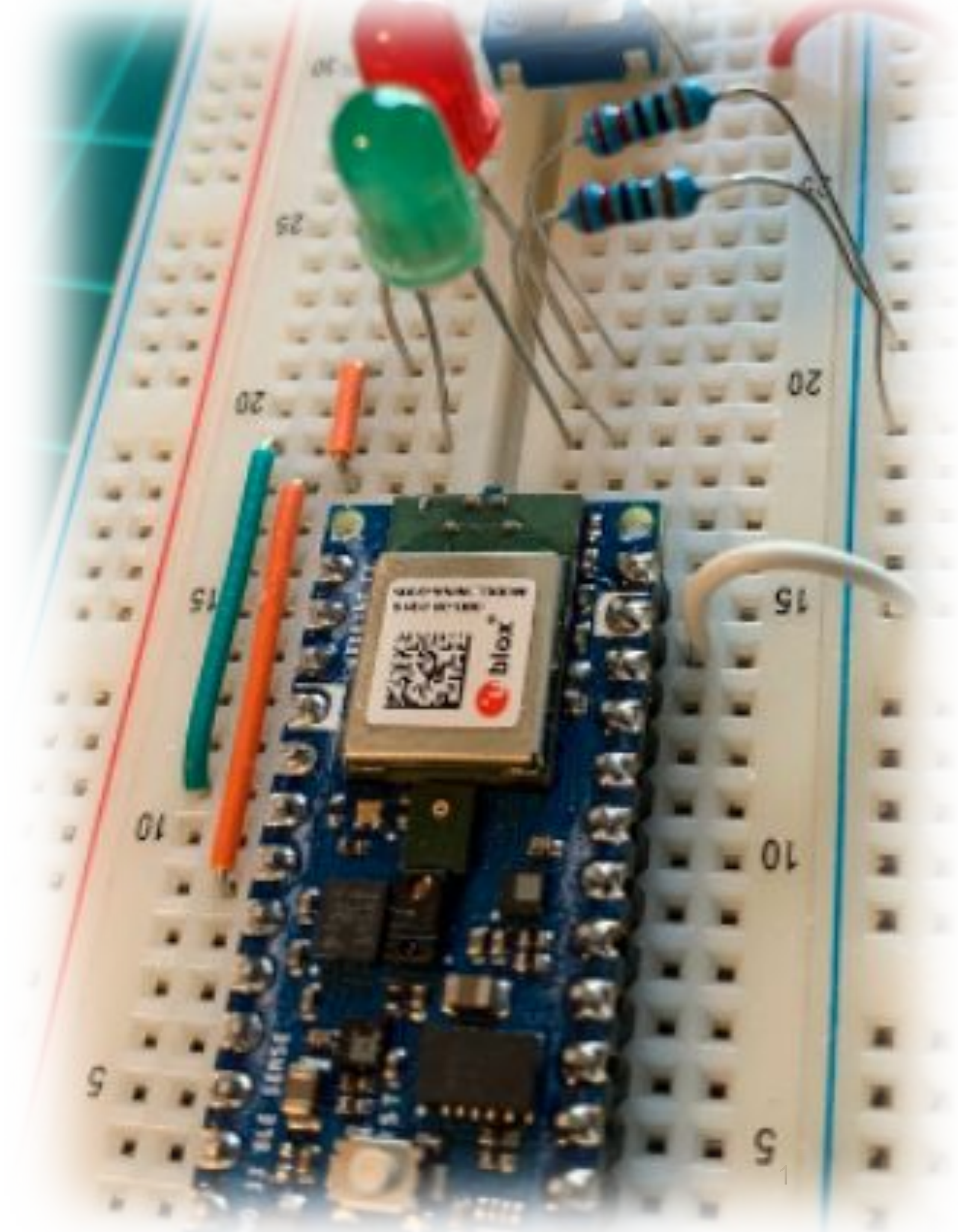
IESTI01 – TinyML

Embedded Machine Learning

23. KeyWord Spotting (KWS) Introduction

Prof. Marcelo Rovai
UNIFEI



What is KeyWord Spotting?

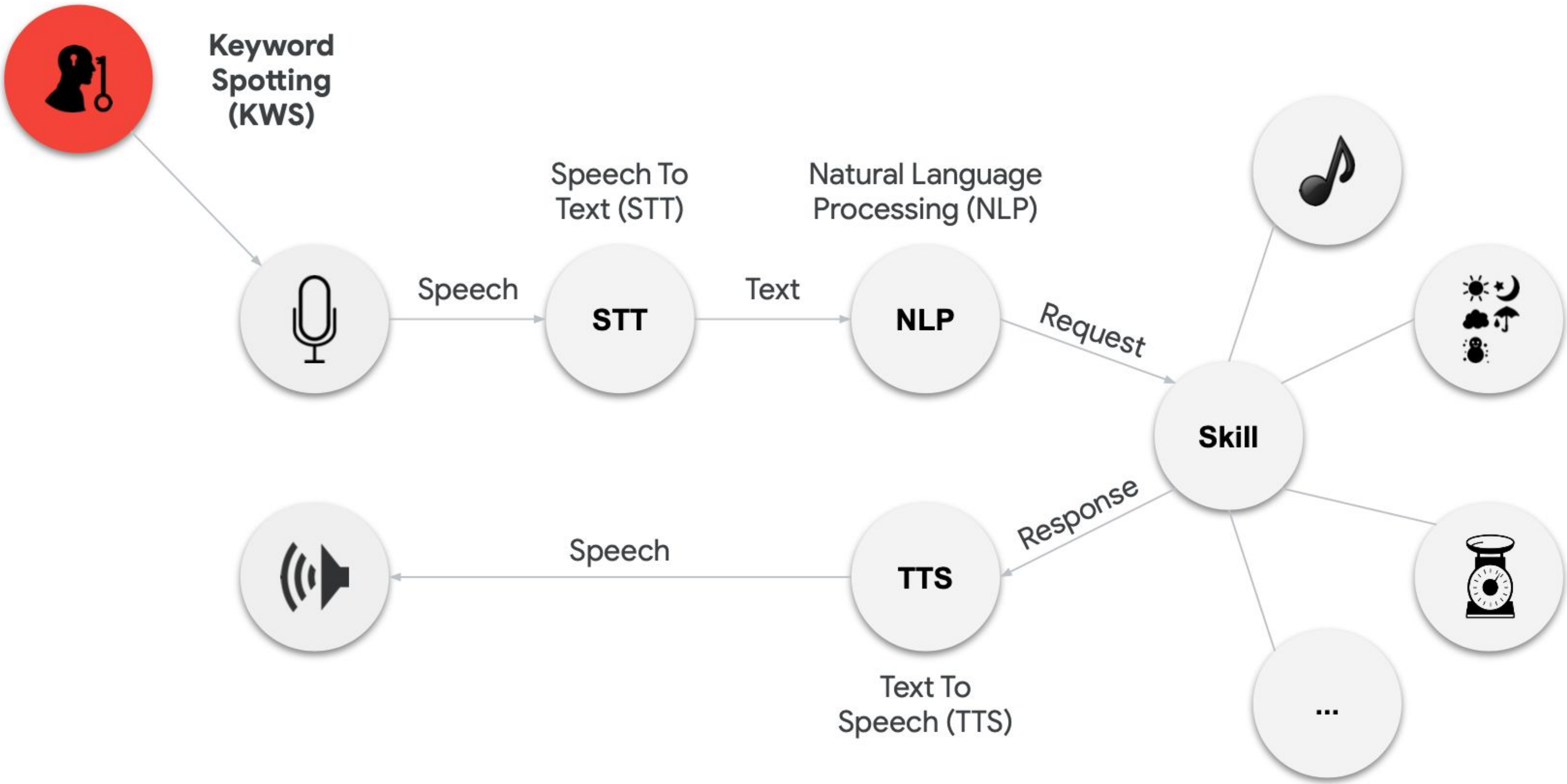
Keyword Spotting v.
General Speech Recognition

- **Keyword spotting** is one of the most successful examples of **TinyML**
  - Low-power, continuous, on-device
  - Common Voice SWTS* expands keyword spotting to more languages

    * **S**ingle **W**ord **T**arget **S**egment

- **General ASR*** still requires **larger, power-hungry models**
  - But it can run on mobile devices (offline dictation on smartphones)

    * Automatic Speech Recognition


Keyword
Spotting
(KWS)
Speech To
Text (STT)
Natural Language
Processing (NLP)
Speech
STT
Text
NLP
Request
Skill
Response
Speech
TTS
...
Text To
Speech (TTS)


amazon


amazon


amazon

More than just voice

- Secutity (Broken Glass)
- Industry (Anomaly Detection)
- Medical (Snore, Toss)
- Nature (Bee, insect sound)


Keyword Spotting
Challenges/Constrains

Challenges and Constraints

- Latency & Bandwidth
- Accuracy & Personalization
- Security & Privacy
- Battery & Memory

Challenges and Constraints

Latency & Bandwidth

Accuracy & Personalization

Security & Privacy

Battery & Memory

LATENCY

Provide results quickly, respond in real-time to the user

Challenges and Constraints

Latency & Bandwidth

Accuracy & Personalization

Security & Privacy

Battery & Memory

BANDWIDTH

Minimize data sent over the network (slow and expensive)

