POLITECNICO
MILANO 1863

Hardware Architectures for Embedded and Edge AI

Prof Manuel Roveri – manuel.roveri@polimi.it
Massimo Pavan – massimo.pavan@polimi.it

Exercise session 5 – Keyword spotting training

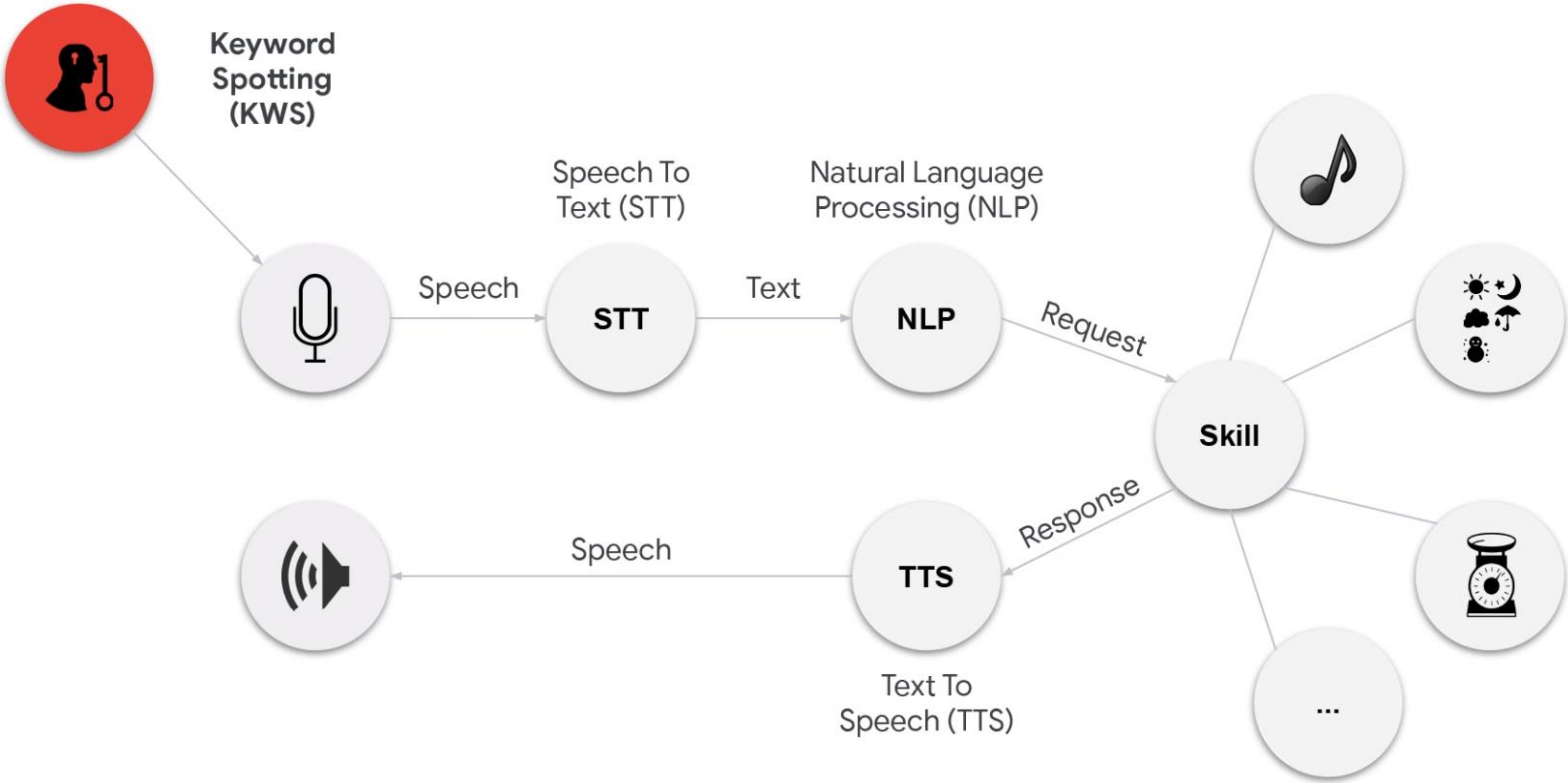
What's keyword spotting?

- Keyword spotting is one of the most successful examples of TinyML
  - Low-power, continuous, on-device
  - Started with english, expanded to many more languages (on-going process)

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

The application

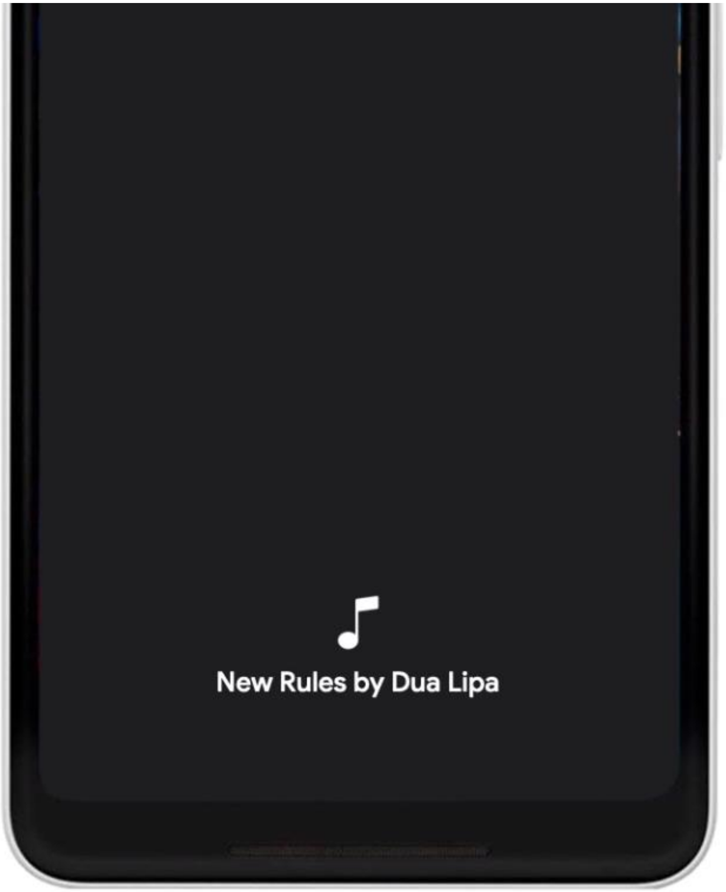
Cascade infrastracture!



