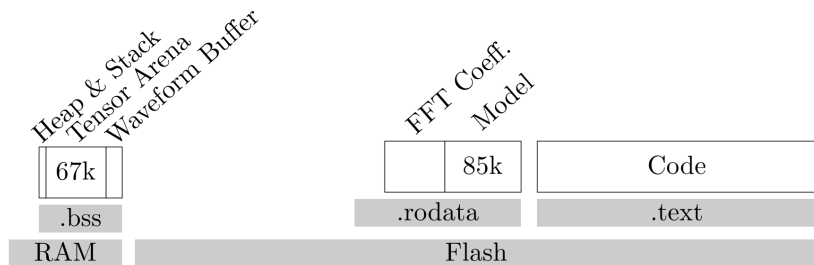
Memory and Storage Footprint

- Limiting factor is RAM for tensor arena
- 74.5 % SRAM used
- 45.9 % Flash used



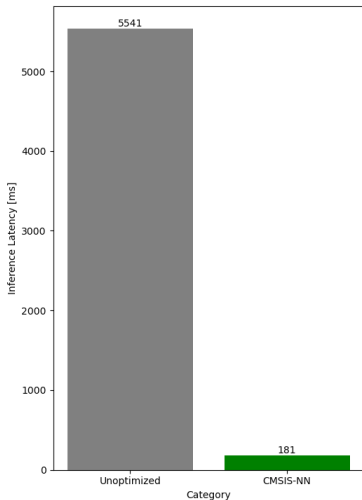
Memory and Storage Footprint

- Limiting factor is RAM for tensor arena
- 74.5 % SRAM used
- 45.9 % Flash used



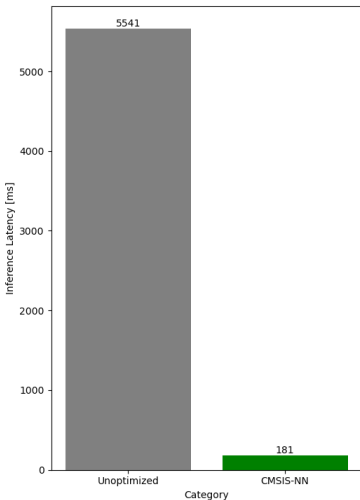
Optimized TFLite Kernel (using CMSIS-NN)

- Inference latency decreases by a factor of 30
- Program size increases by 37 kiB



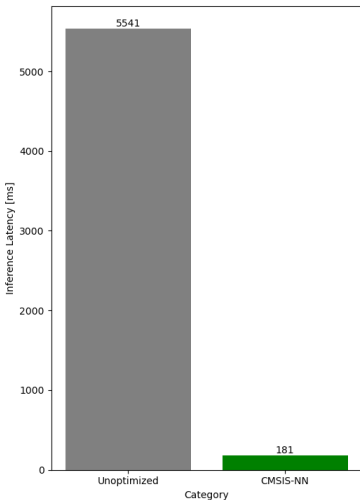
Optimized TFLite Kernel (using CMSIS-NN)

- Inference latency decreases by a factor of 30
- Program size increases by 37 kiB



Optimized TFLite Kernel (using CMSIS-NN)

- Inference latency decreases by a factor of 30
- Program size increases by 37 kiB



Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

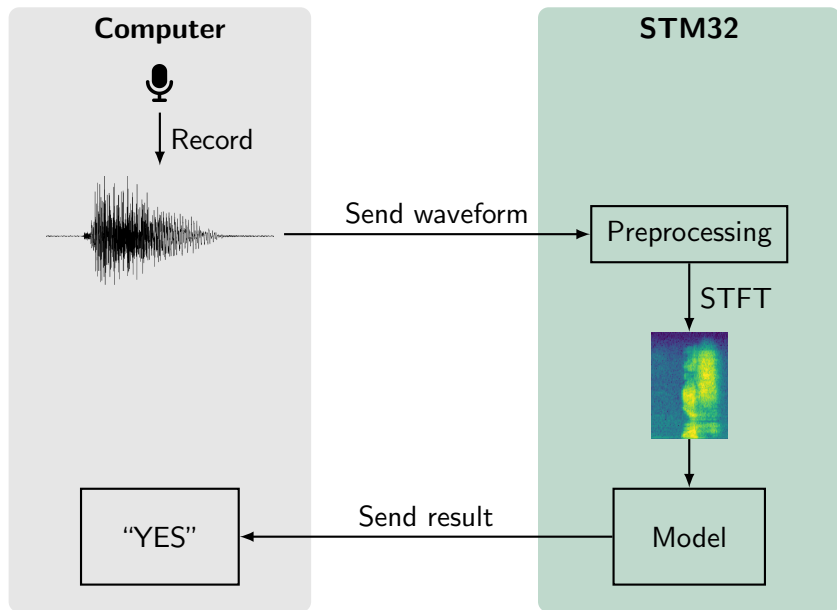
Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

Limitations

- Accuracy could be better
- “Spotting” of keywords in background noise
- Microphone acquisition not working
 - Started to read out on-board PDM microphone
 - No support for 8-bit output with DMA
 - Amplitude and sampling rate seem off
- Optimize code, refactoring

Demo



Real-Time Keyword Recognition on STM32 using TensorFlow Lite

Stefan Gloor

ETH Zürich

June 12, 2024