Challenges and Constraints

Latency & Bandwidth

Accuracy & Personalization

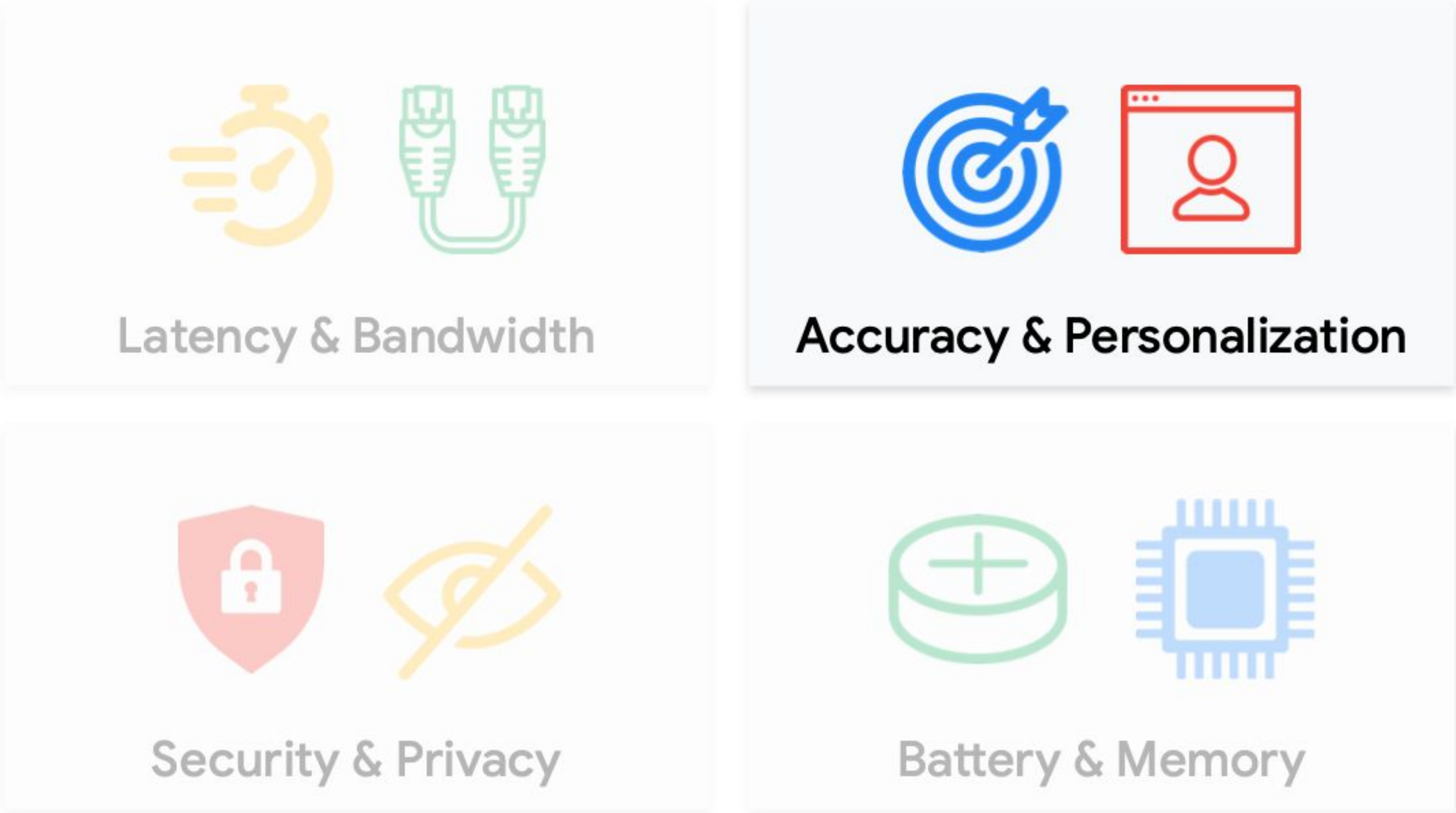
Security & Privacy

Battery & Memory

ACCURACY

Listen continuously,
but only trigger at the right time

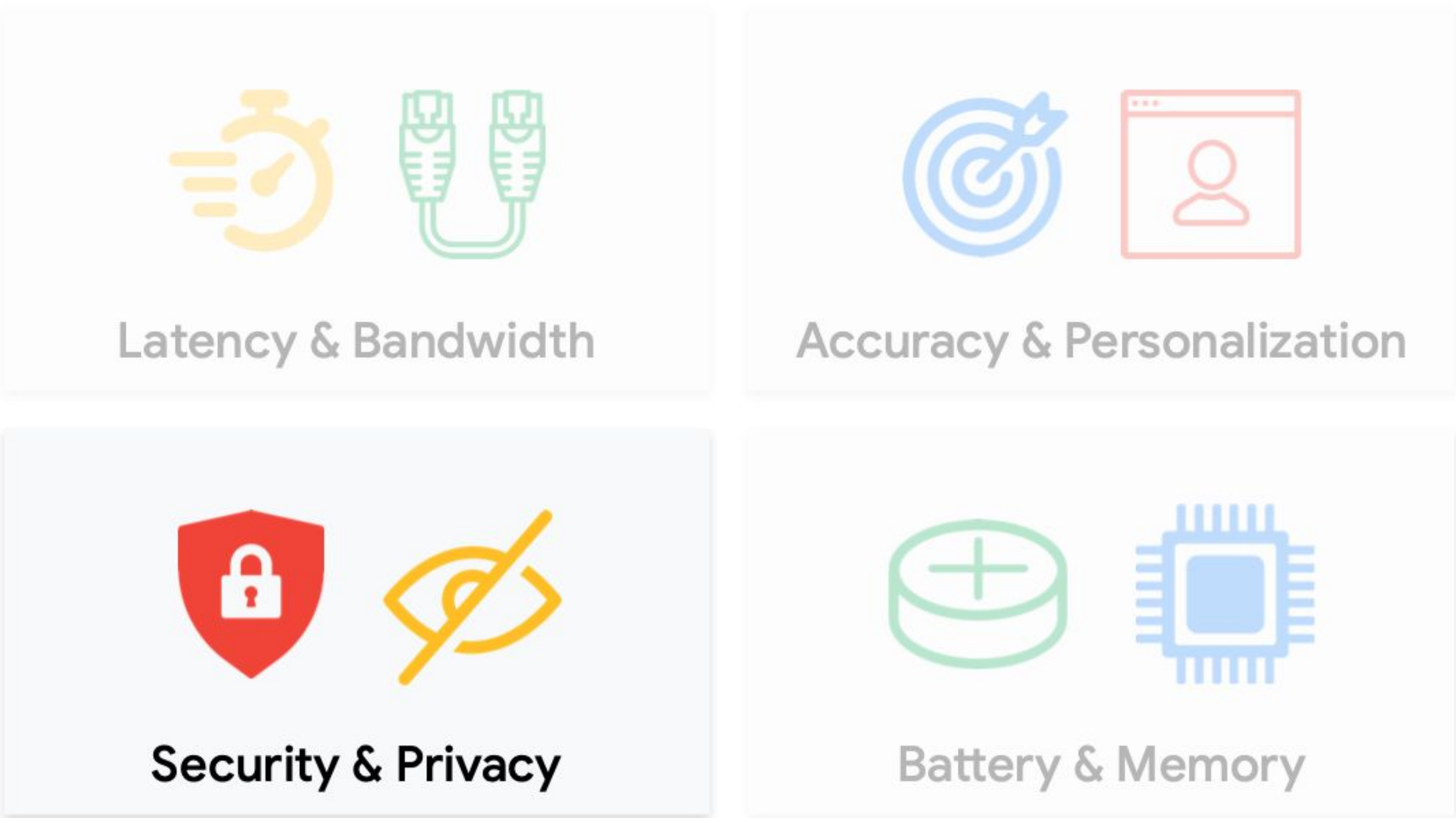
Challenges and Constraints



PERSONALIZATION

Trigger for the user and **not** for background noise

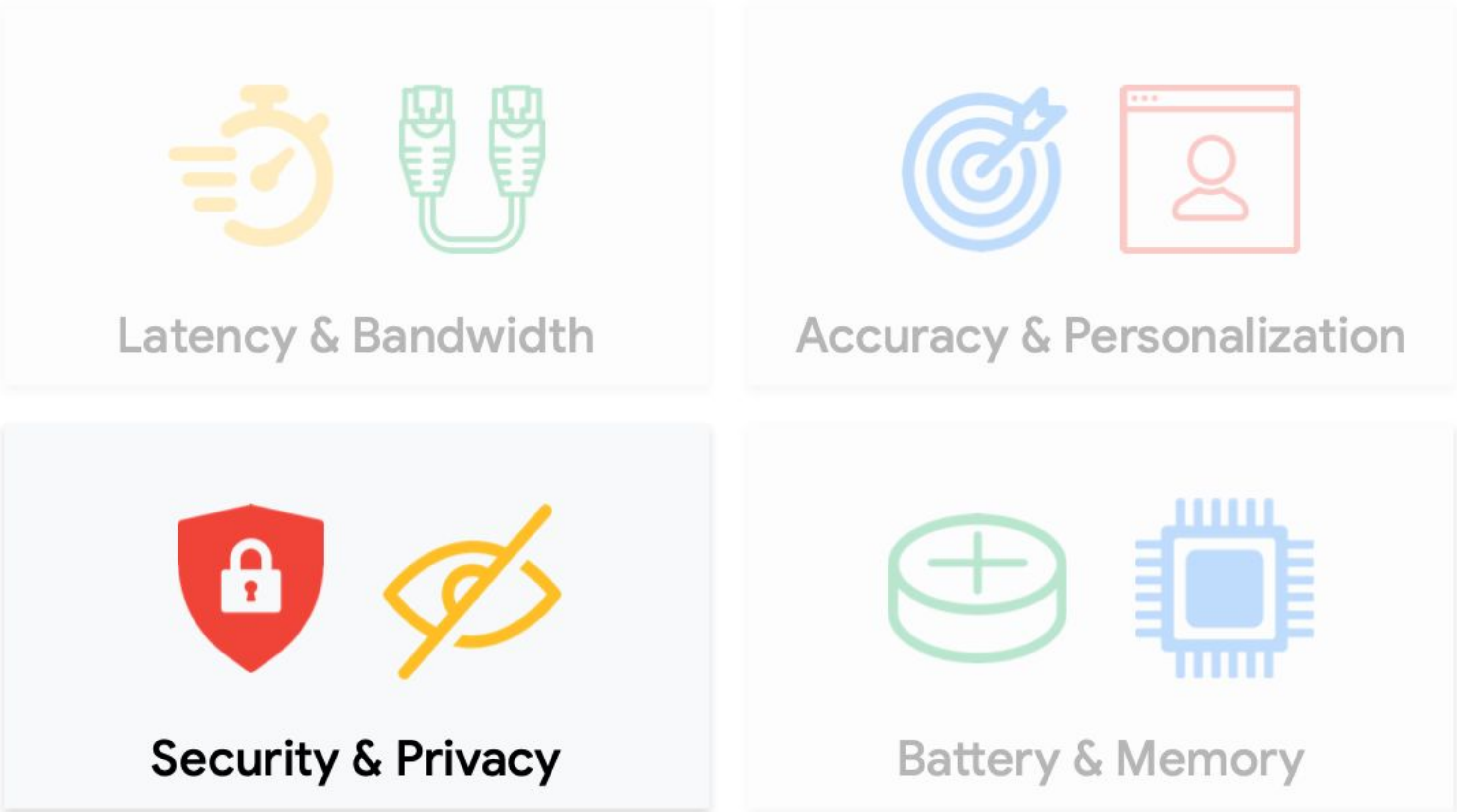
Challenges and Constraints



SECURITY

Safeguarding the data that is being sent to the cloud

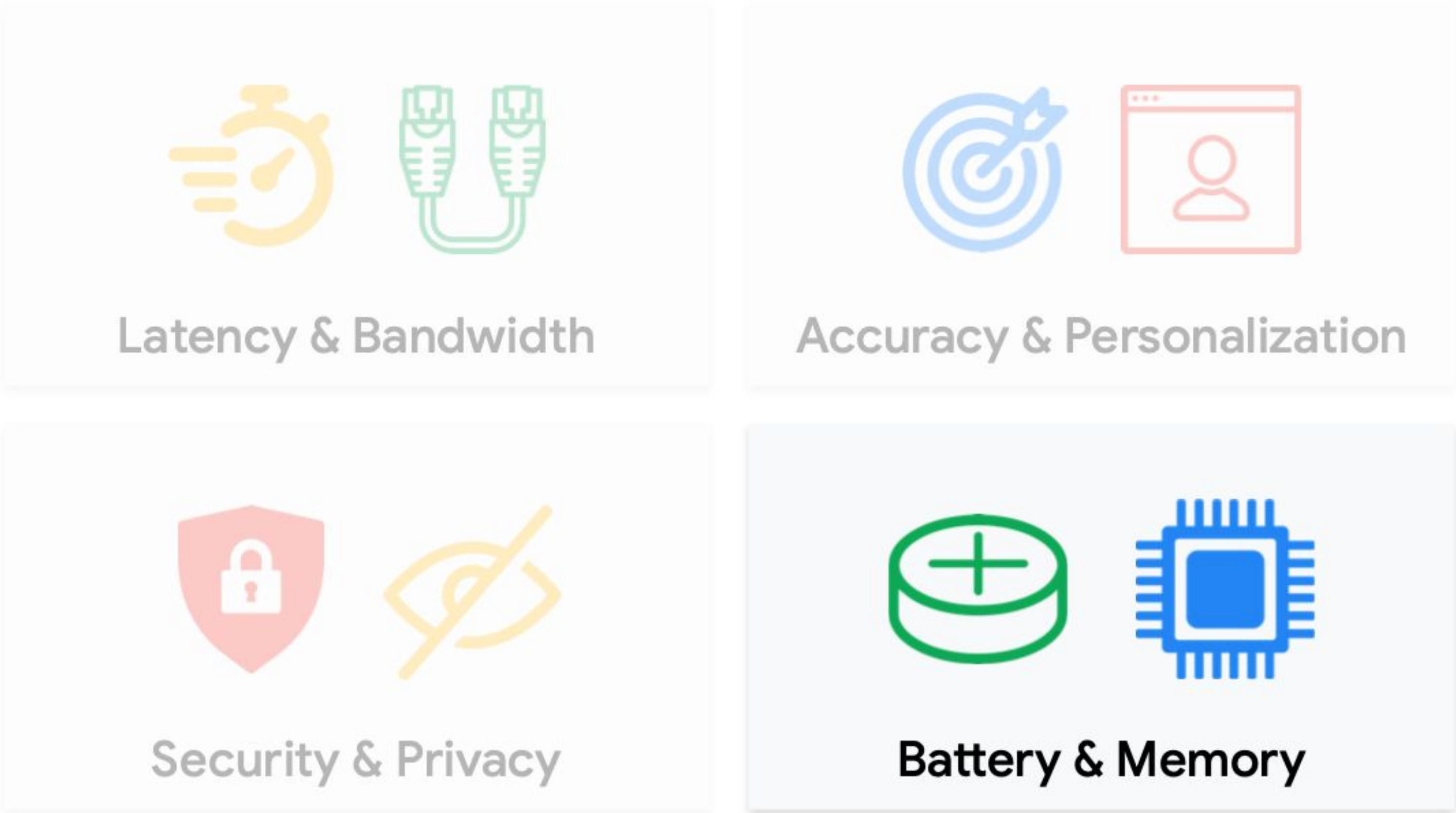
Challenges and Constraints



PRIVACY

Safeguarding the data that is being sent to the cloud

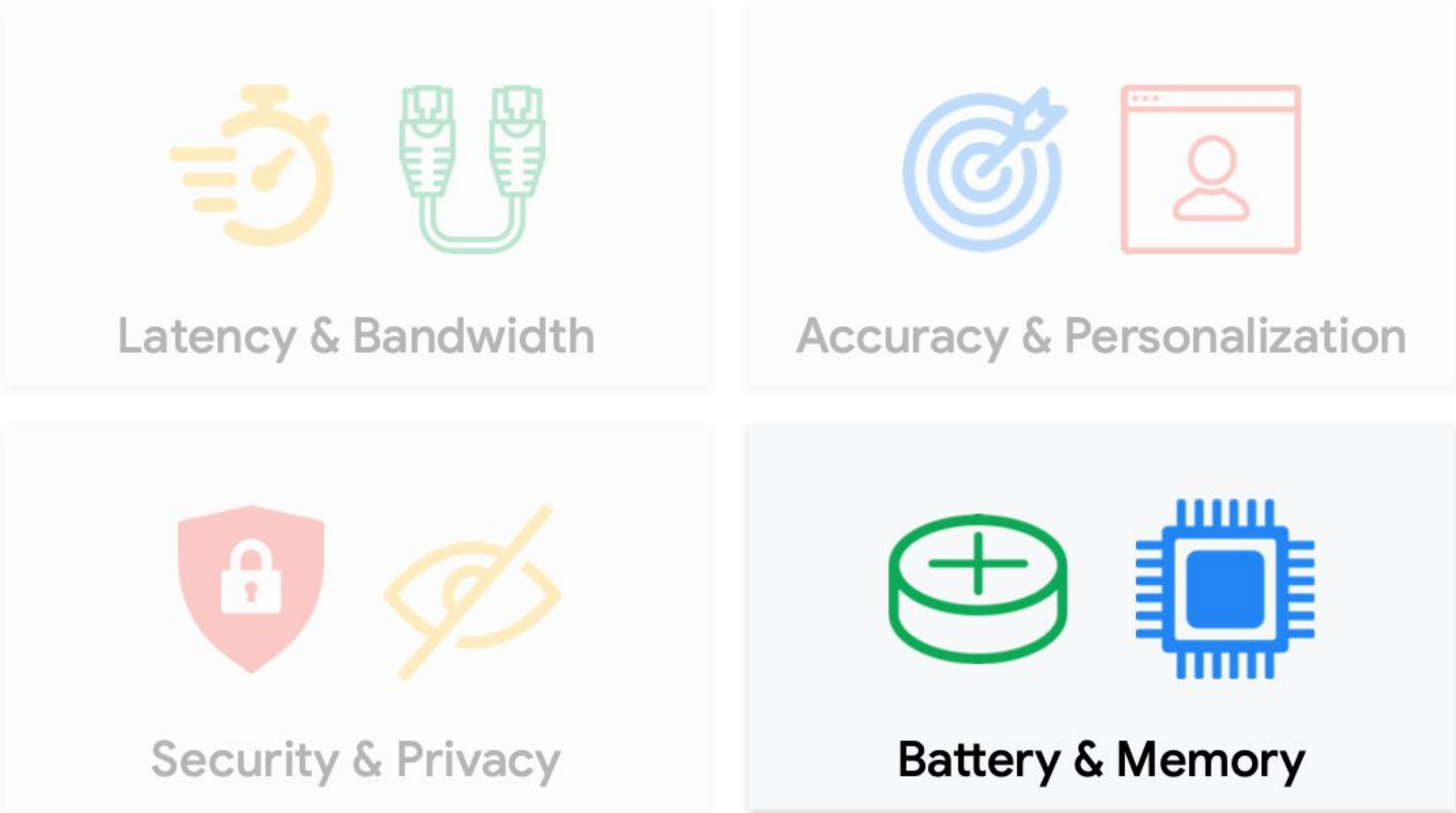
Challenges and Constraints



BATTERY

Limited energy, operate on coin-cell type batteries

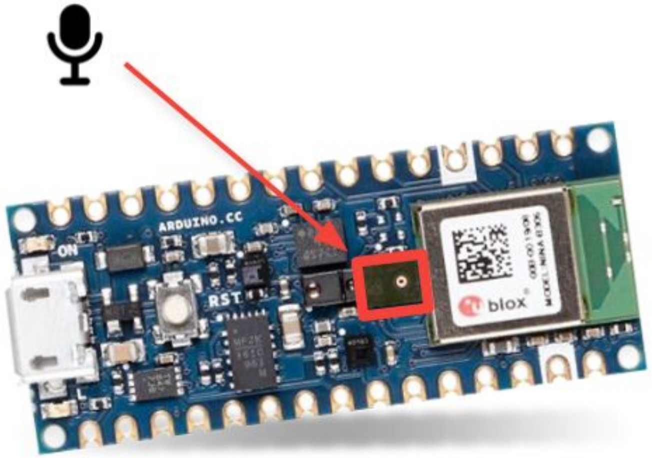
Challenges and Constraints



MEMORY

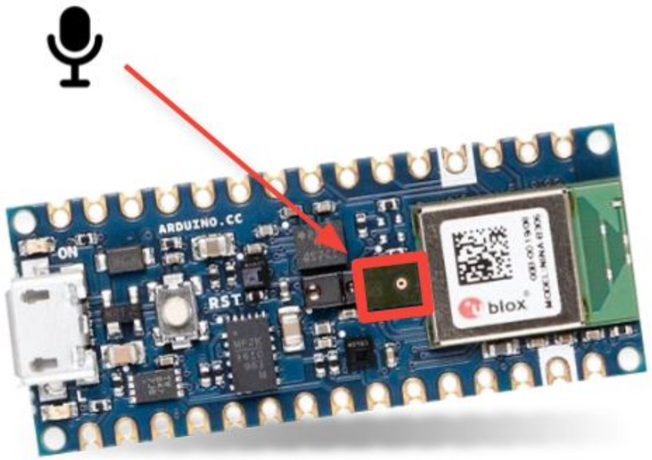
Run on resource constrained devices

Anatomy of a Keyword Spotting Application



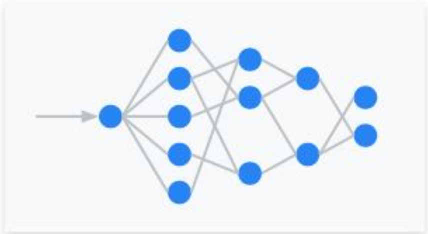
1 Continuously listen on the microphone
the microcontroller

Anatomy of a Keyword Spotting Application



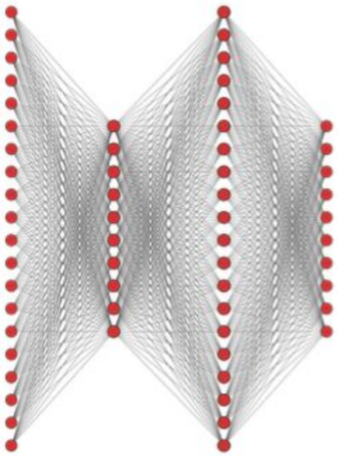
1 Continuously listen on the microcontroller

2 Process the data with **TinyML** at the edge

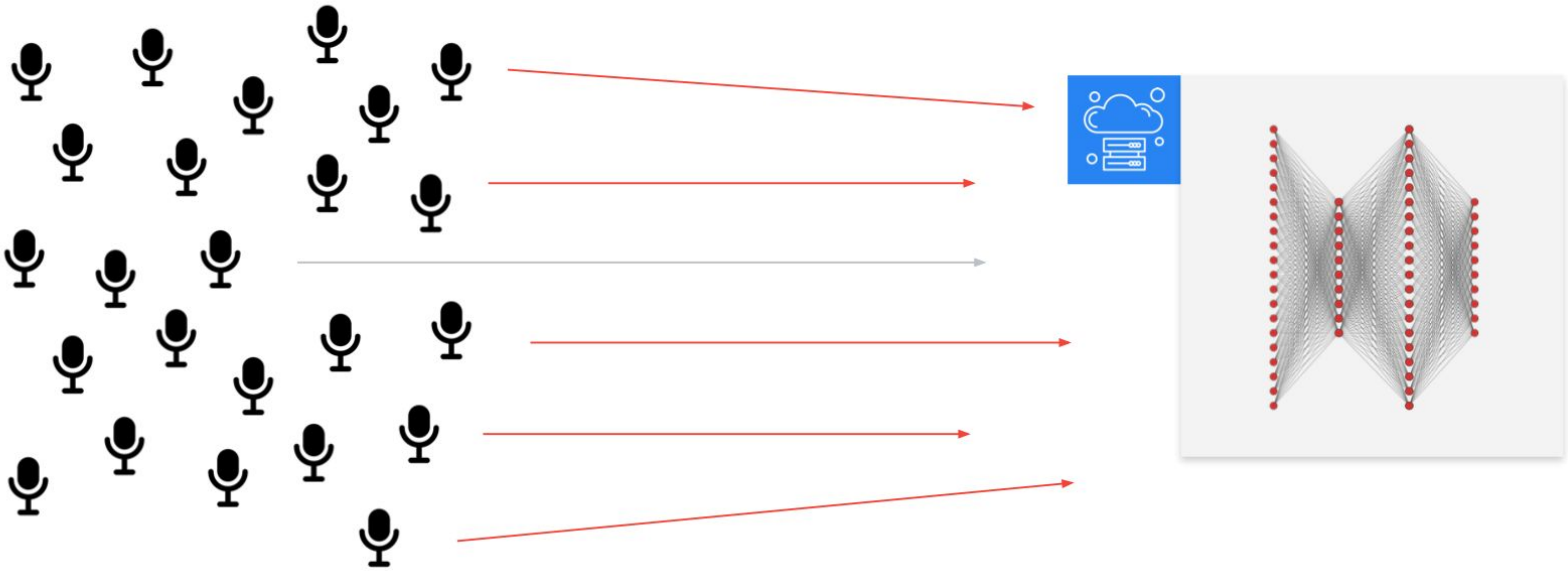


3 Send the data to the cloud when triggered

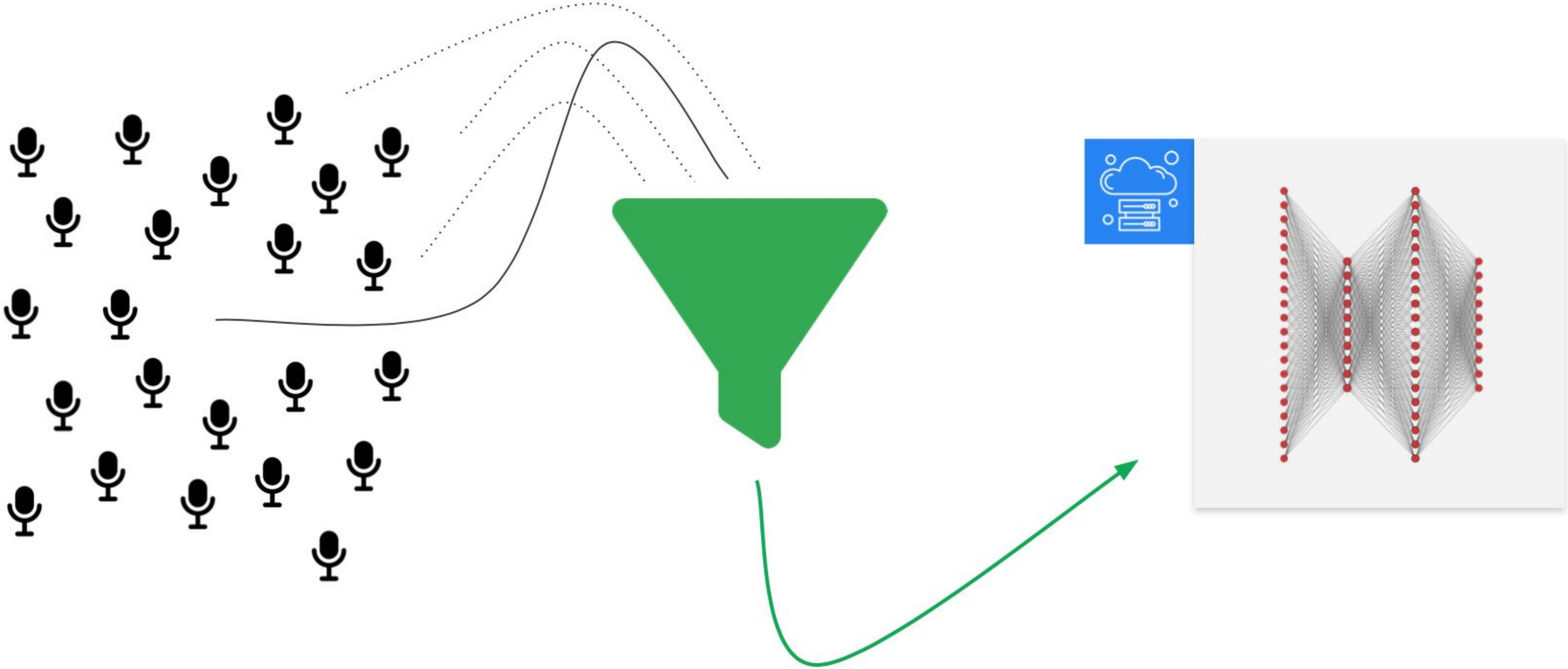
4 Process the full speech data with a large model in the cloud



Anatomy of a Keyword Spotting Application



Anatomy of a Keyword Spotting Application



Keyword Spotting
Datasets

How do we build a **good** dataset?

- Who are the **users**?
- What do they **need**?
- What **task** are they trying to solve?
- How do they **interact** with the system?
- How does the **real world** make this hard?

Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition

Pete Warden
Google Brain
Mountain View, California
petewarden@google.com

April 2018

https://arxiv.org/pdf/1804.03209.pdf

Requirements

"yes" ✔

"no" ✖

Common Use

"left"
"right"
"go"
"stop"

Robotics

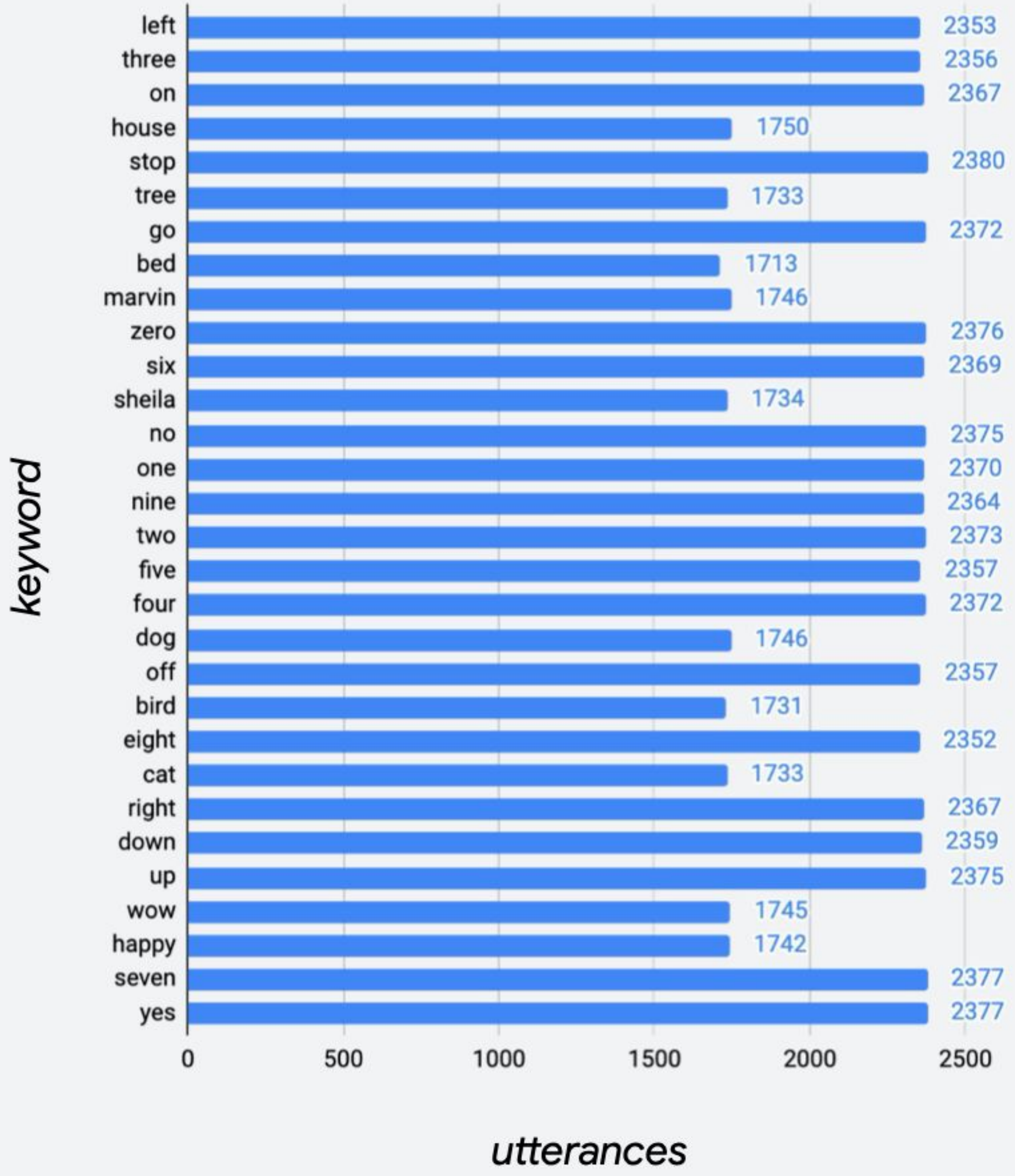
"one"
"two"
"four"
"six"

#

Numbers

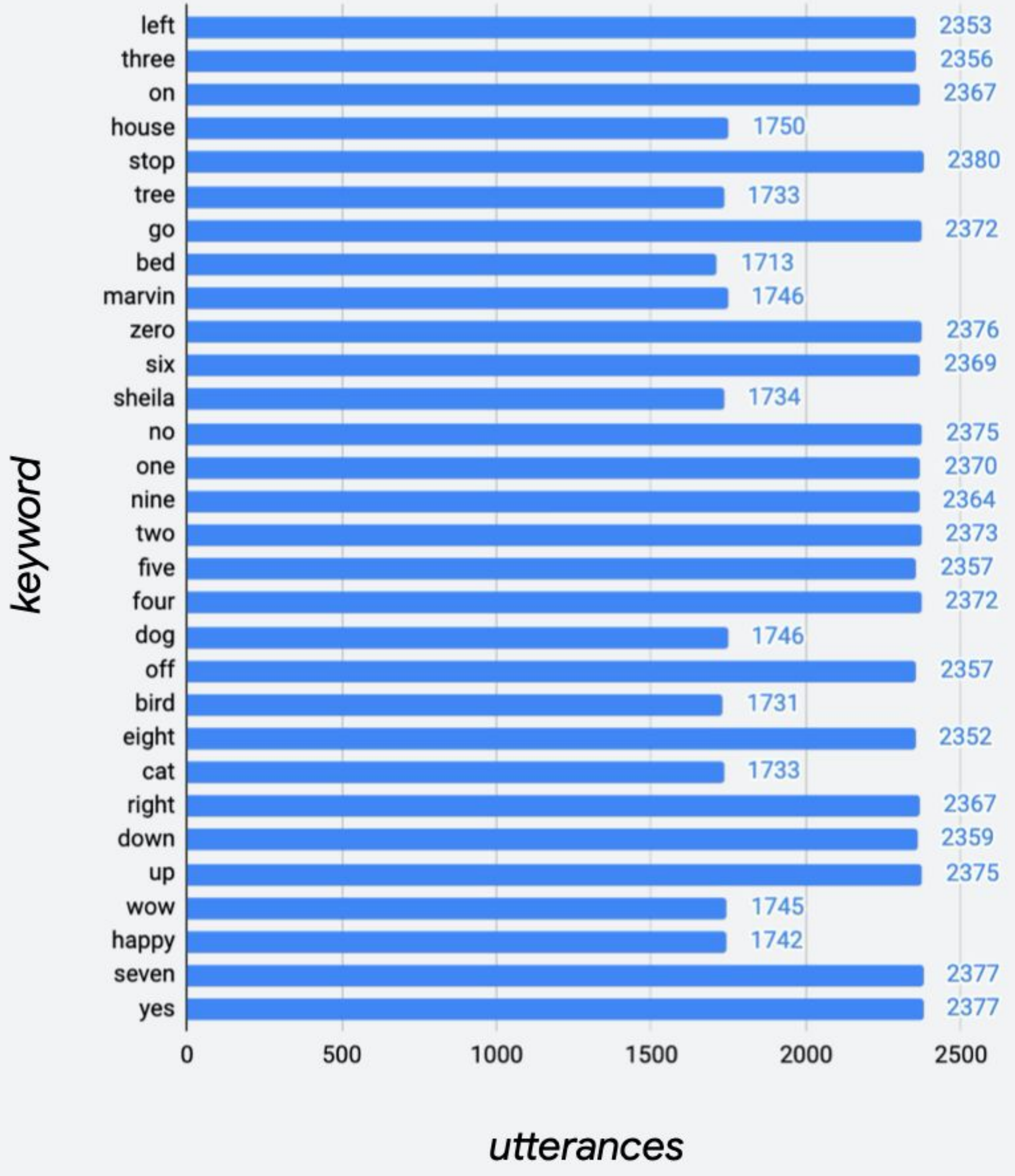
Data Collection

- **2,618** volunteers
  - consented to have their voices redistributed
  - Variety of accents
- > 1,000 examples for **each** keyword
- **Browser-based**
  (no app to install)



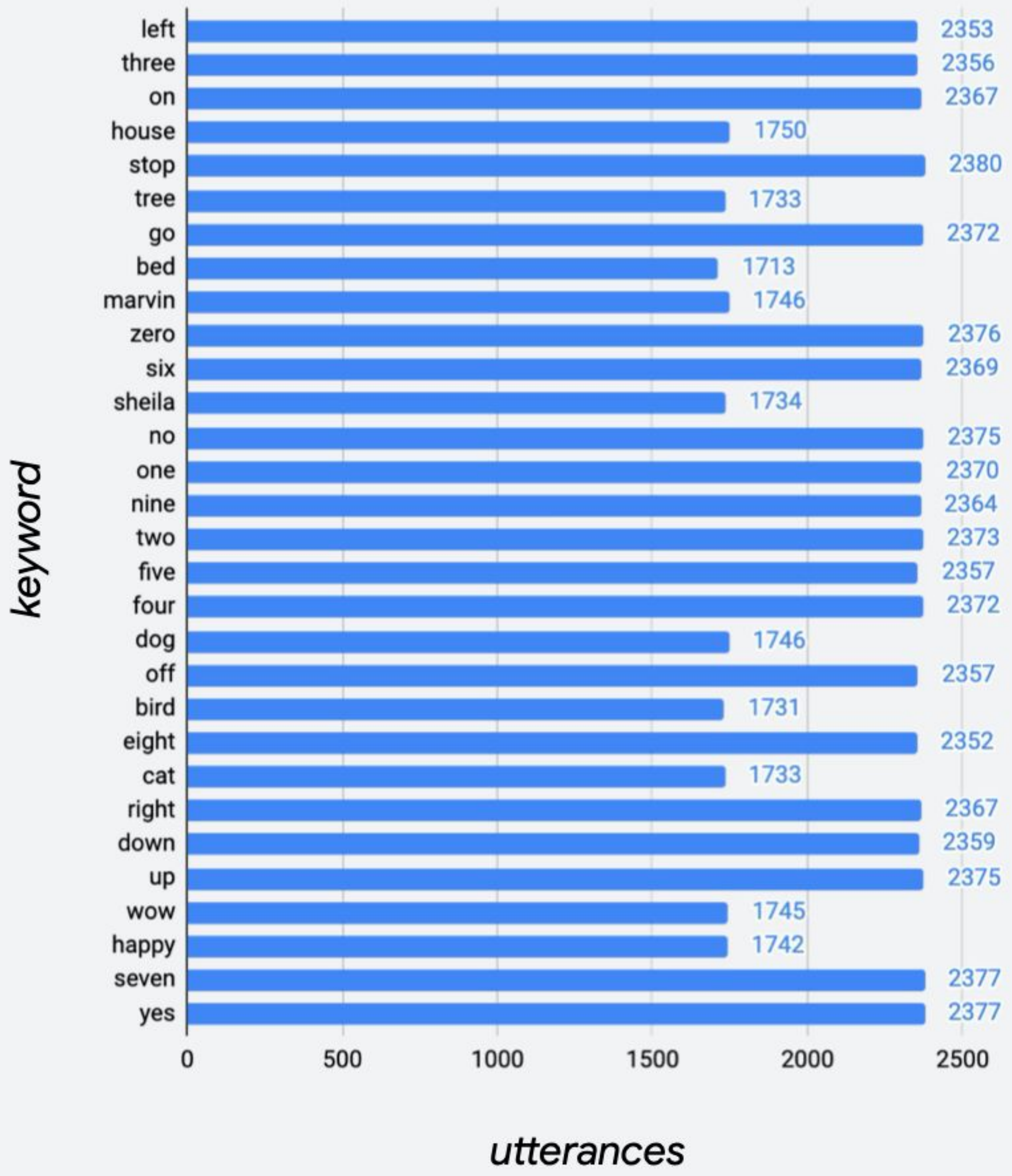
Data Validation

- Some data is **unusable**
  - Too quiet, wrong word, etc
- Started with **automated tools**
  - Remove low volume recordings
  - Extract loudest 1s (from 1.5sec examples)
- All 105,829 remaining utterances **manually reviewed** through crowdsourcing



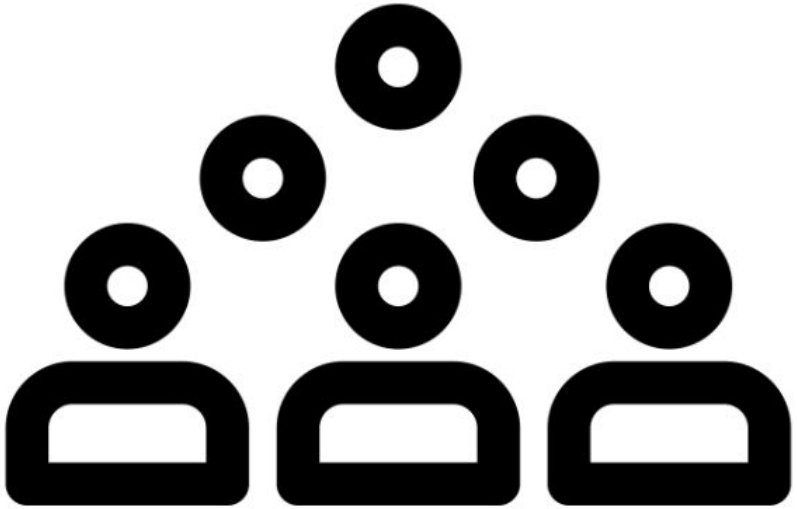
Sustaining KWS Research

- Speech Commands is now in **v2**
  - **Expanded to 35** keywords from original 10
- Includes train/validation/test splits
- Expand to **new languages**?



Common Voice

- **Crowdsourcing** platform

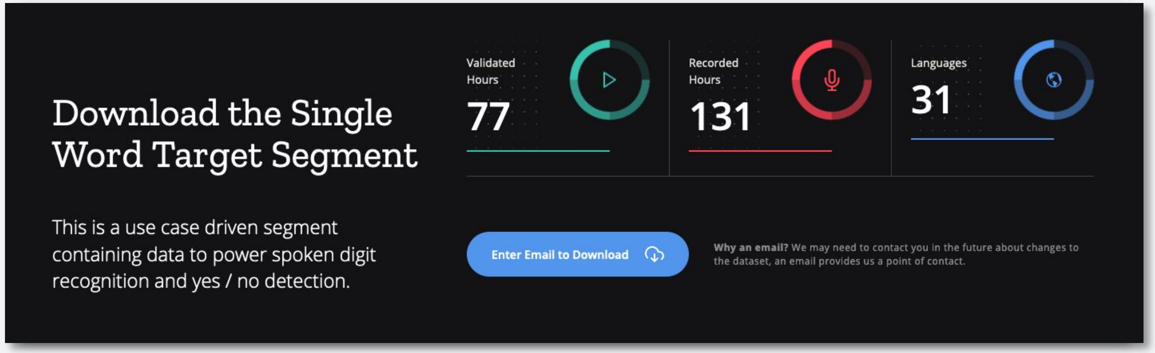


https://commonvoice.mozilla.org/en

Single Word Target Segment

A ***speech commands-style*** dataset for **18 languages**

- "Yes" // "no"
- "hey" & "Firefox"
- **digits** 0-9



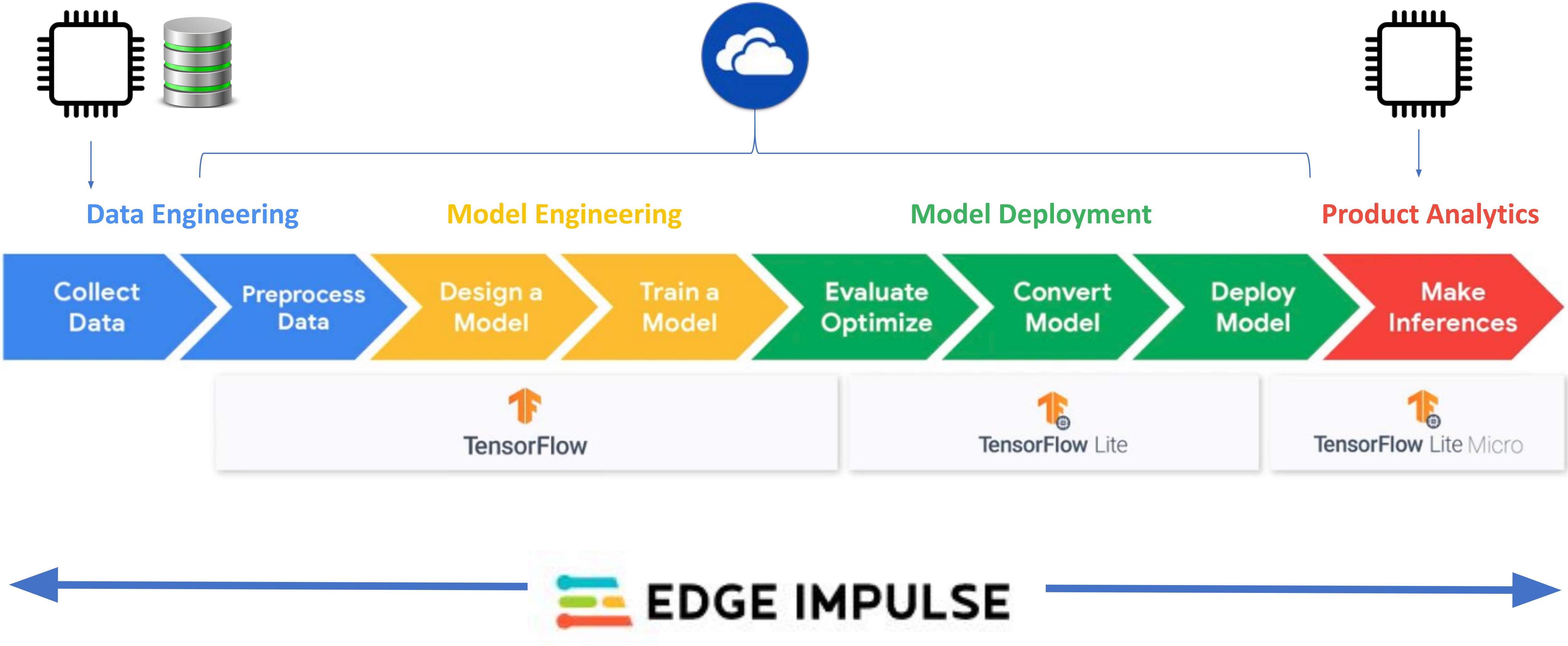
https://commonvoice.mozilla.org/en/datasets