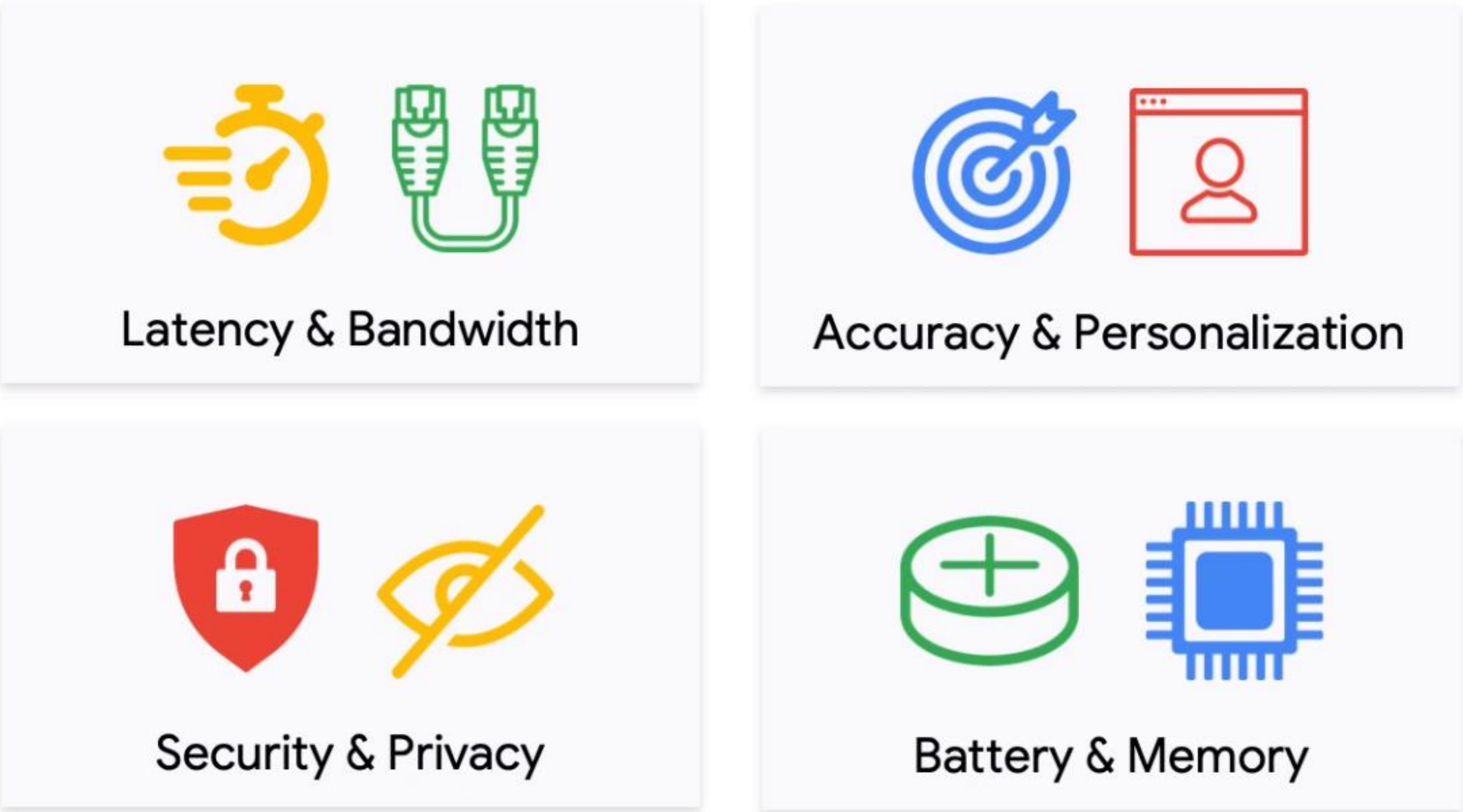
Interesting application with audio?

What else can **TinyML** do with audio data?

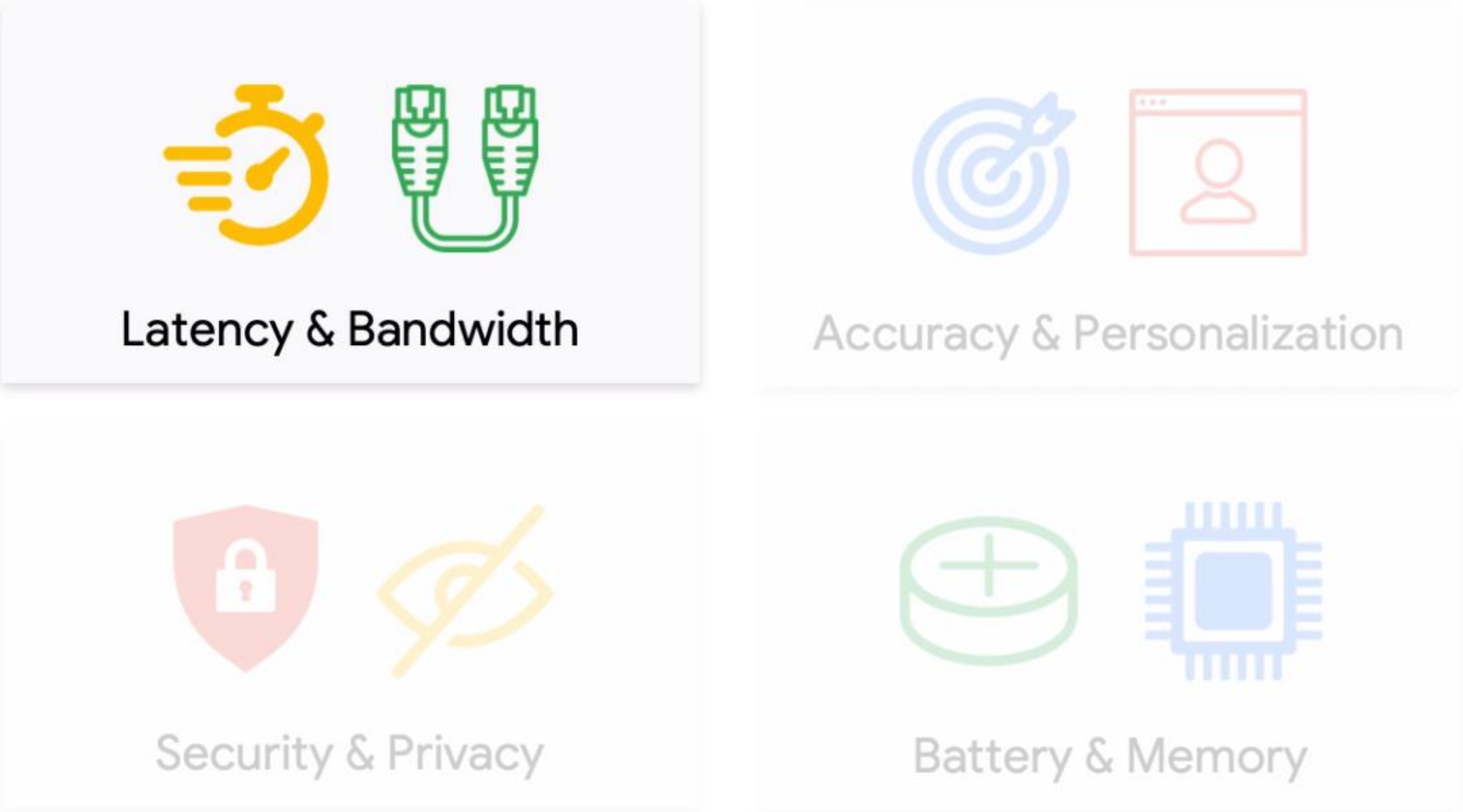
Now Playing
by Google



Callenges in wake-word detection



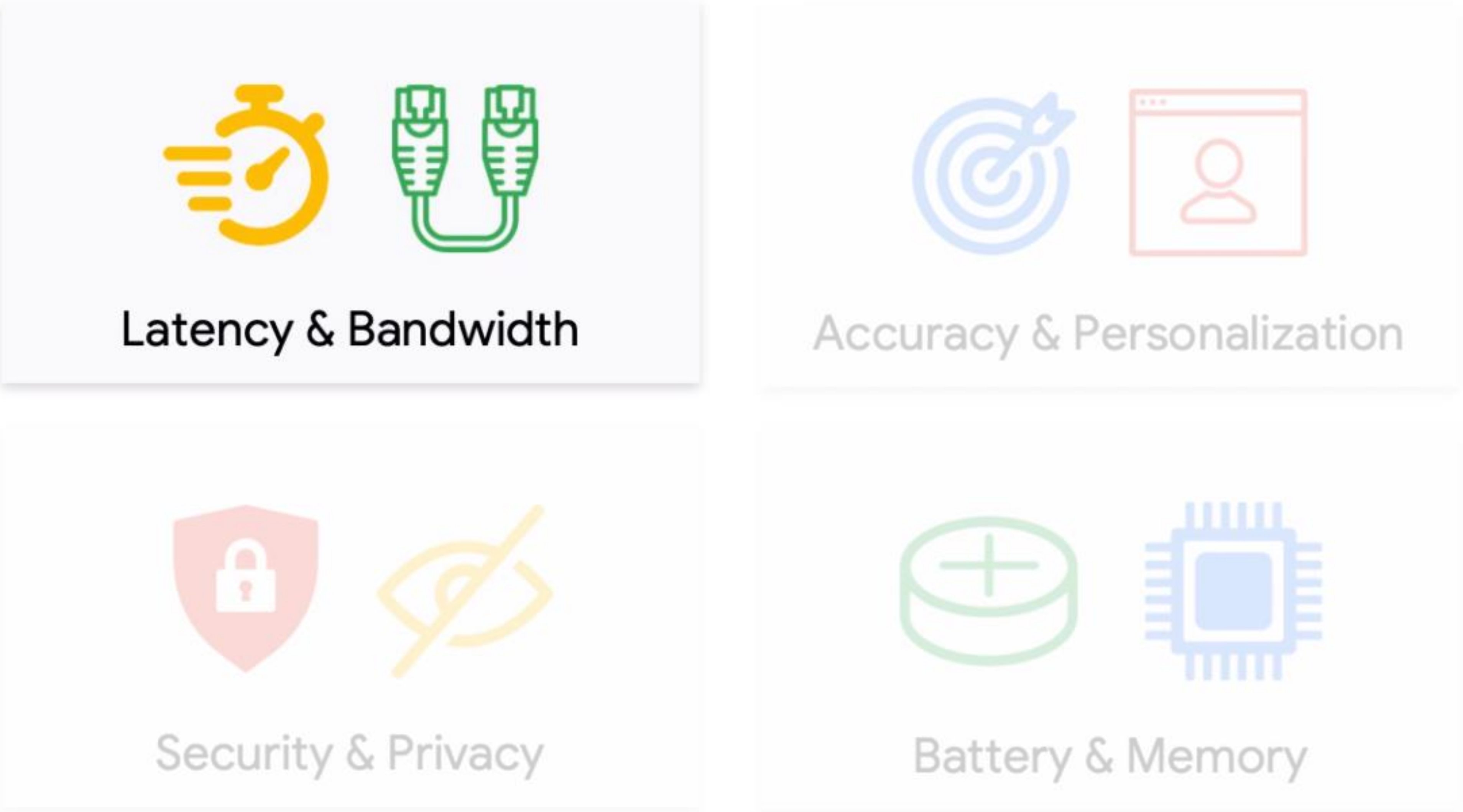
Callenges in wake-word detection



Latency

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

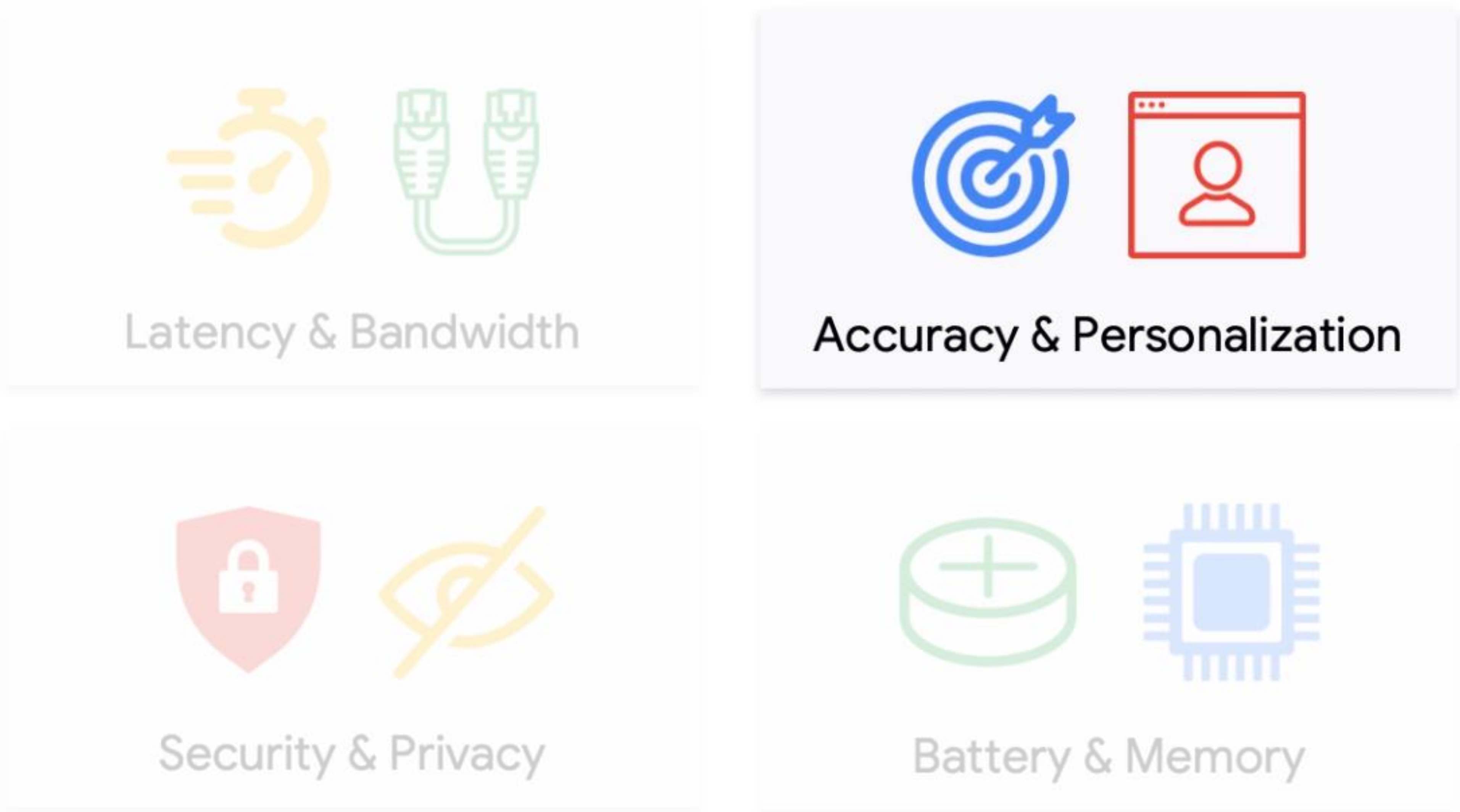
Callenges in wake-word detection



Bandwidth

Minimize data sent over the network (slow and expensive)