Food for Thought

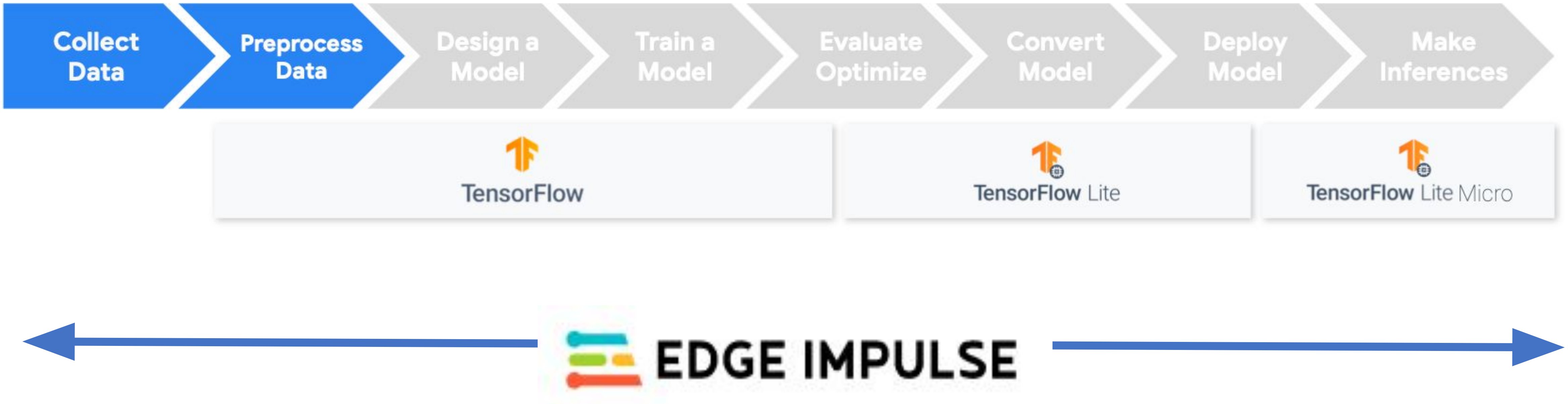
QC (Quality Control)

- Need to keep **only** what a human can hear
- Microphone issues
- **Noisy** backgrounds

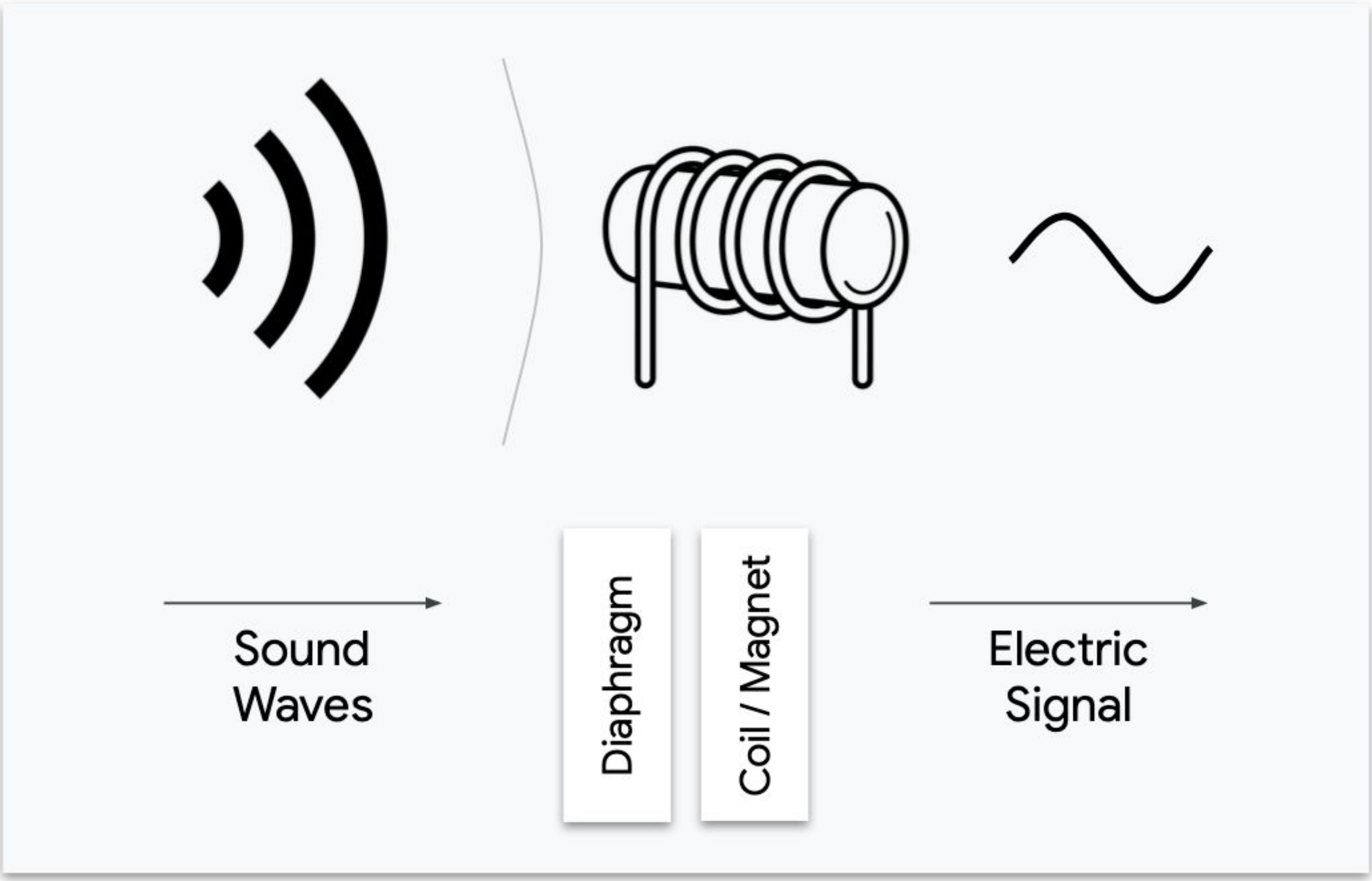
KWS Data Collection & Pre-Processing

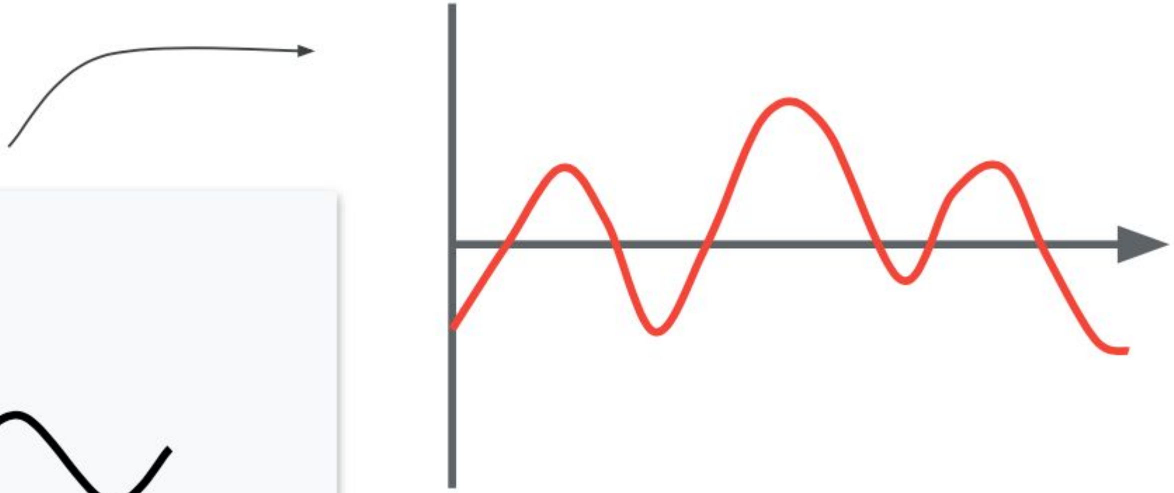


Data Engineering
Model Engineering
Model Deployment
Product Analytics
Collect Data
Preprocess Data
Design a Model
Train a Model
Evaluate Optimize
Convert Model
Deploy Model
Make Inferences
TensorFlow
TensorFlow Lite
TensorFlow Lite Micro
EDGE IMPULSE

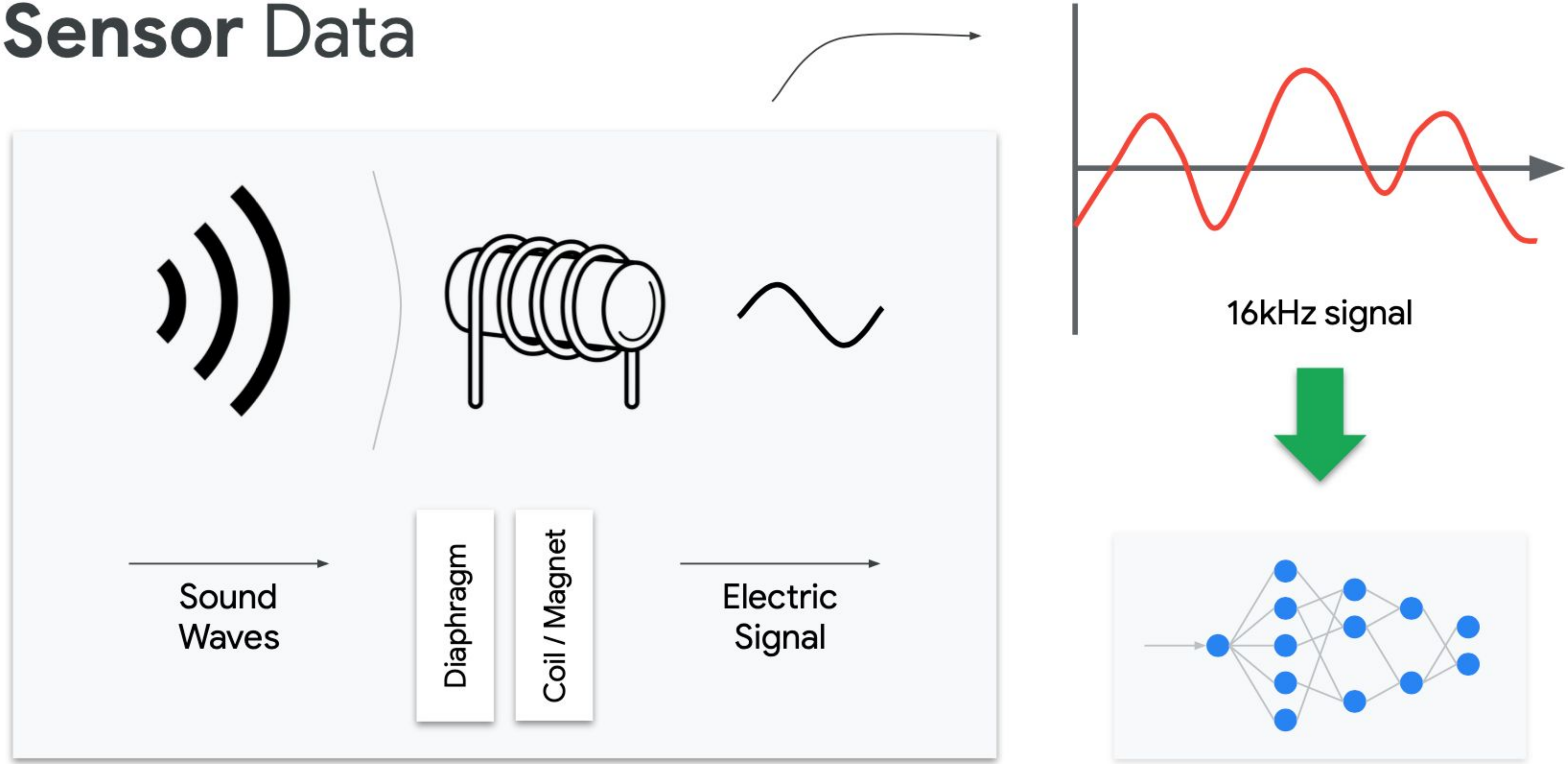

Collect Data
Preprocess Data
Design a Model
Train a Model
Evaluate Optimize
Convert Model
Deploy Model
Make Inferences
TensorFlow
TensorFlow Lite
TensorFlow Lite Micro
EDGE IMPULSE

Sensor Data





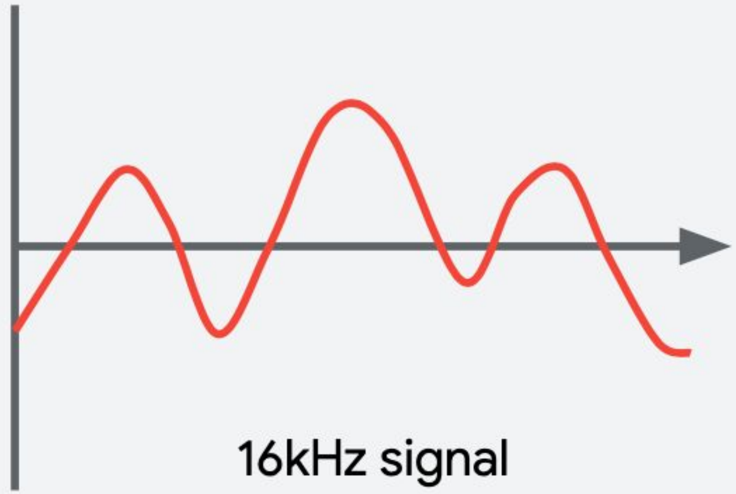
Sensor Data



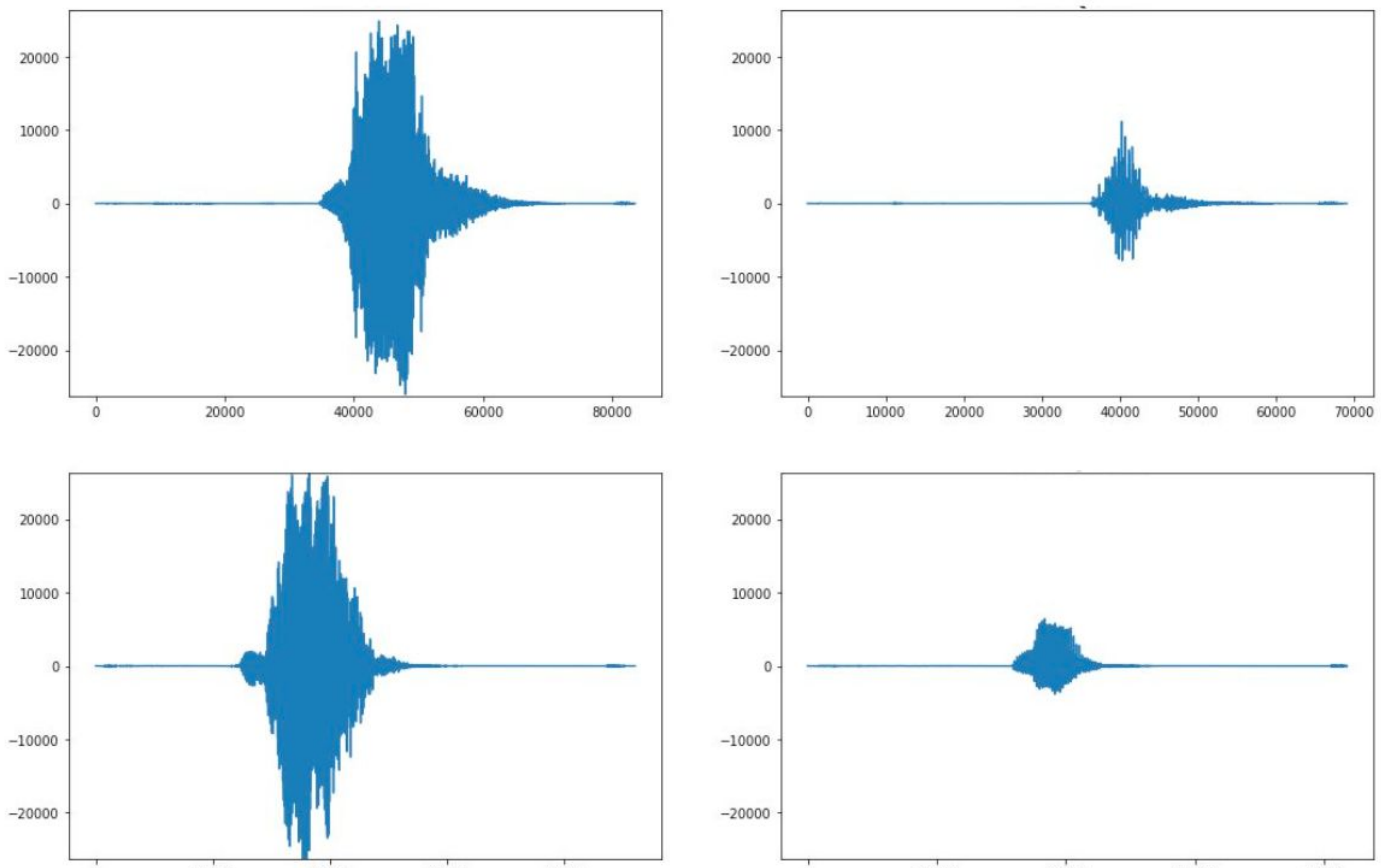
Sound
Waves
Diaphragm
Coil / Magnet
Electric
Signal
16kHz signal

Sensor Data

- 16kHz signal, so that's **16000** samples (points / second)
- How do you feed ***all*** of that data into the network?
- Need to **think creatively** about the input signal!



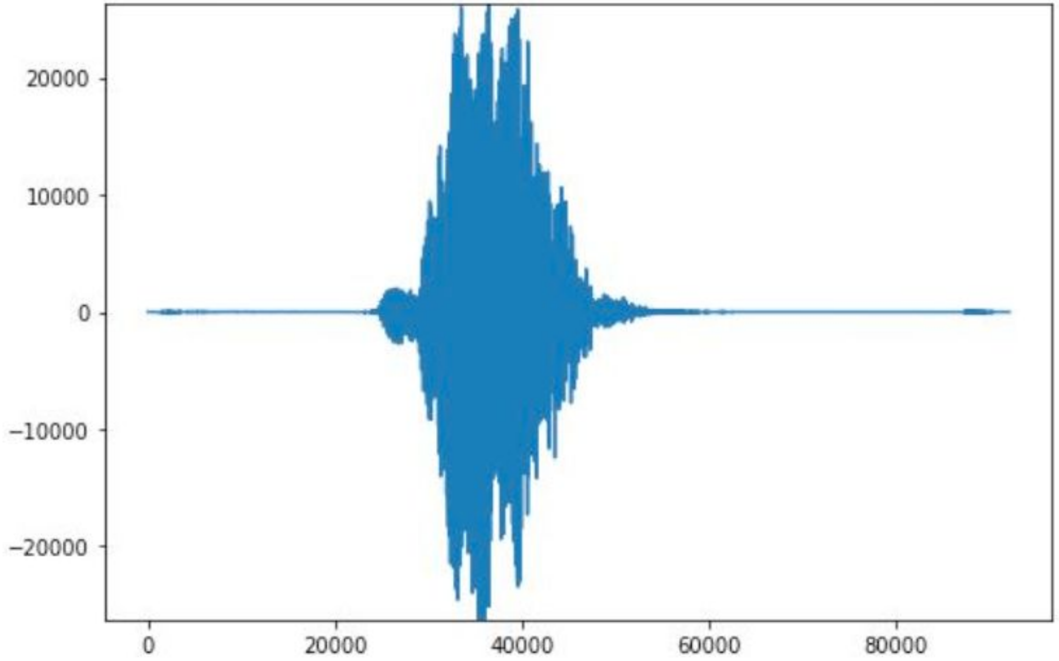
Guess!