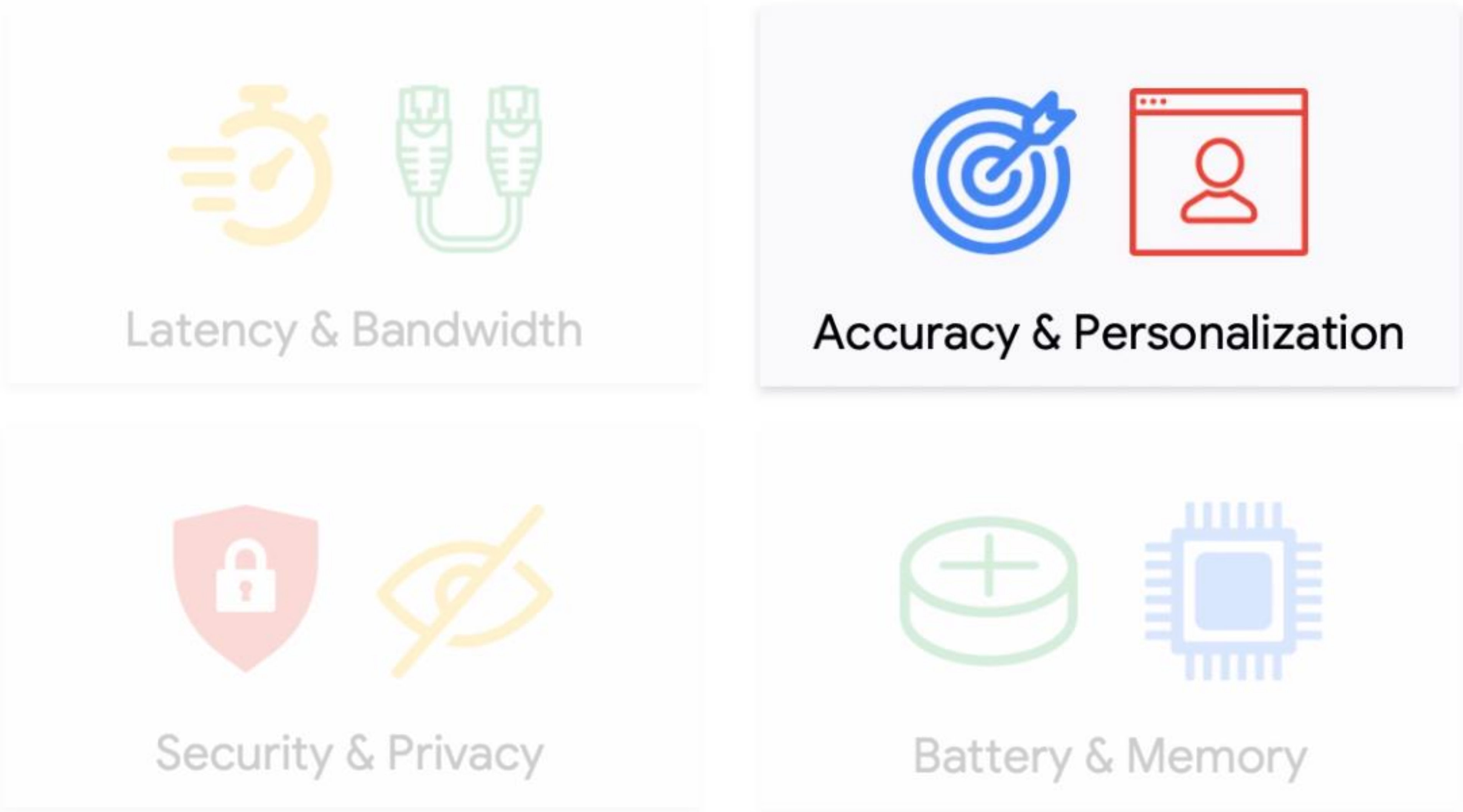
Callenges in wake-word detection



Accuracy

Listen continuously, but only trigger at the right time

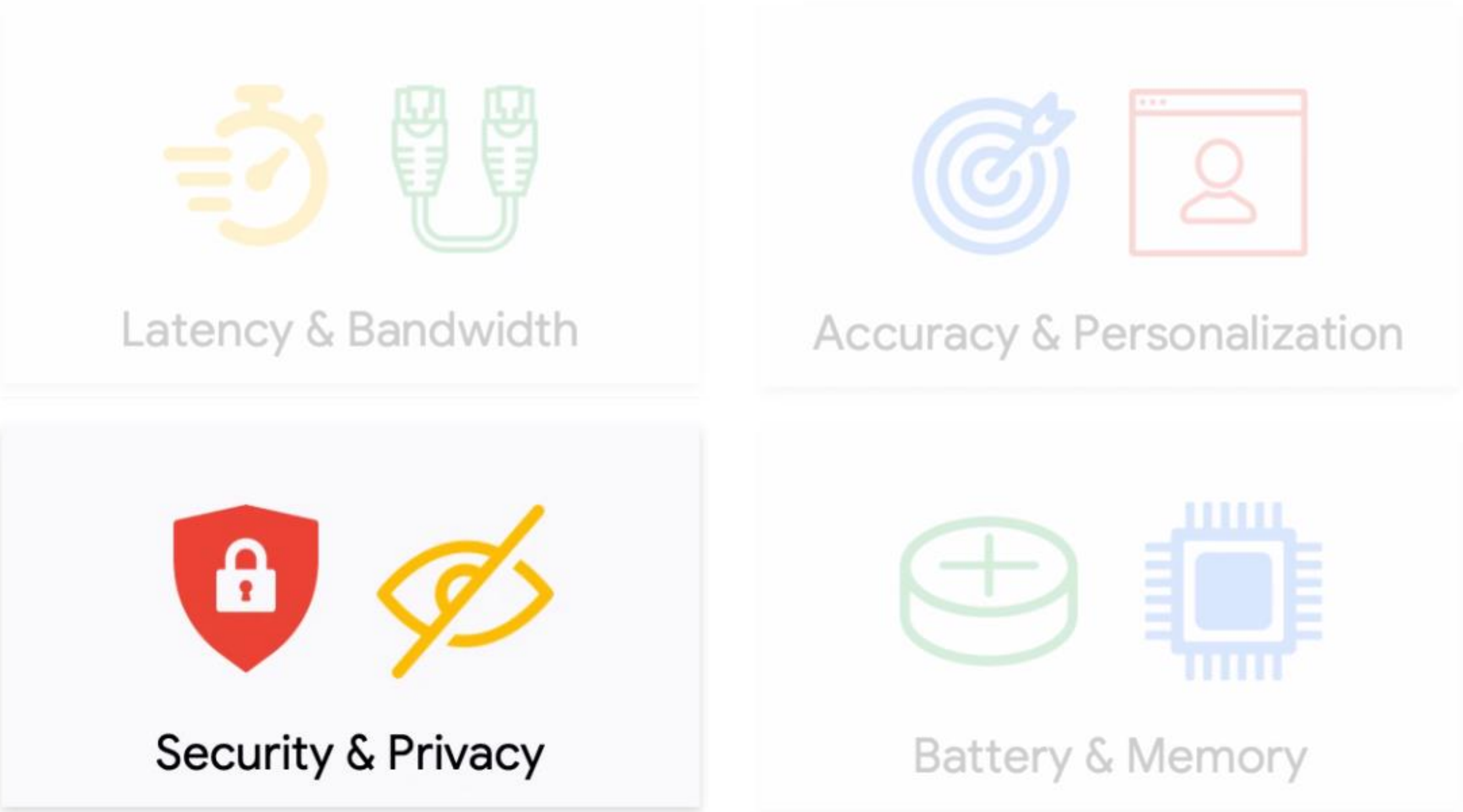
Callenges in wake-word detection



Personalization

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

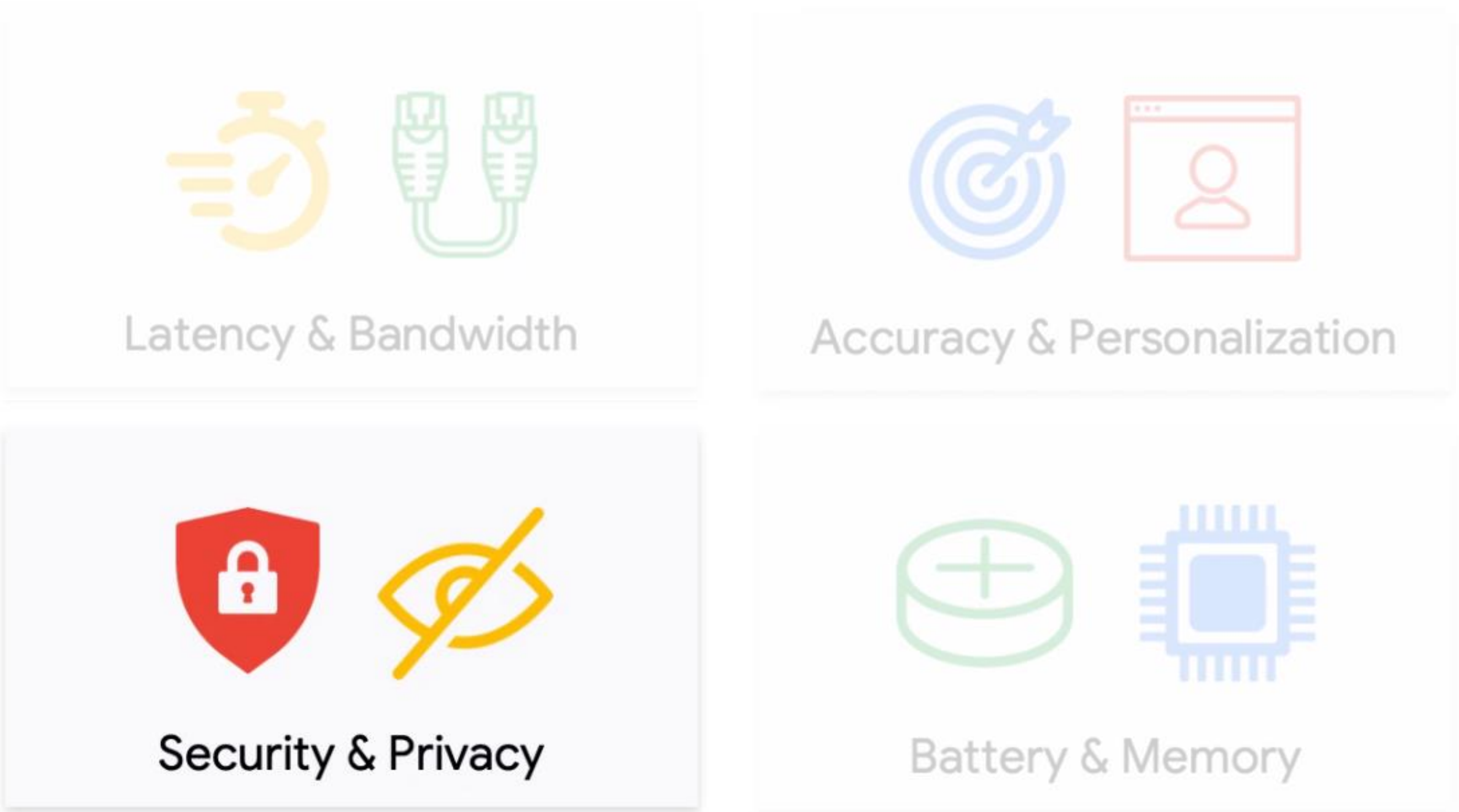
Callenges in wake-word detection



Security

Safeguarding the data that is being sent to the cloud (from a malevolous actor)

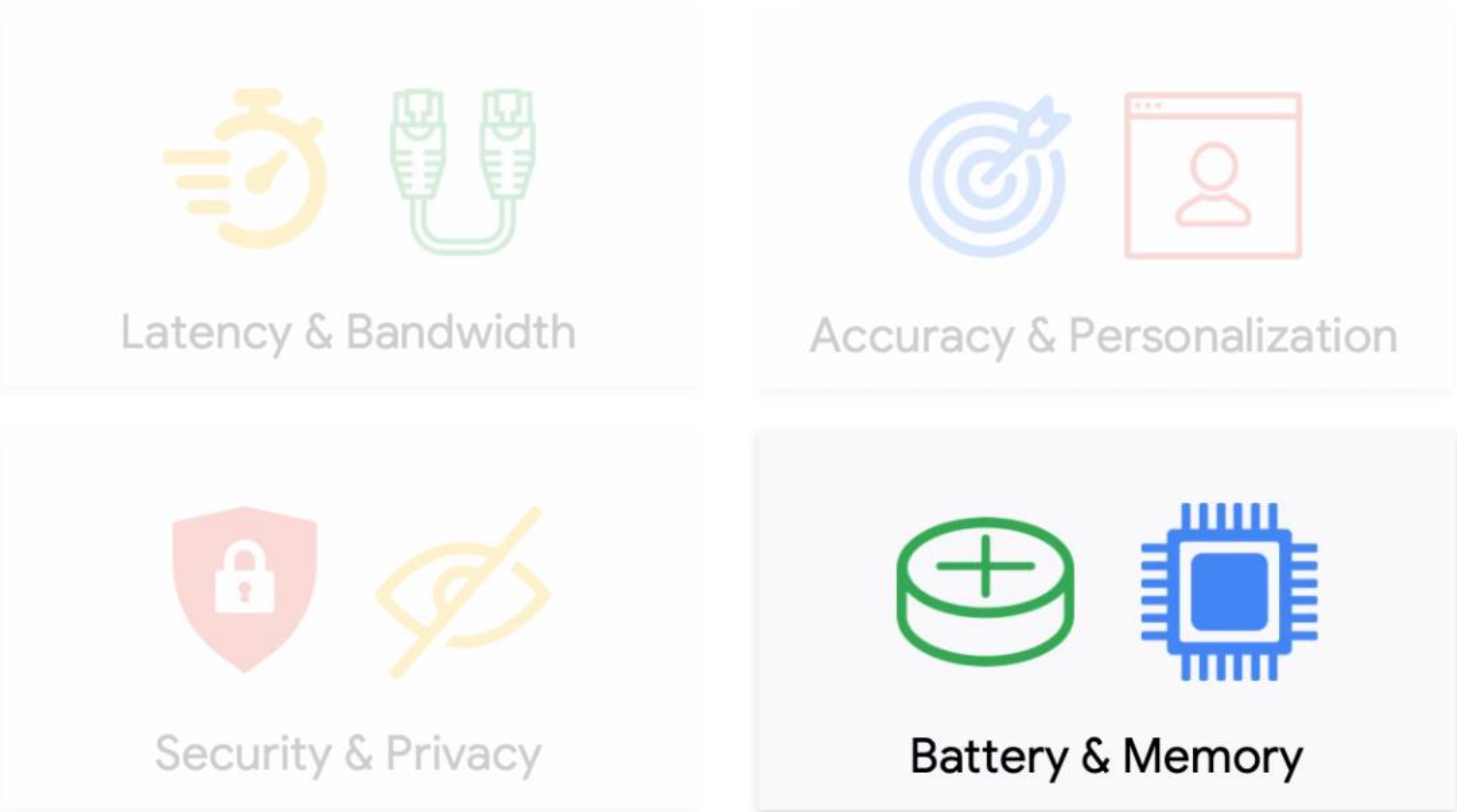
Callenges in wake-word detection



Privacy

Safeguarding the data that is being sent to the cloud (from anyone)