20000
10000
0
−10000
−20000
0
20000
40000
60000
80000
20000
10000
0
−10000
−20000
0
10000
20000
30000
40000
50000
60000
70000
20000
10000
0
−10000
−20000
0
20000
40000
60000
80000
20000
10000
0
−10000
−20000
0
20000
40000
60000
80000

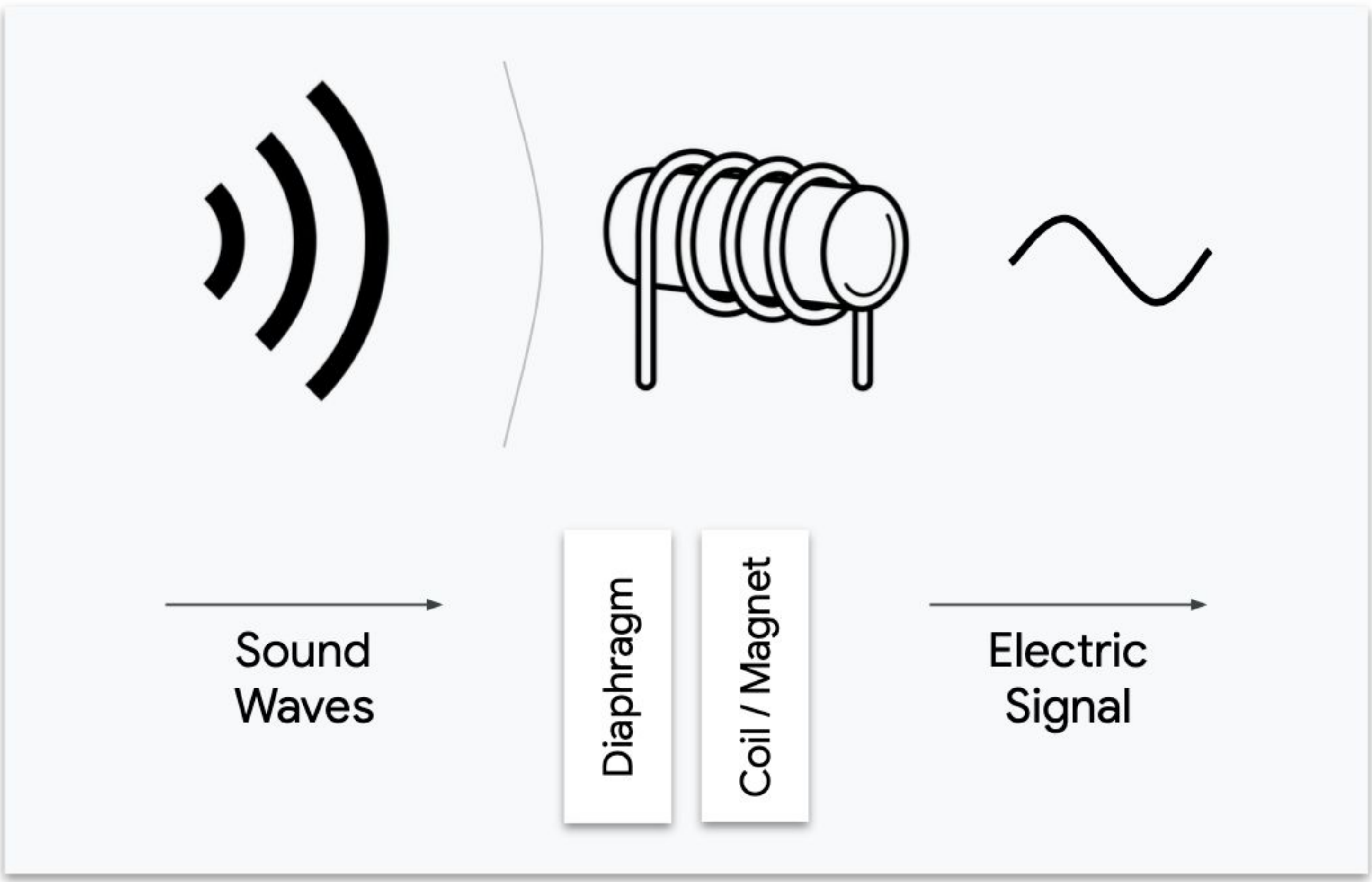
What are interesting **challenges**?

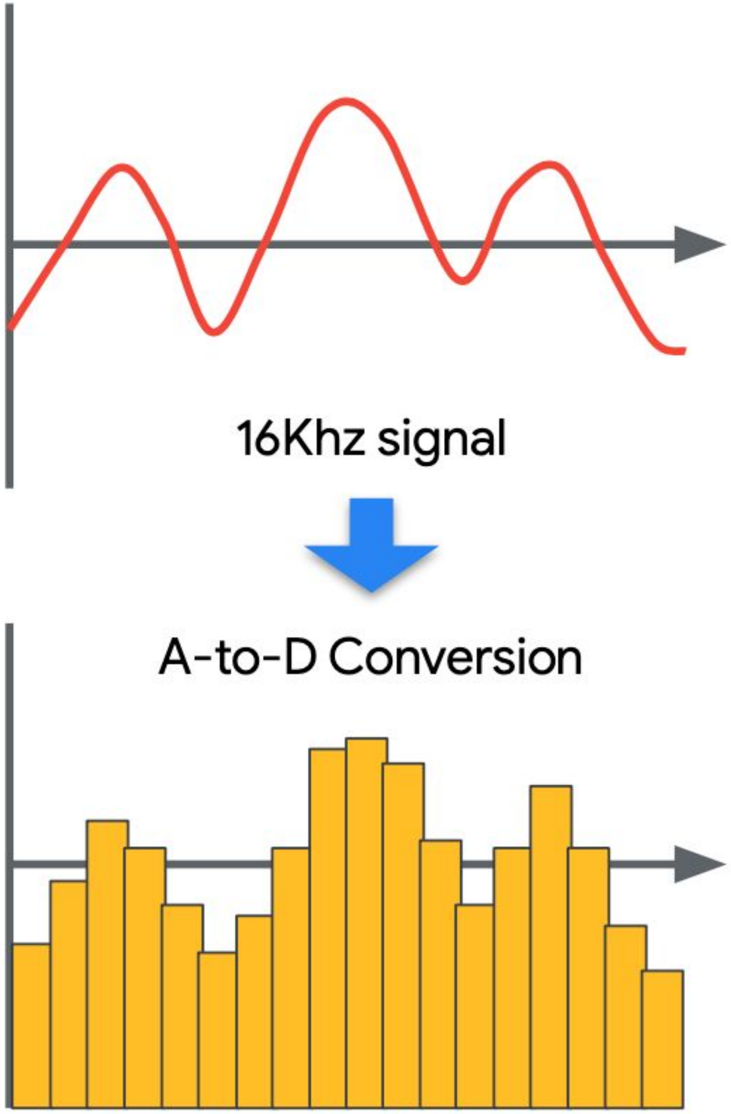
- It is a continuous signal, so **when does the word start**?
- How do you **“align”** on the starting point?
- How do we **extract the vital parts** of the signal that matter?

“No” *(spoken loudly)*

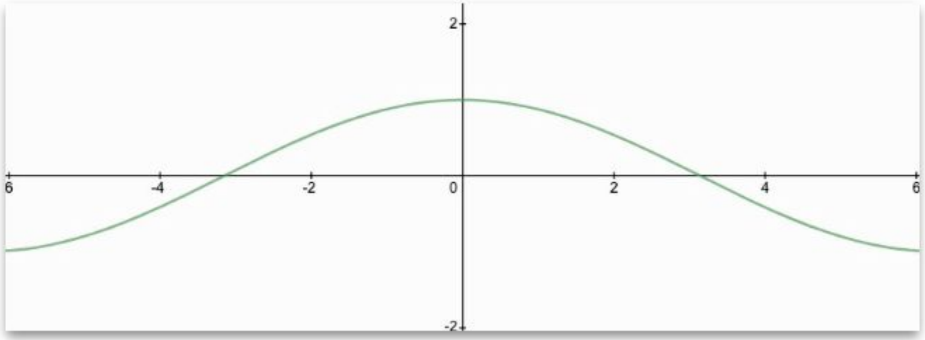


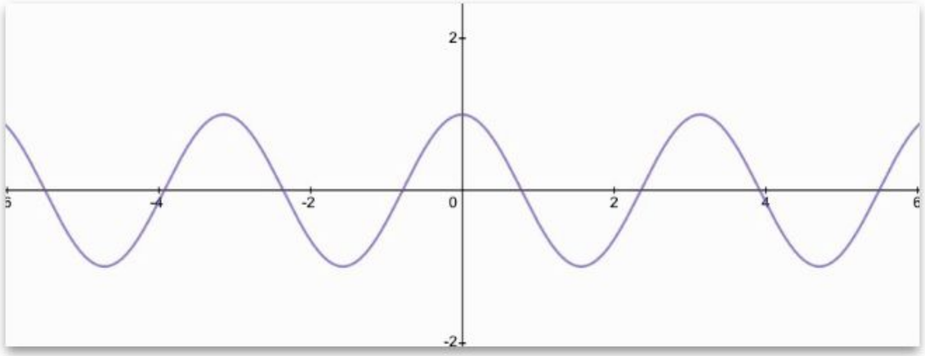
Sensor Data

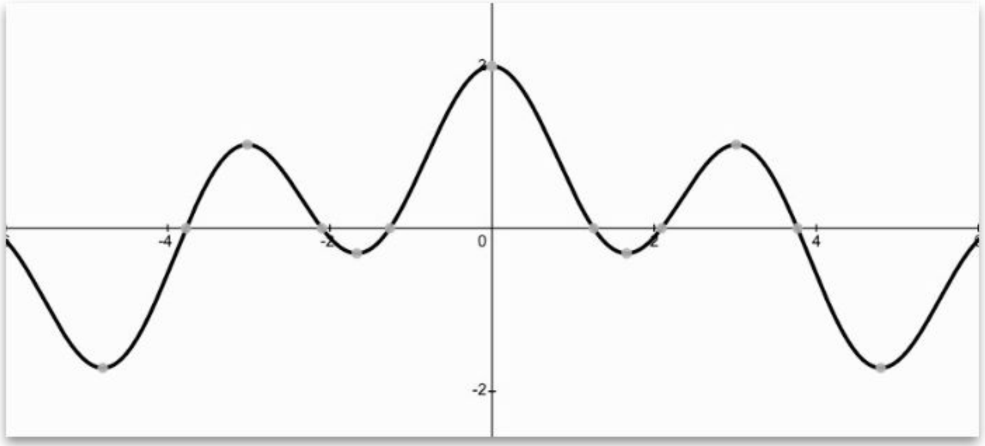

Sound
Waves
Diaphragm
Coil / Magnet
Electric
Signal


16Khz signal
A-to-D Conversion

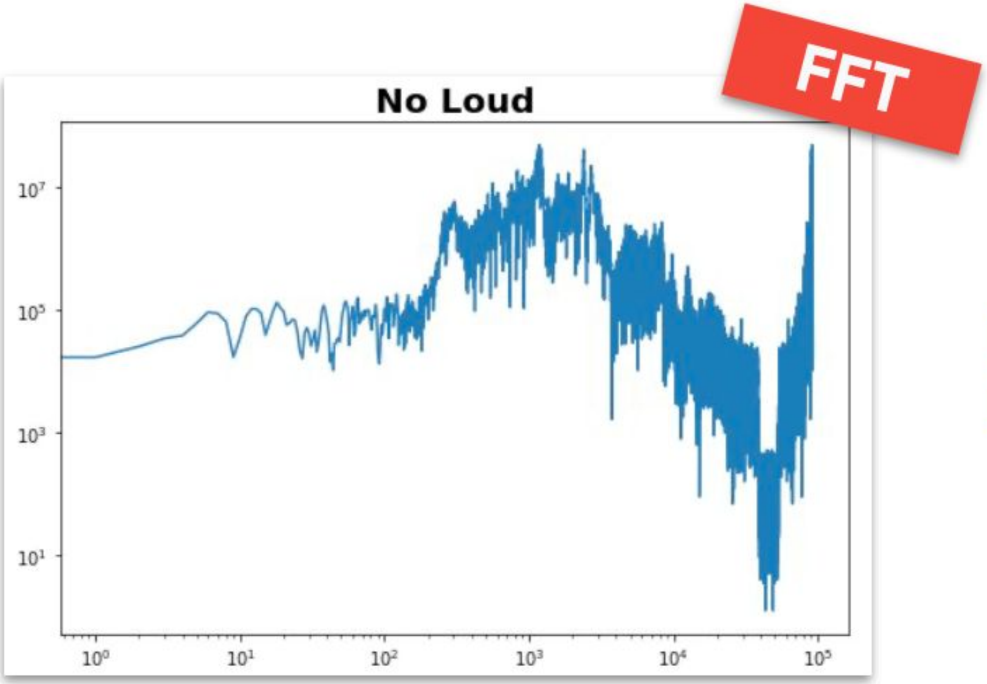
Signal **Components?**





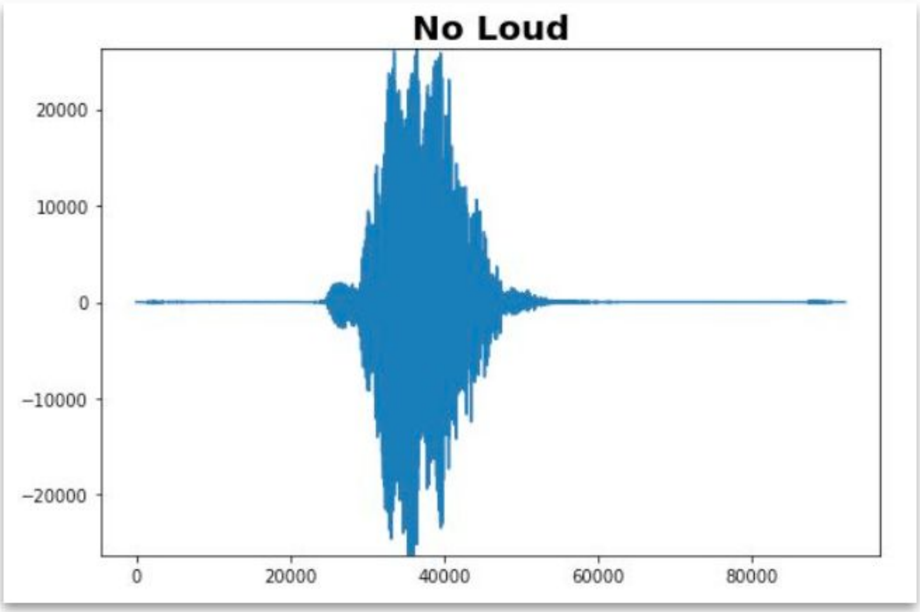


Signal **Components?**



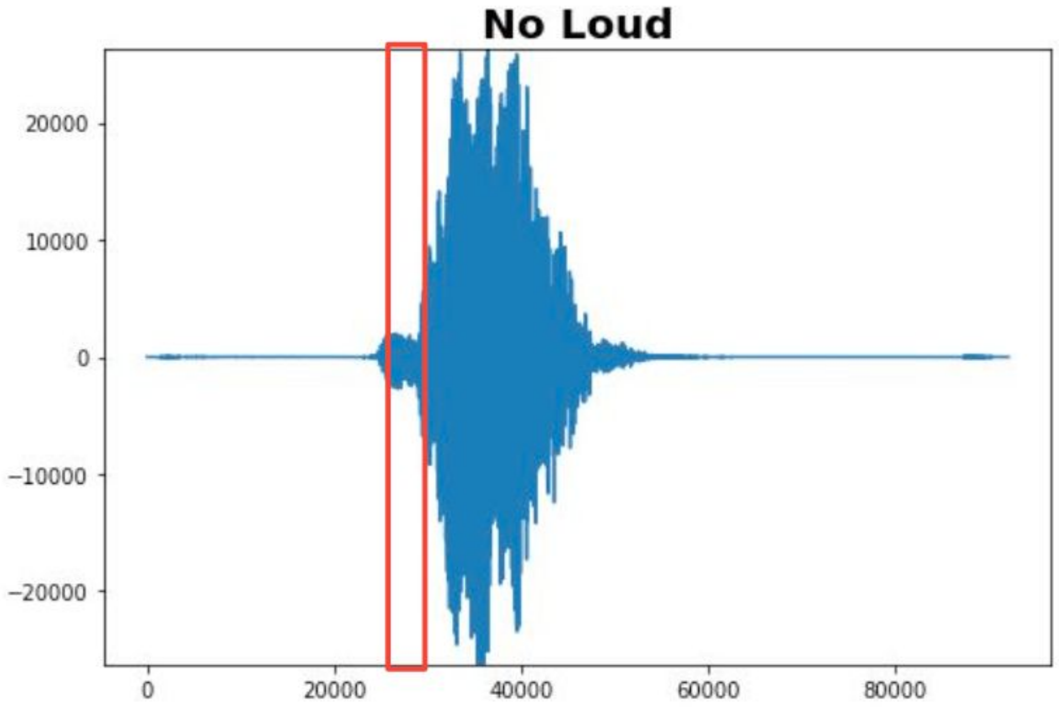
Frequency

=

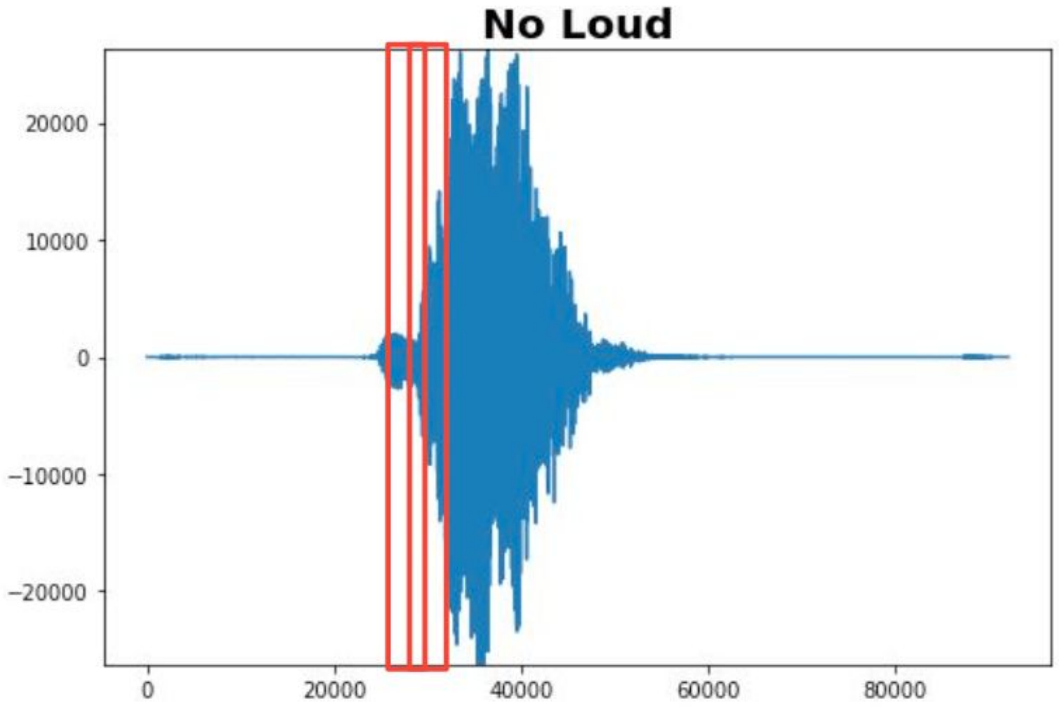


Time

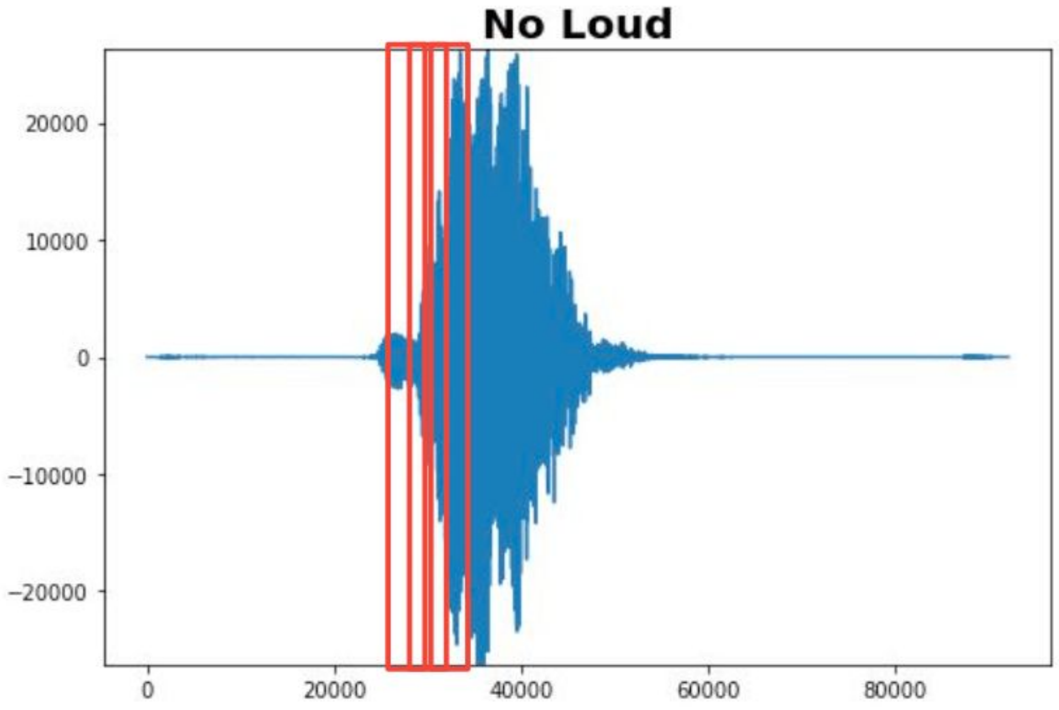
Data Preprocessing



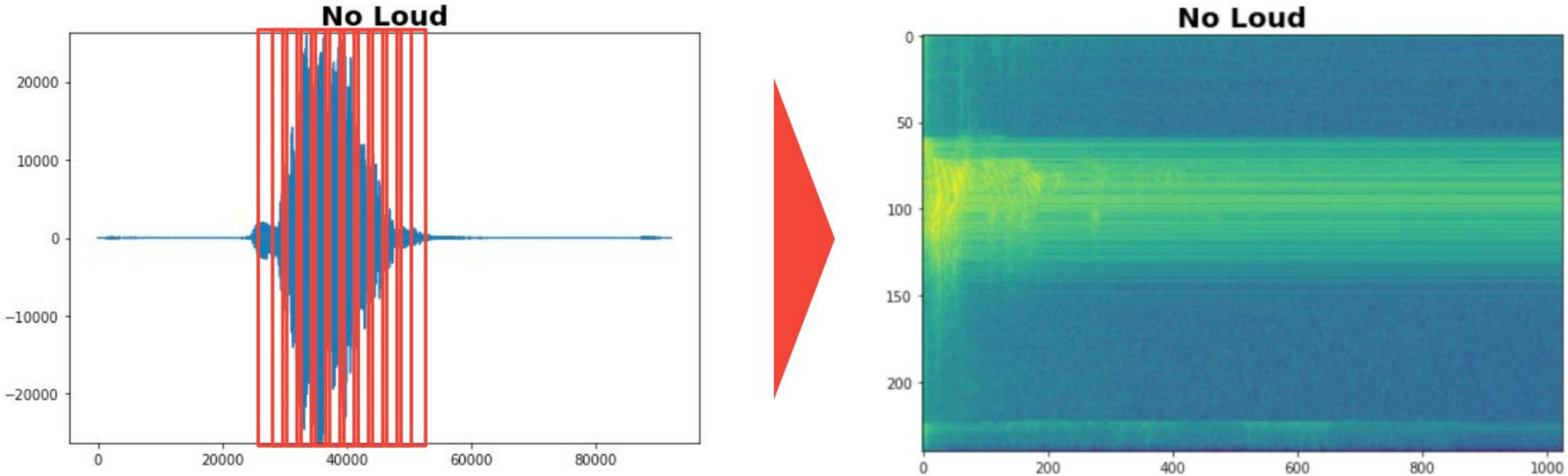
Data Preprocessing



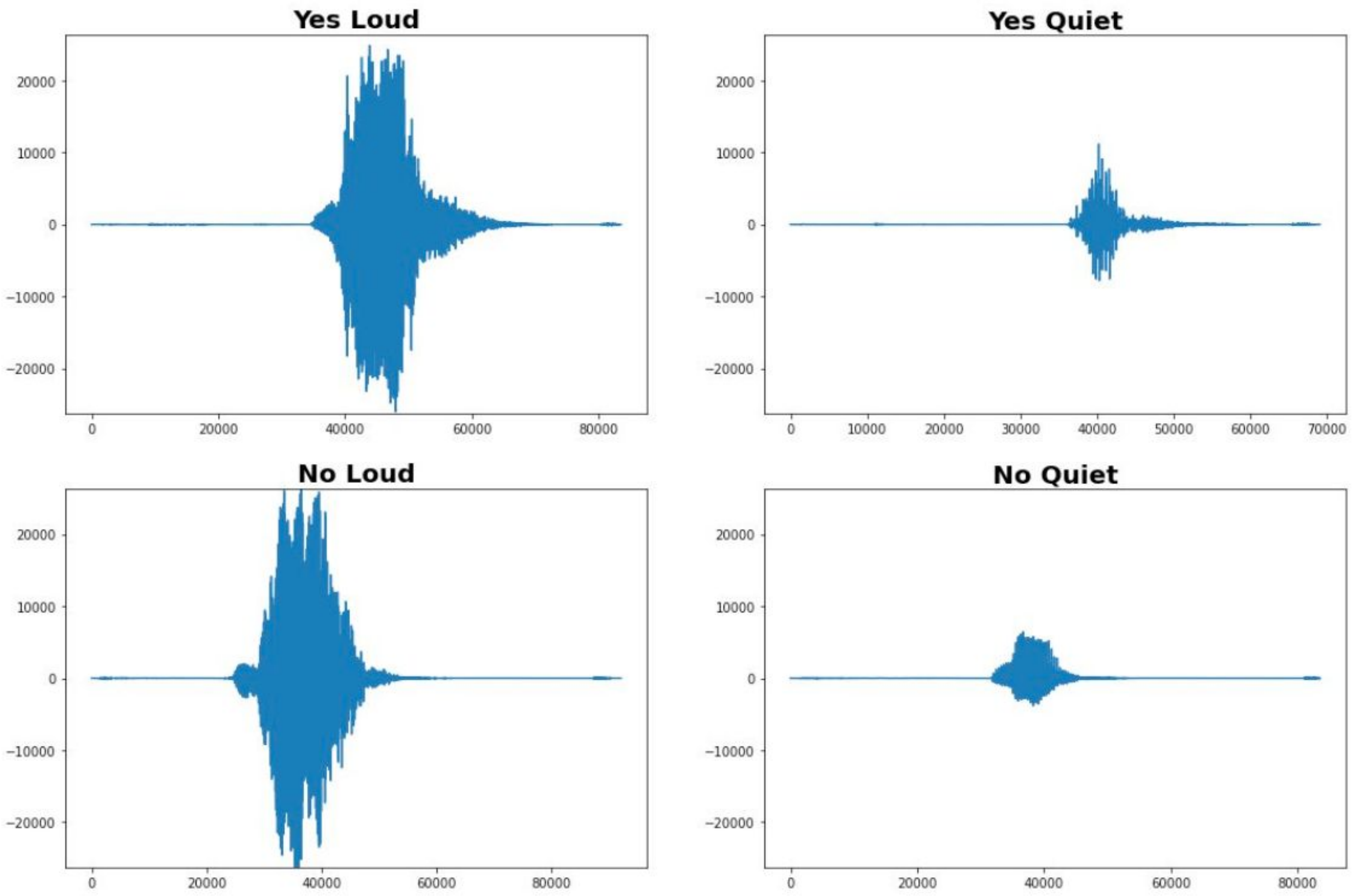
Data Preprocessing



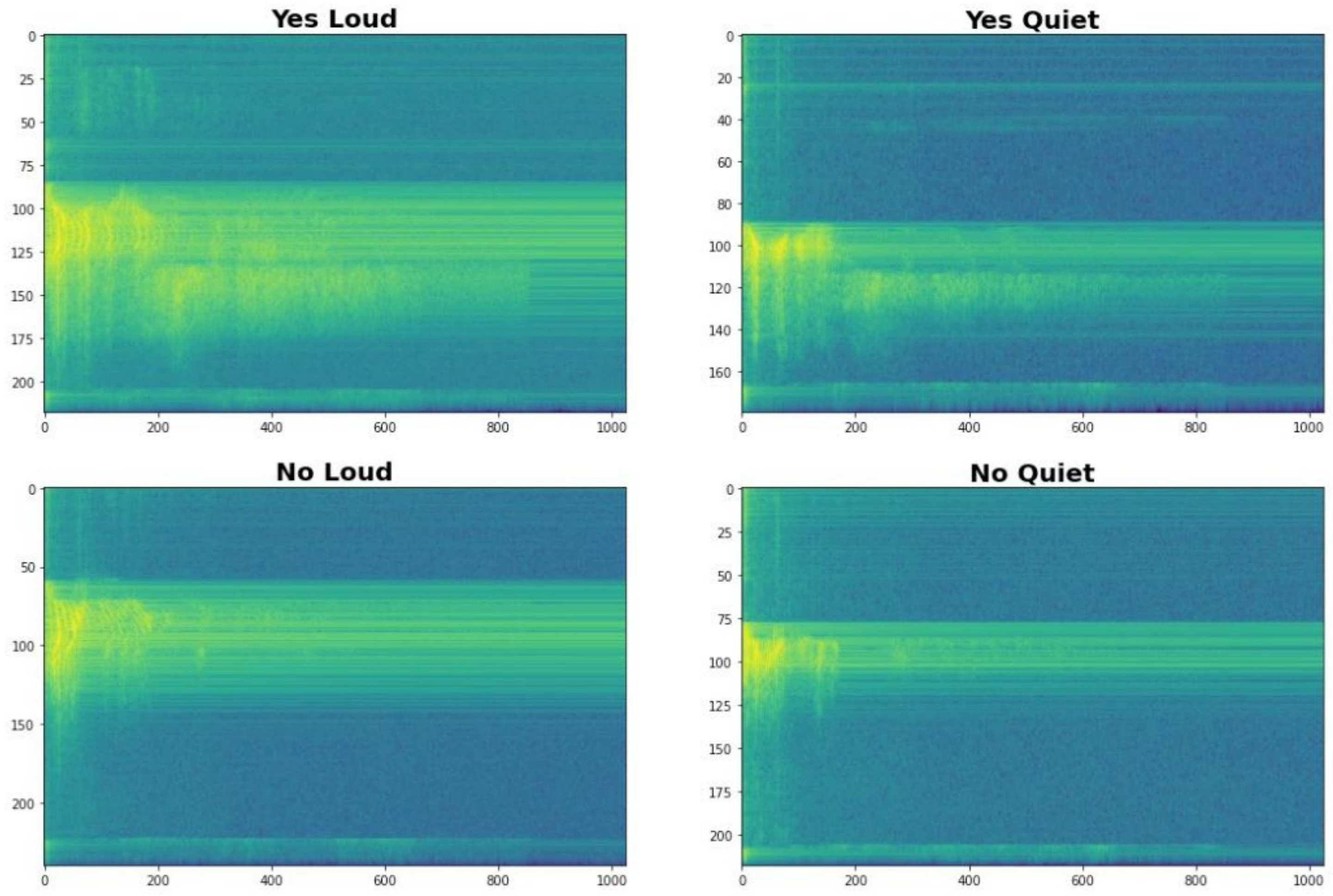
Data Preprocessing: **Spectrograms**



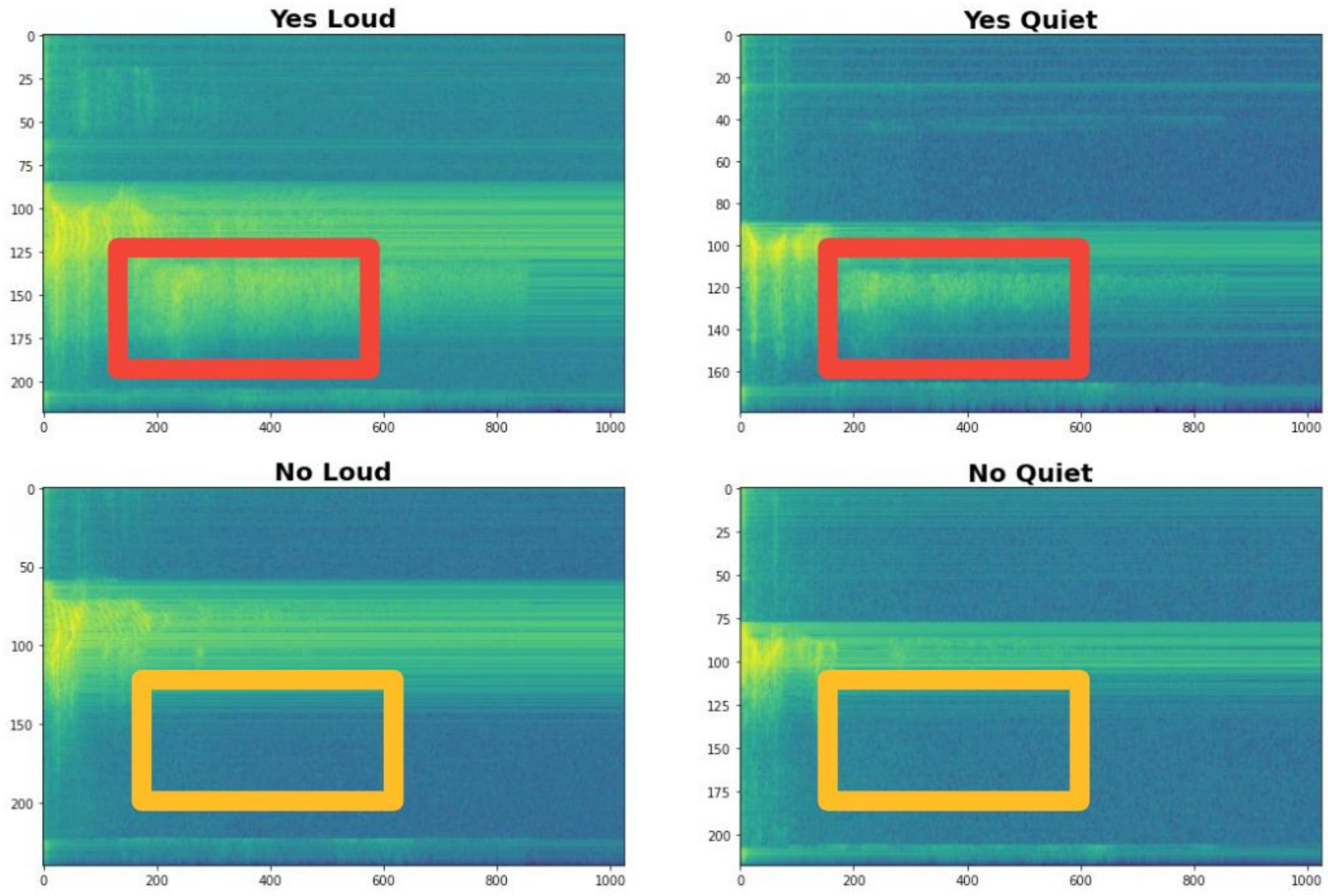