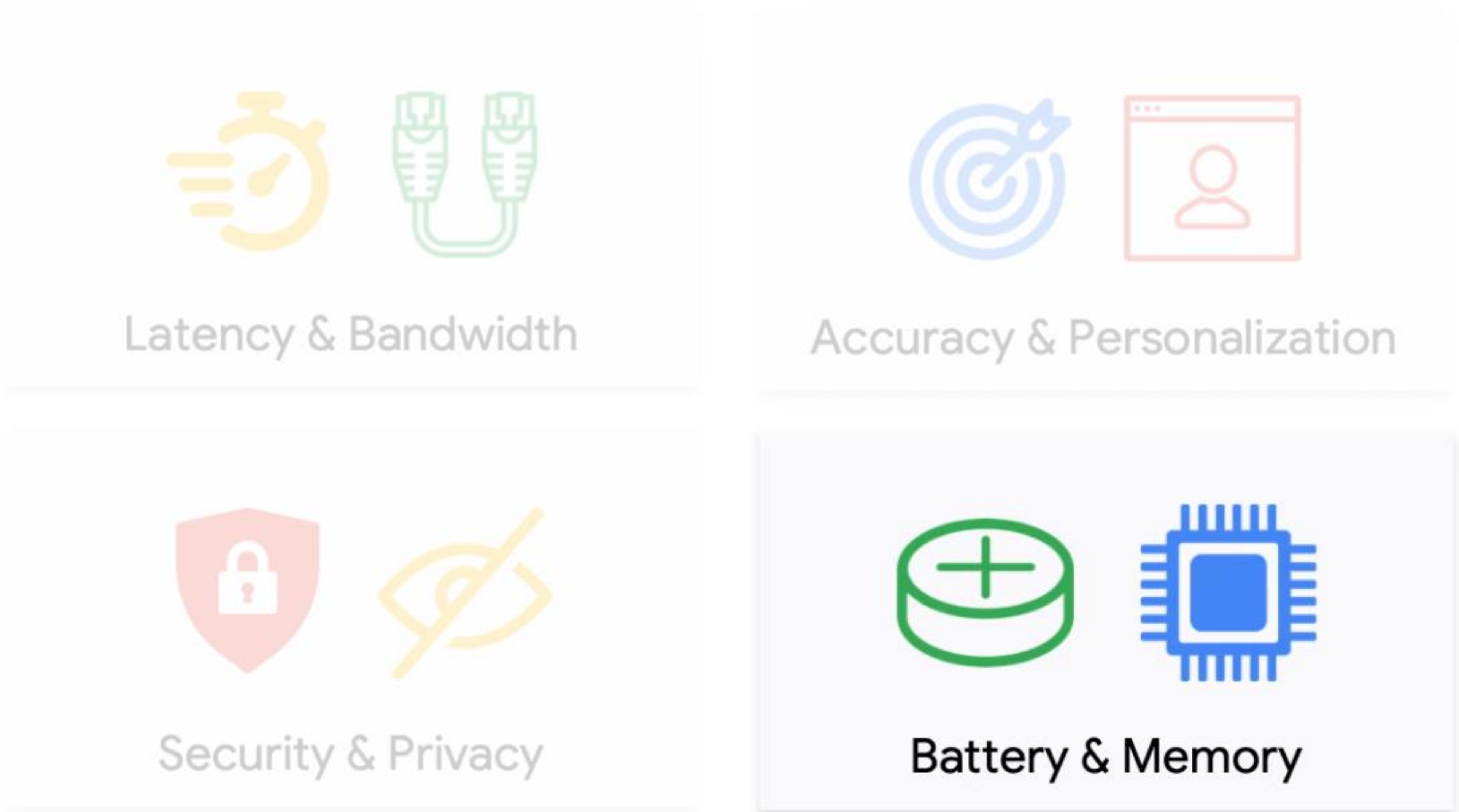
Callenges in wake-word detection



Battery

Limited energy, operate on coin-cell type batteries

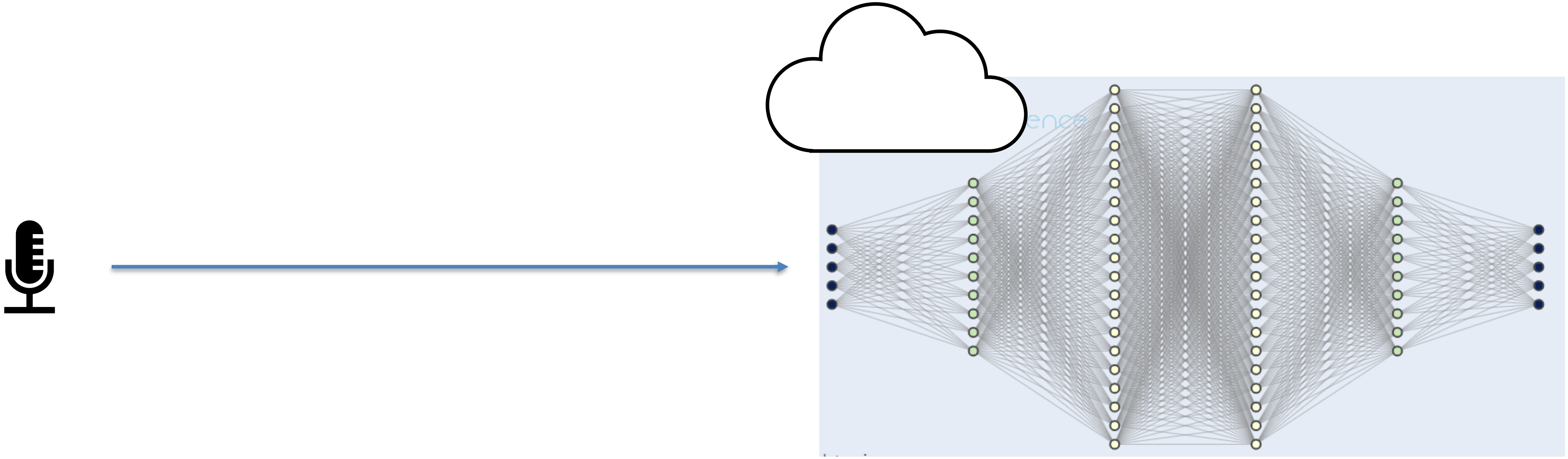
Callenges in wake-word detection



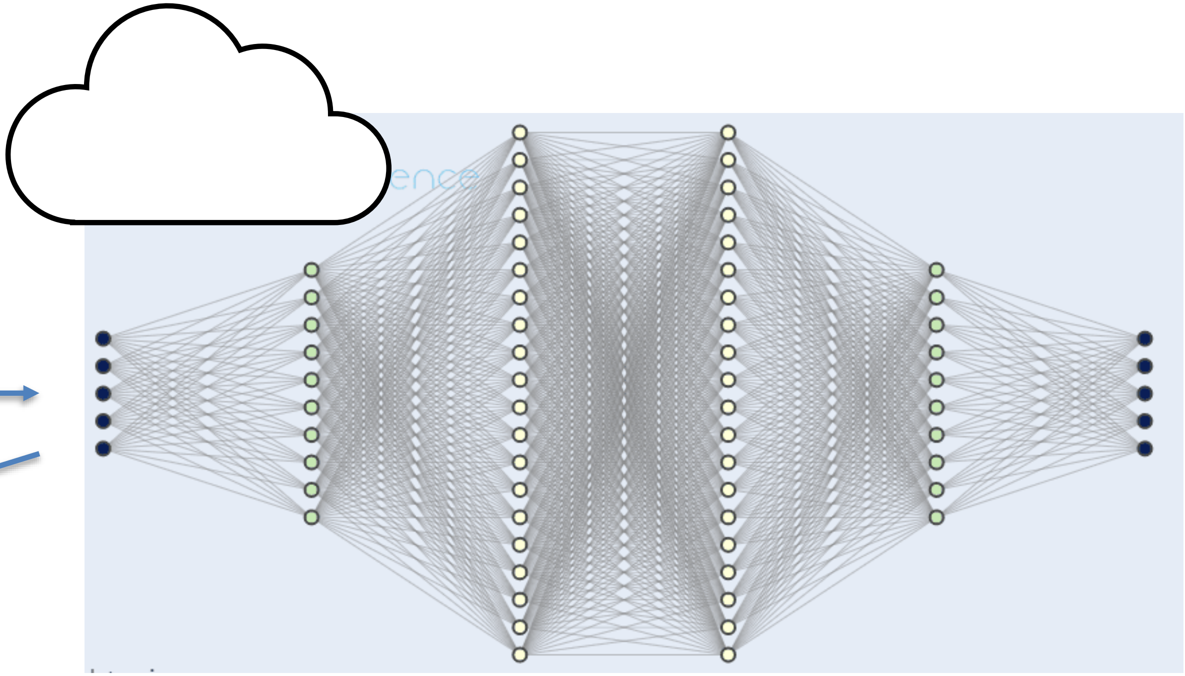
Memory

Run on resource constrained devices

Why do we need keyword spotting?