Data Preprocessing: **Spectrograms**

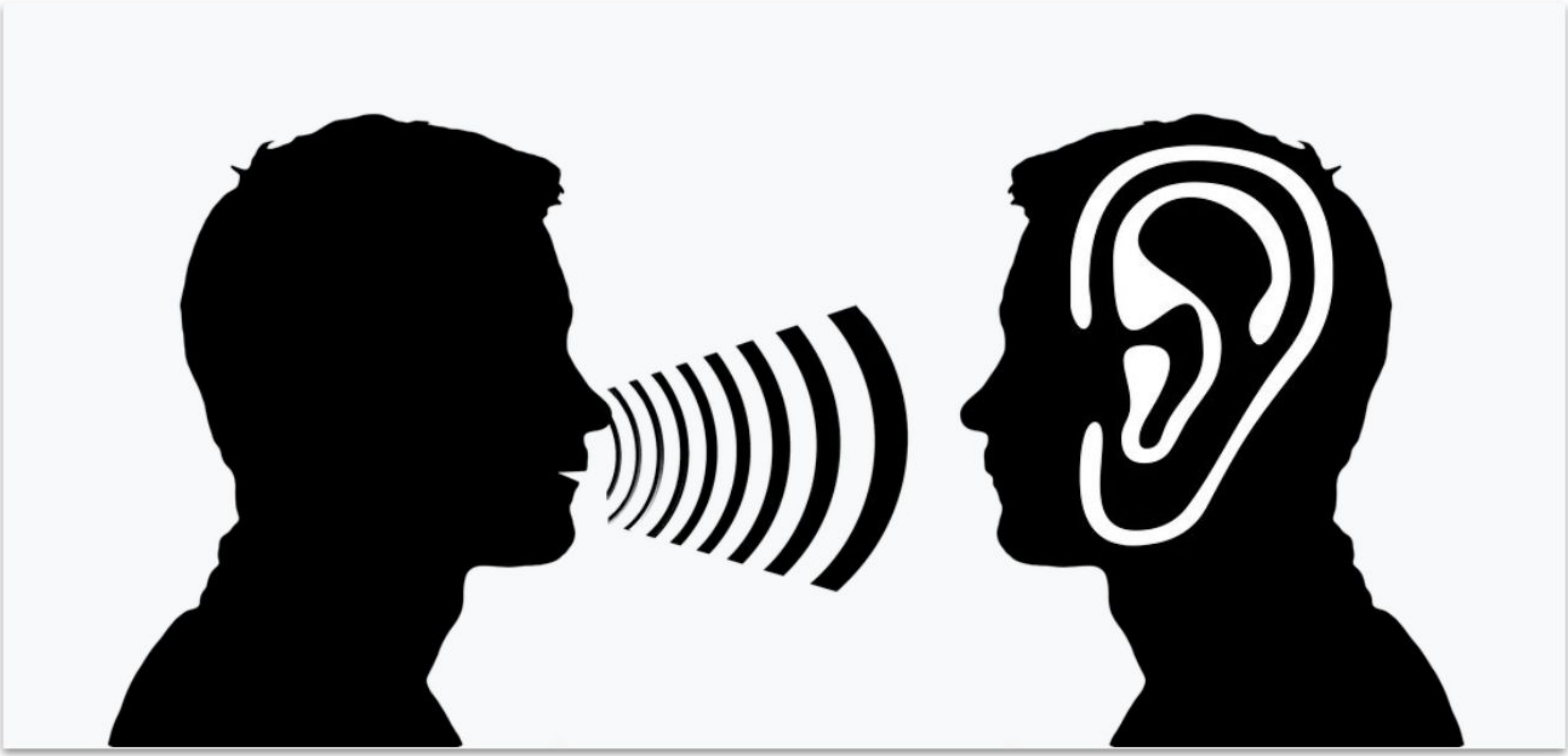


Data Preprocessing: **Spectrograms**



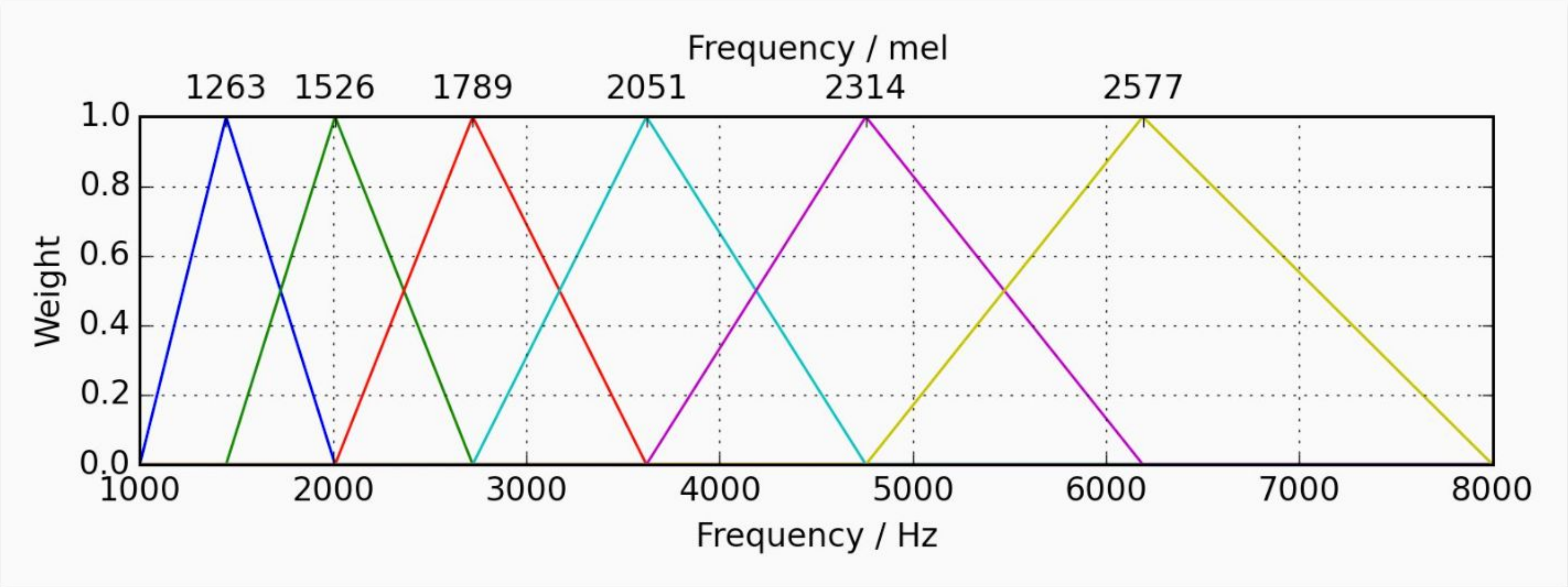
Data Preprocessing: **Spectrograms**



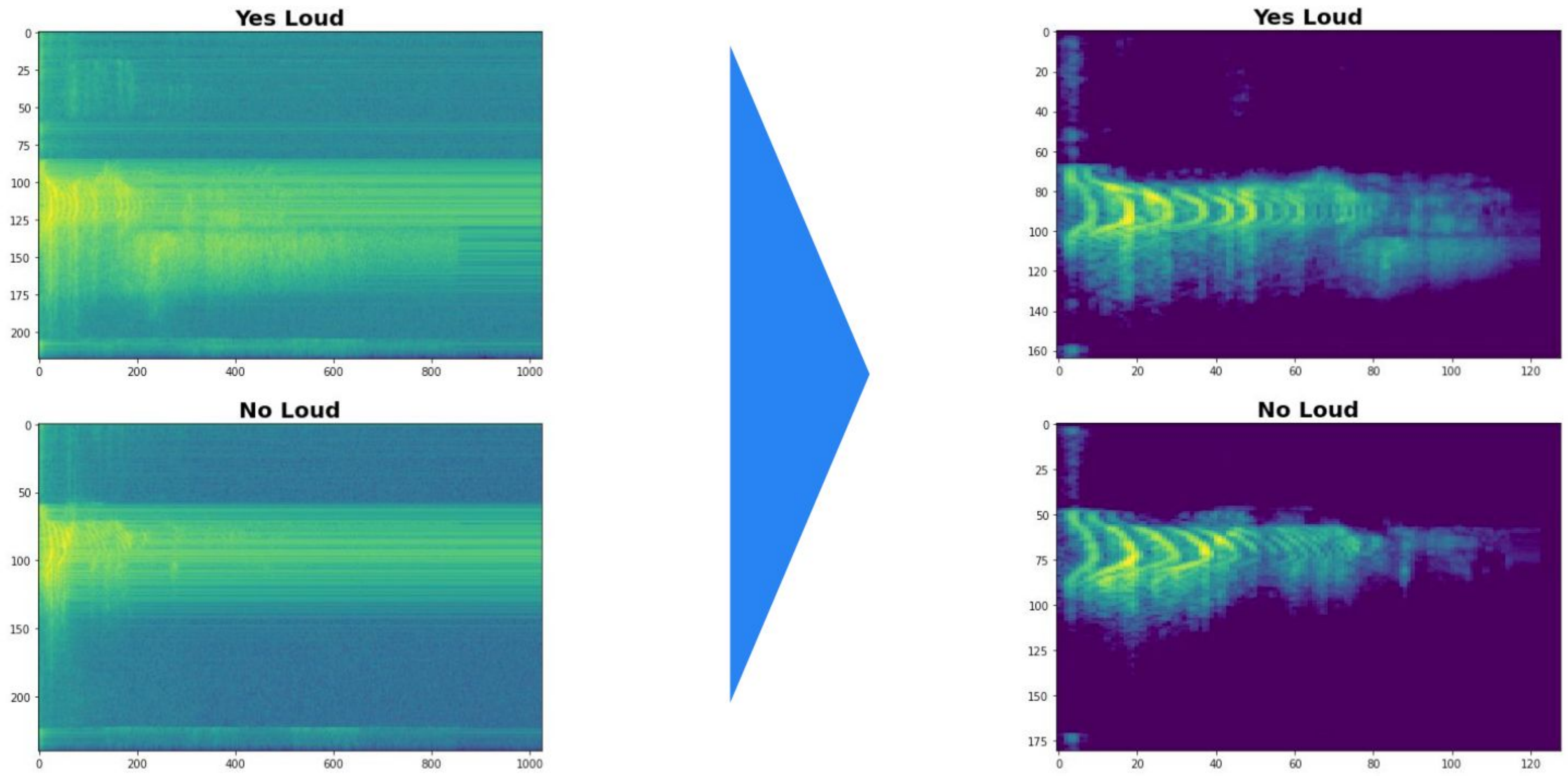


The lower band frequencies is much more crisper to us

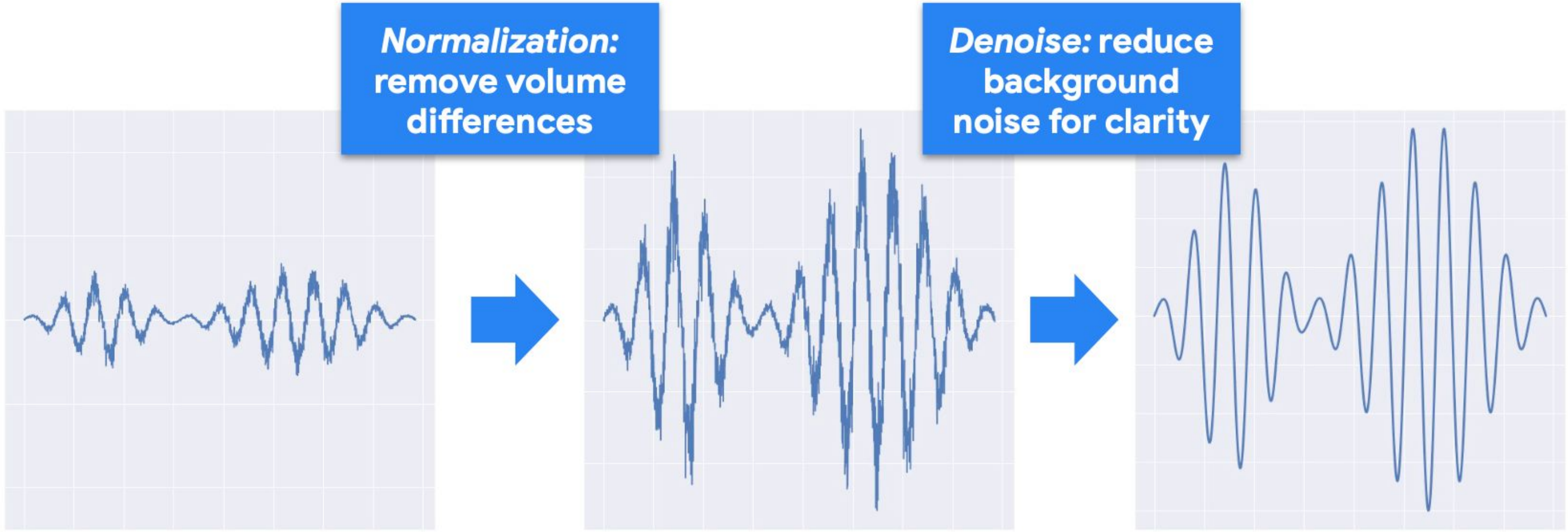
Mel Filterbanks



Spectrograms v. MFCCs



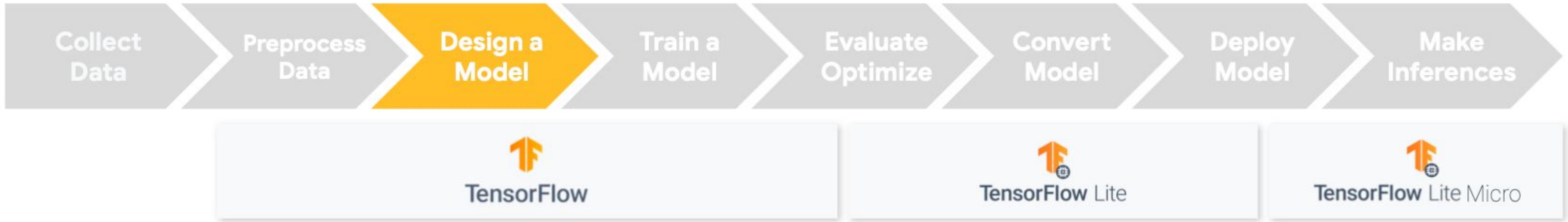
Additional **Feature Engineering**



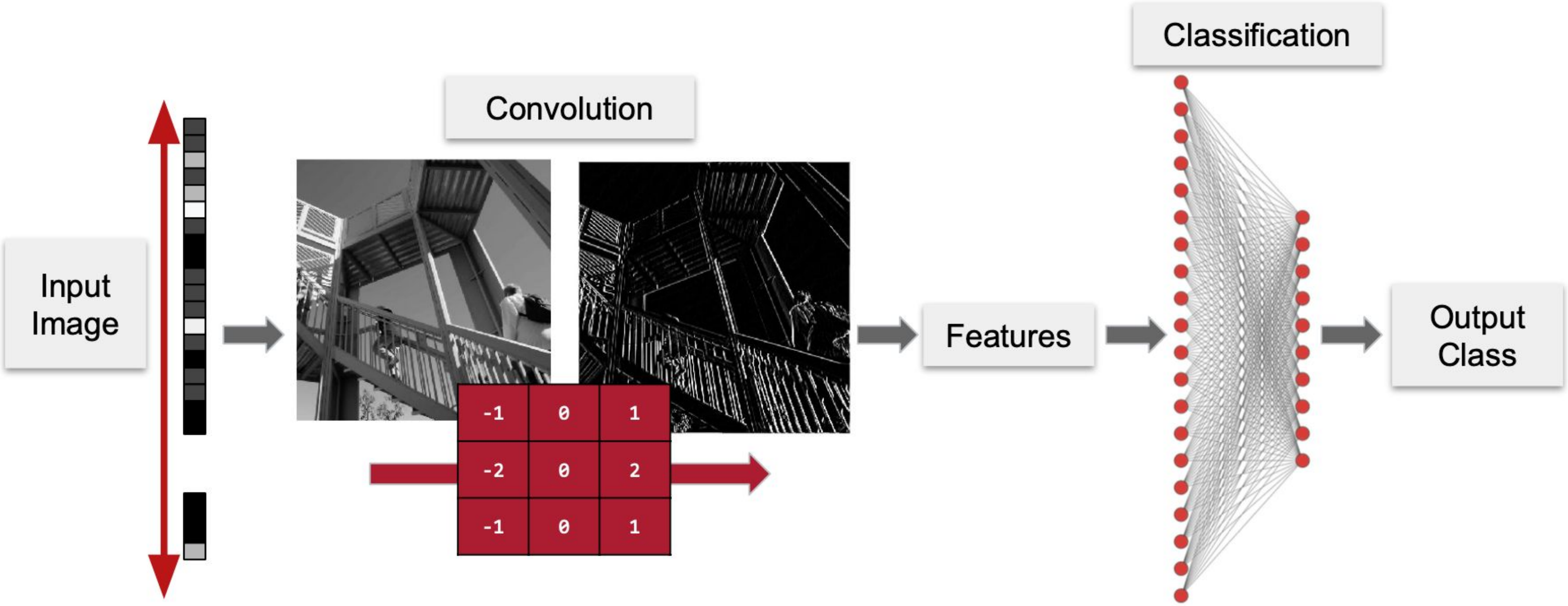
Spectrograms and MFCCs
Code Time!

SpectrogramsMFCCs.ipynb

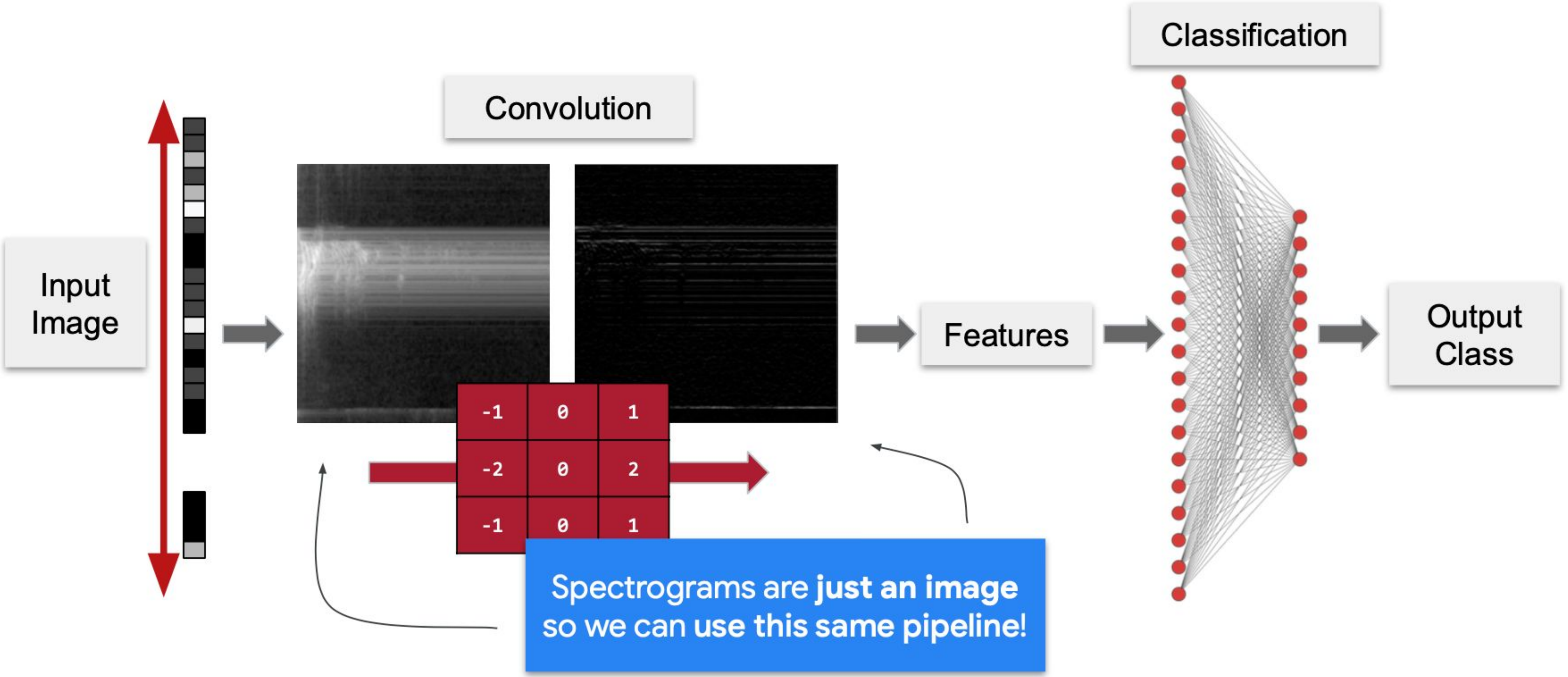
A Keyword Spotting Model


Collect Data
Preprocess Data
Design a Model
Train a Model
Evaluate Optimize
Convert Model
Deploy Model
Make Inferences
TensorFlow
TensorFlow Lite
TensorFlow Lite Micro