Why do we need keyword spotting?

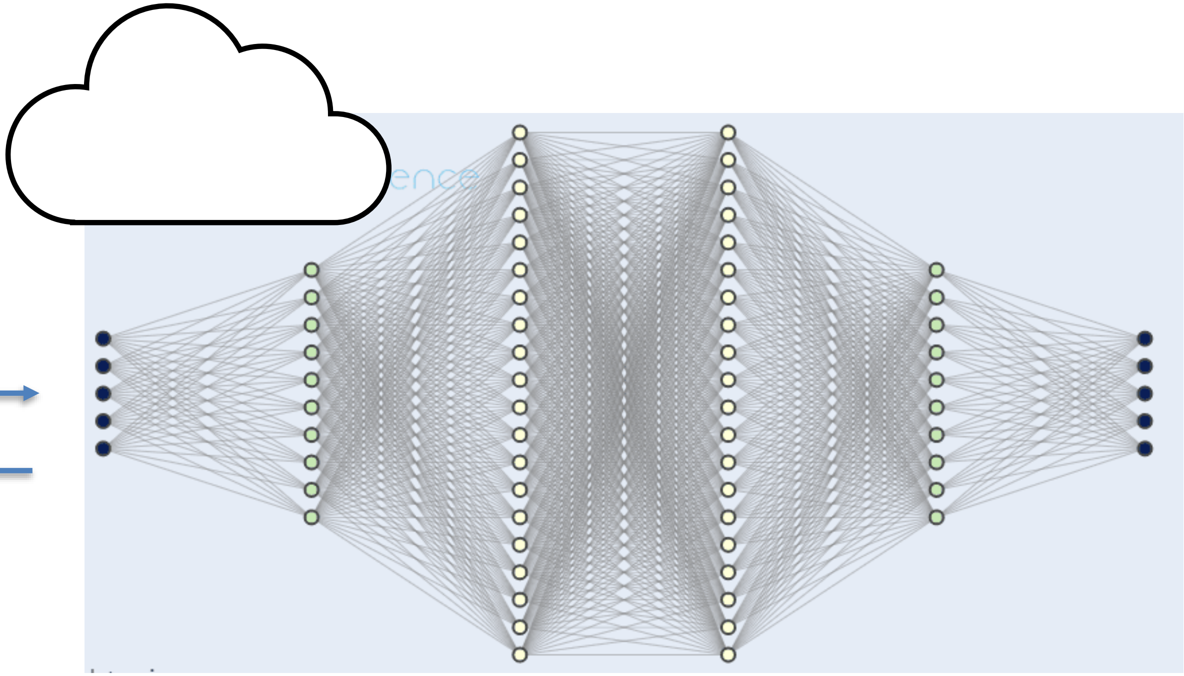


Play response

Need #1: **Latency**

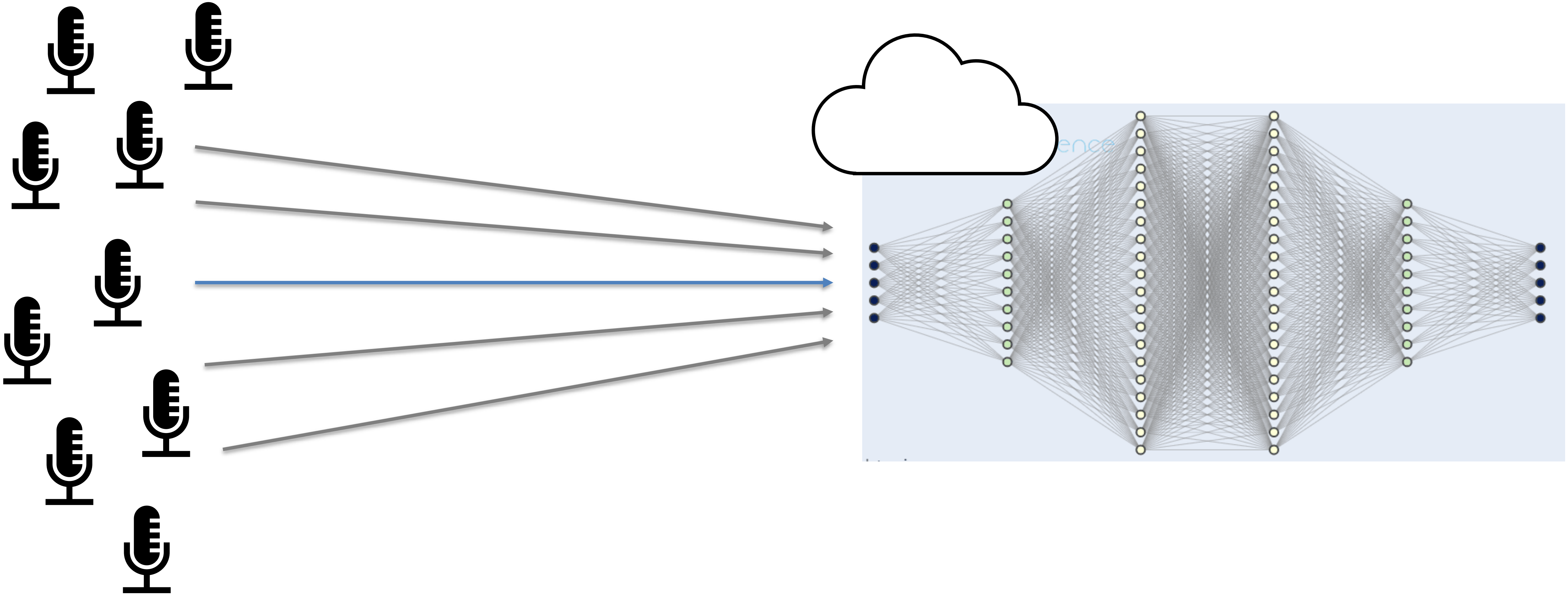
Why do we need keyword spotting?