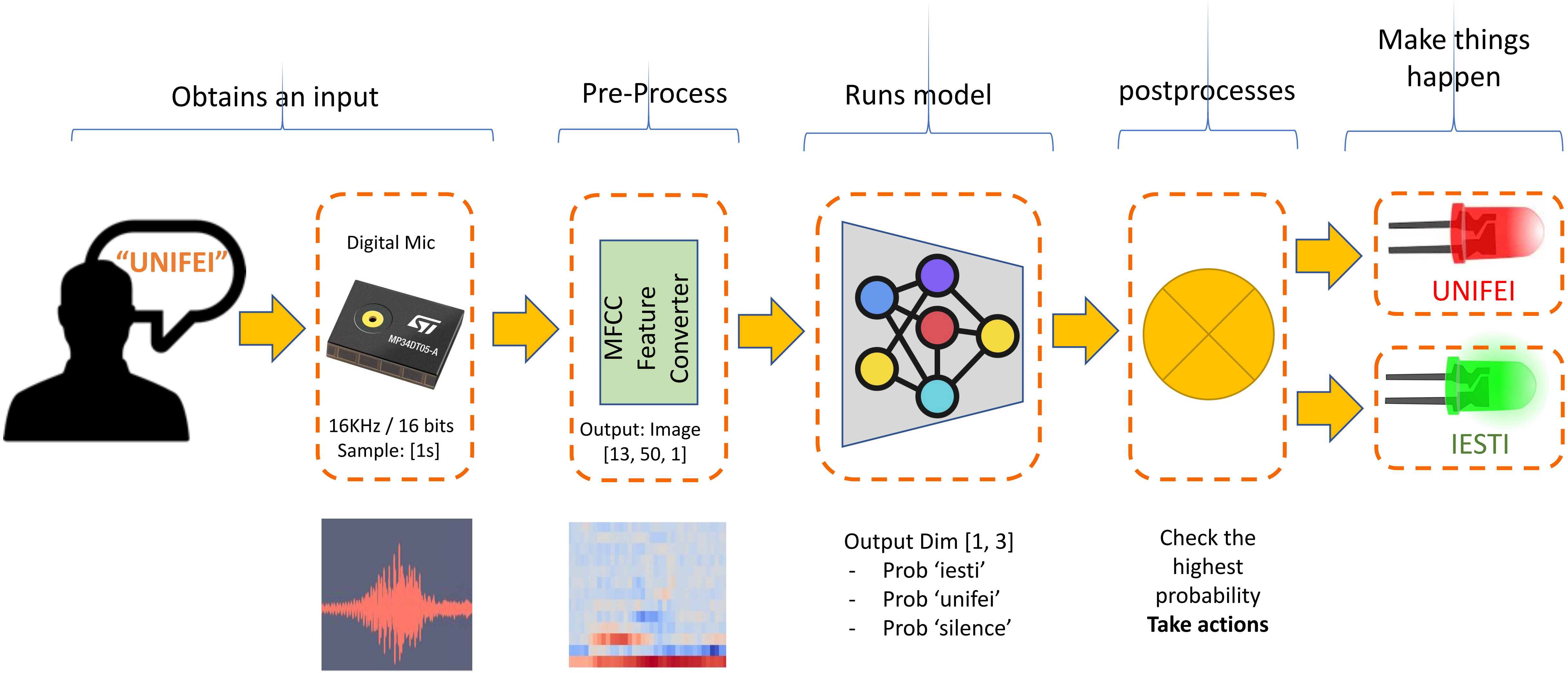
A model for **Keyword Spotting**



A model for **Keyword Spotting**

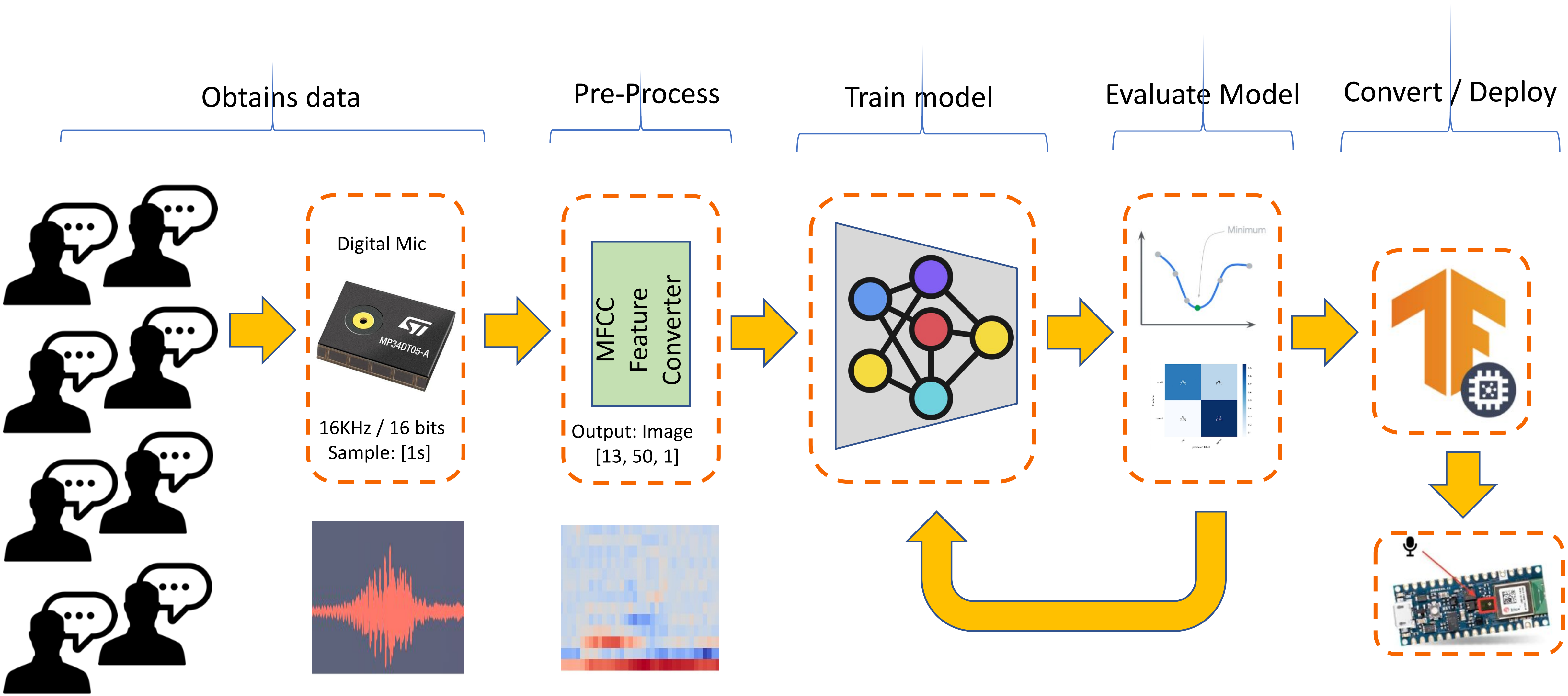


KeyWord Spotting (KWS) - **Inference**

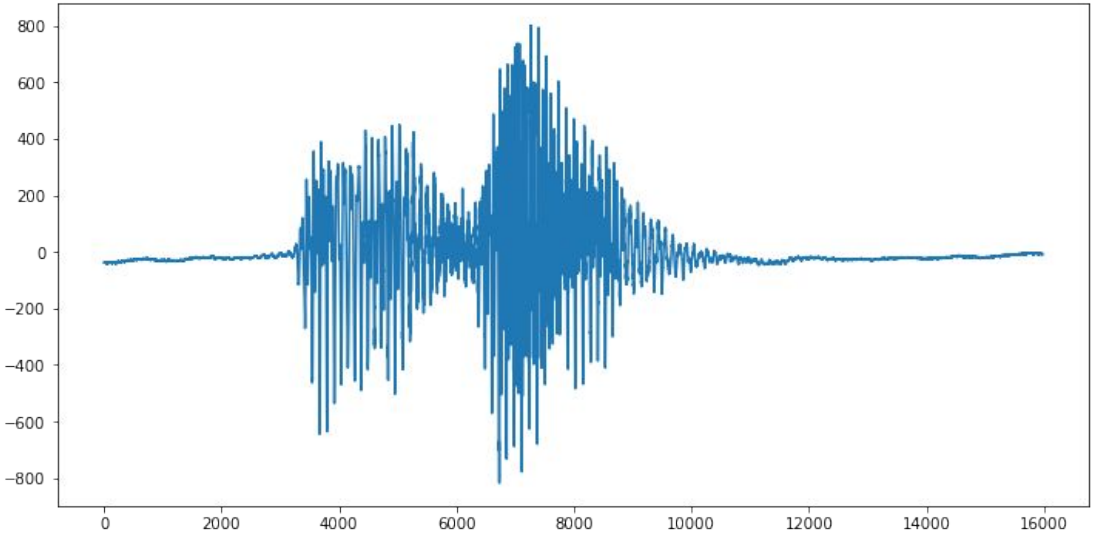


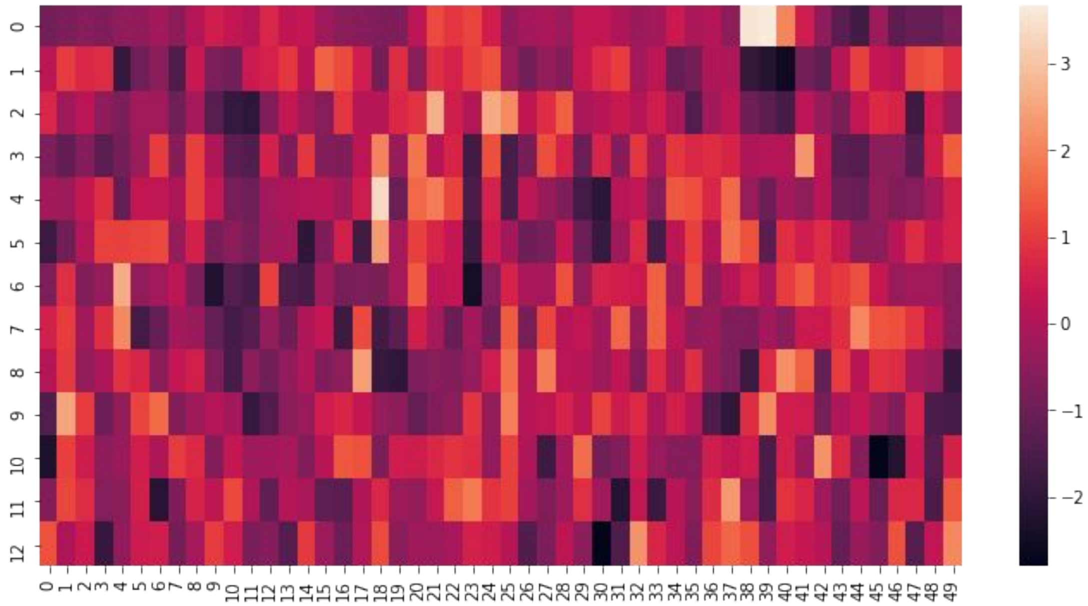
https://youtu.be/XnFYz-RSNe8

KeyWord Spotting (KWS) – Create Model (Training)



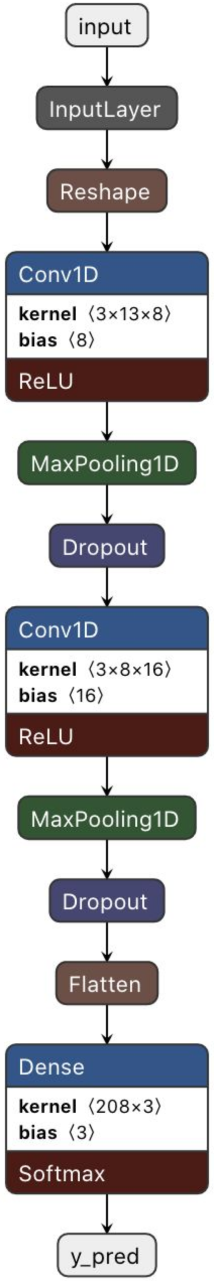
“UNIFEI”





13 x 50 = 650

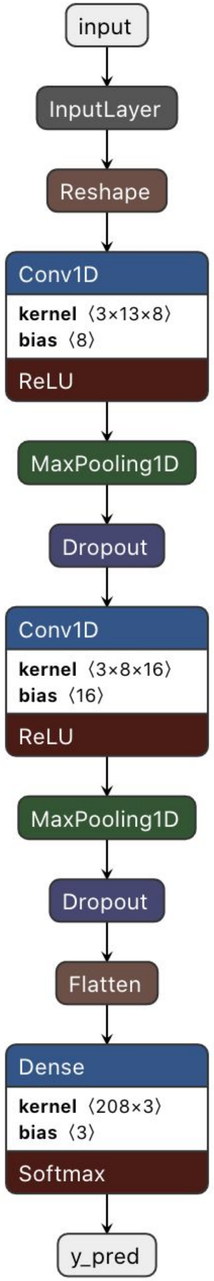
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 50, 13)            0
_________________________________________________________________
conv1d (Conv1D)              (None, 50, 8)             320
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 25, 8)             0
_________________________________________________________________
dropout (Dropout)            (None, 25, 8)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 25, 16)            400
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 13, 16)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 16)            0
_________________________________________________________________
flatten (Flatten)            (None, 208)               0
_________________________________________________________________
y_pred (Dense)               (None, 3)                 627
=================================================================
Total params: 1,347
Trainable params: 1,347
Non-trainable params: 0
_________________________________________________________________
```



Model size: 200KB

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
reshape (Reshape)            (None, 50, 13)            0
_________________________________________________________________
conv1d (Conv1D)              (None, 50, 8)             320
_________________________________________________________________
max_pooling1d (MaxPooling1D) (None, 25, 8)             0
_________________________________________________________________
dropout (Dropout)            (None, 25, 8)             0
_________________________________________________________________
conv1d_1 (Conv1D)            (None, 25, 16)            400
_________________________________________________________________
max_pooling1d_1 (MaxPooling1 (None, 13, 16)            0
_________________________________________________________________
dropout_1 (Dropout)          (None, 13, 16)            0
_________________________________________________________________
flatten (Flatten)            (None, 208)               0
_________________________________________________________________
y_pred (Dense)               (None, 3)                 627
=================================================================
Total params: 1,347
Trainable params: 1,347
Non-trainable params: 0
_________________________________________________________________
```

Our board **Arduino Nano-33**
has **256KB** of RAM (memory)



```python
# Save the model to disk
model.save('cnn_v1_saved_model')
```

executed in 882ms, finished 17:53:28 2021-07-05

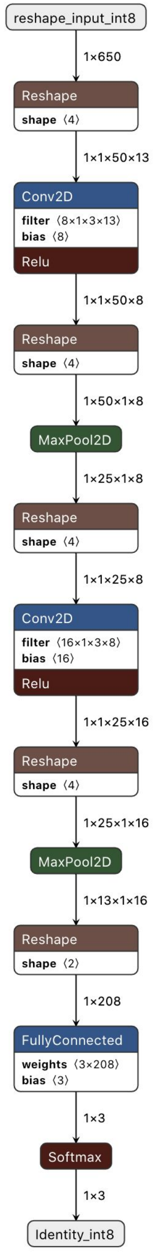
```python
# Convert TF model to a tflite model
from tensorflow.keras.models import load_model

model_cnn_v1 = load_model('cnn_v1_saved_model')
converter = tf.lite.TFLiteConverter.from_keras_model(model_cnn_v1)
converter.optimizations = [tf.lite.Optimize.DEFAULT]
tflite_model = converter.convert()

tflite_model_size = open("cnn_v1.tflite","wb").write(tflite_model)
print("Quantized model (DEFAULT) is {:,} bytes".format(tflite_model_size))
```

```
Quantized model (DEFAULT) is 11,536 bytes
```

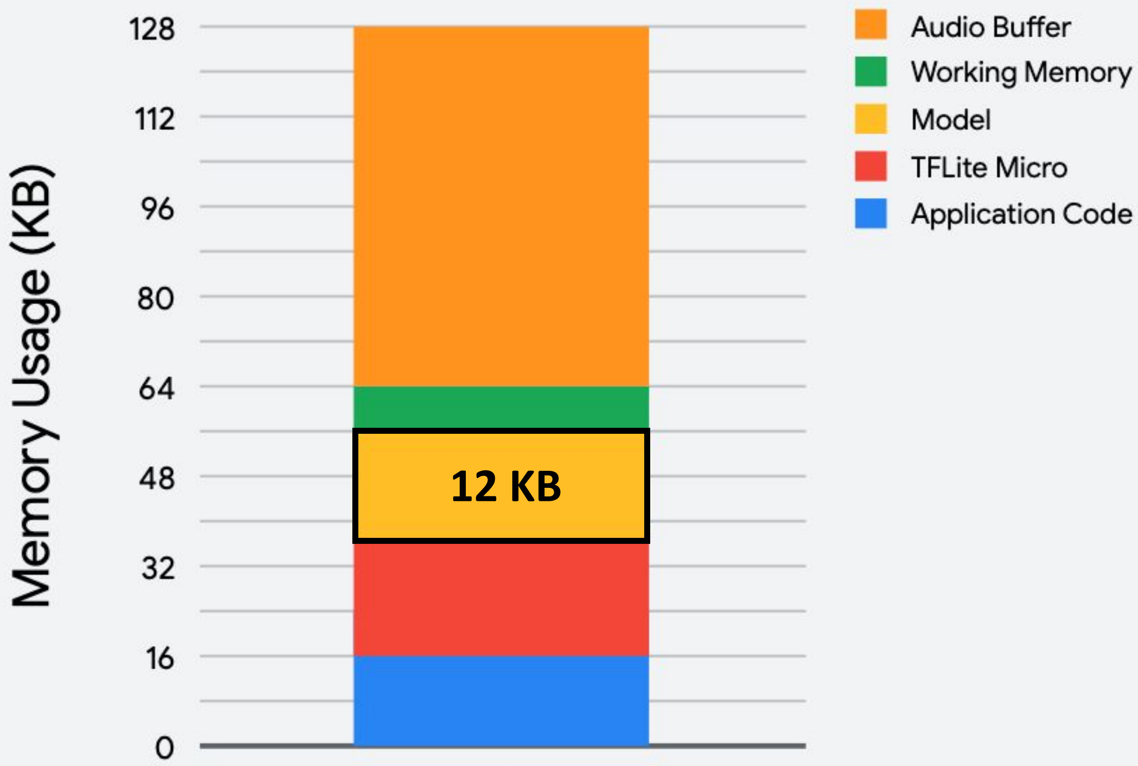
Our board **Arduino Nano-33** has **256KB** of RAM (memory)



Memory Usage

- Need to be **resource aware**
- **Less** compute
- **Less** memory
- Use **quantization**

Our board **Arduino Nano-33** has **256KB** of RAM (memory)



Reading Material

Main references

- Harvard School of Engineering and Applied Sciences - CS249r: Tiny Machine Learning
- Professional Certificate in Tiny Machine Learning (TinyML) – edX/Harvard
- Introduction to Embedded Machine Learning - Coursera/Edge Impulse
- Computer Vision with Embedded Machine Learning - Coursera/Edge Impulse
- Fundamentals textbook: “Deep Learning with Python” by François Chollet
- Applications & Deploy textbook: “TinyML” by Pete Warden, Daniel Situnayake
- Deploy textbook “TinyML Cookbook” by Gian Marco Iodice

I want to thank Shawn Hymel and Edge Impulse, Pete Warden and Laurence Moroney from Google, Professor Vijay Janapa Reddi and Brian Plancher from Harvard, and the rest of the TinyMLedu team for preparing the excellent material on TinyML that is the basis of this course at UNIFEI.

The IESTI01 course is part of the **TinyML4D**, an initiative to make TinyML education available to everyone globally.

Thanks

UNIFEI