1 – KS detection:
visual feedback

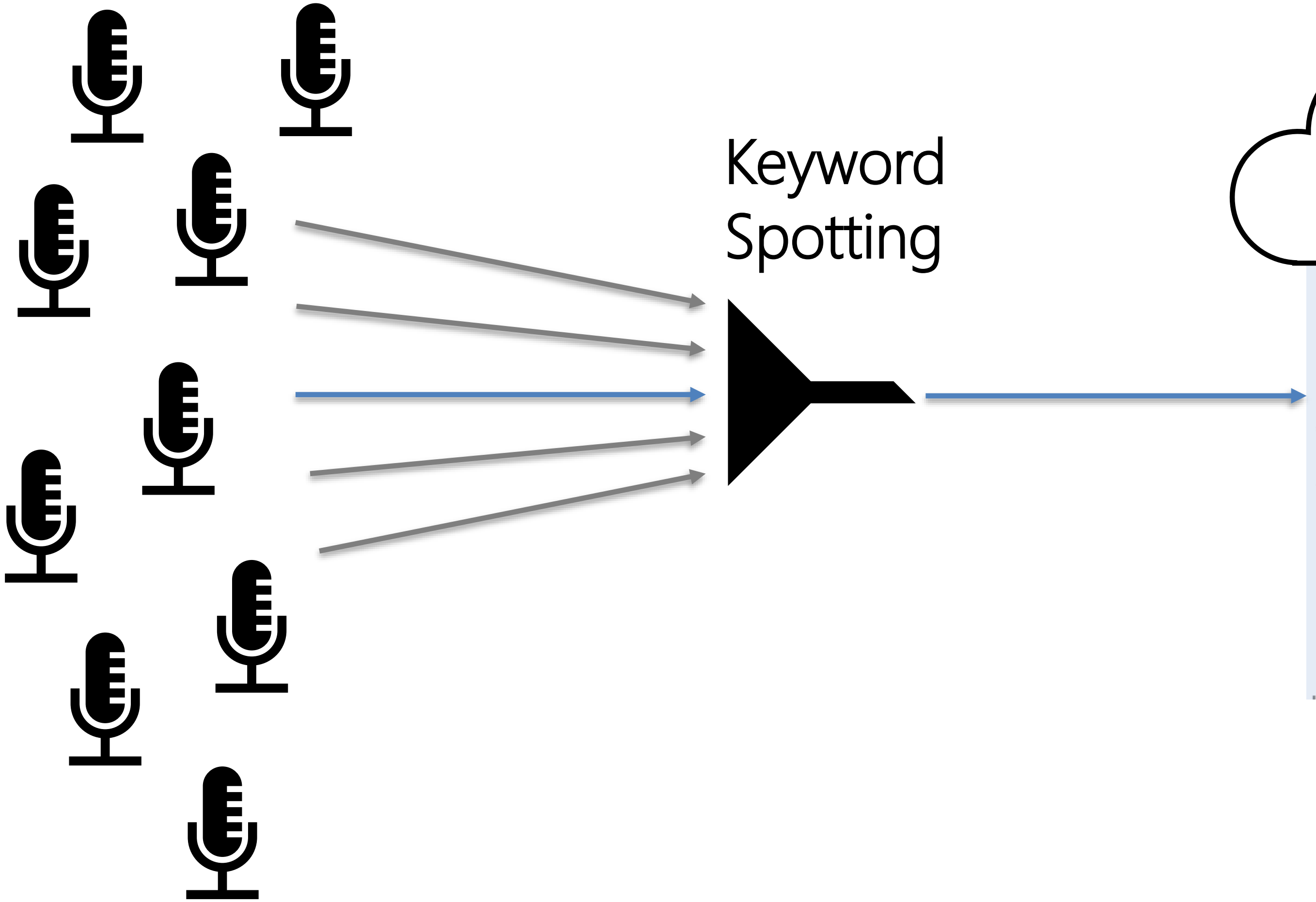


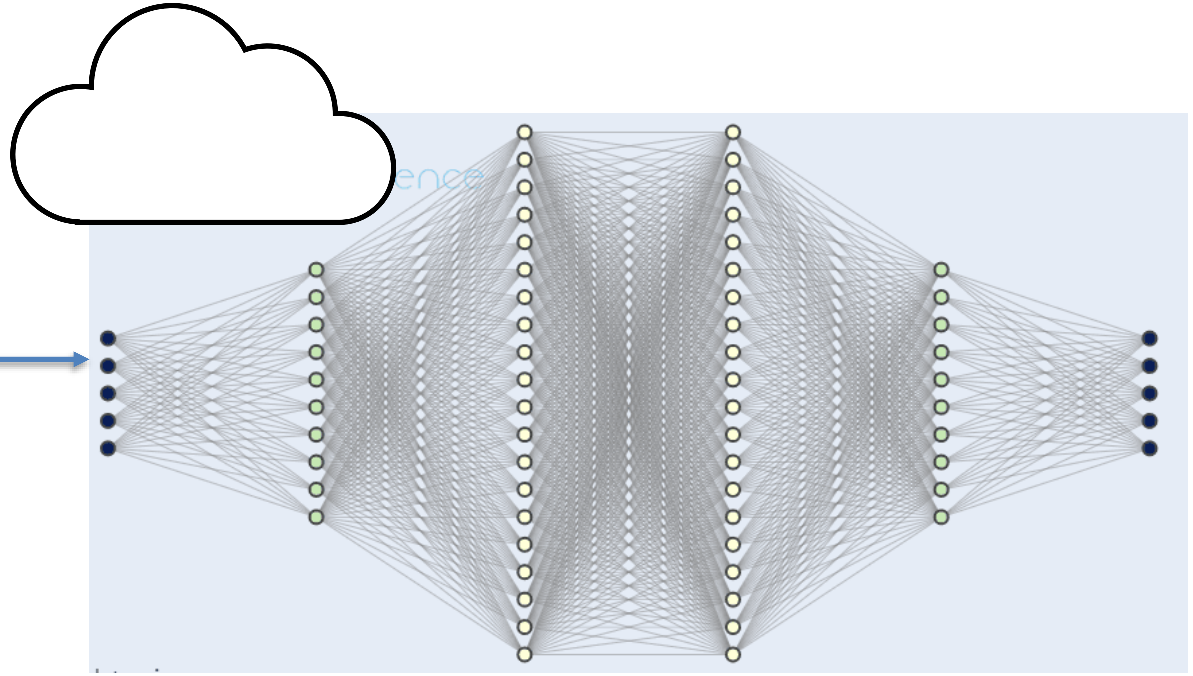
2- Play
response

Why do we need keyword spotting?



Why do we need keyword spotting?





Need #2: **Scalability** of the whole ASR pipeline

POLITECNICO MILANO 1863

The input data

Collect Data → Pre-process data → Design Model → Train a Model → Evaluate optimize → Convert Model → Deploy Model → Make inference

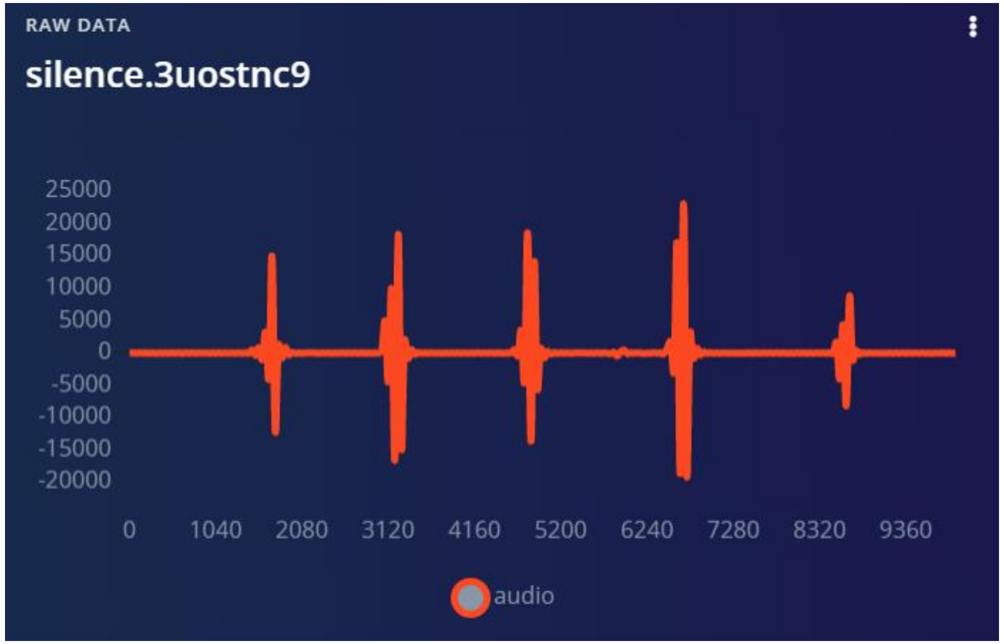
Data collection with edge impulse

Data collection

Label: silence

Length: 10s.

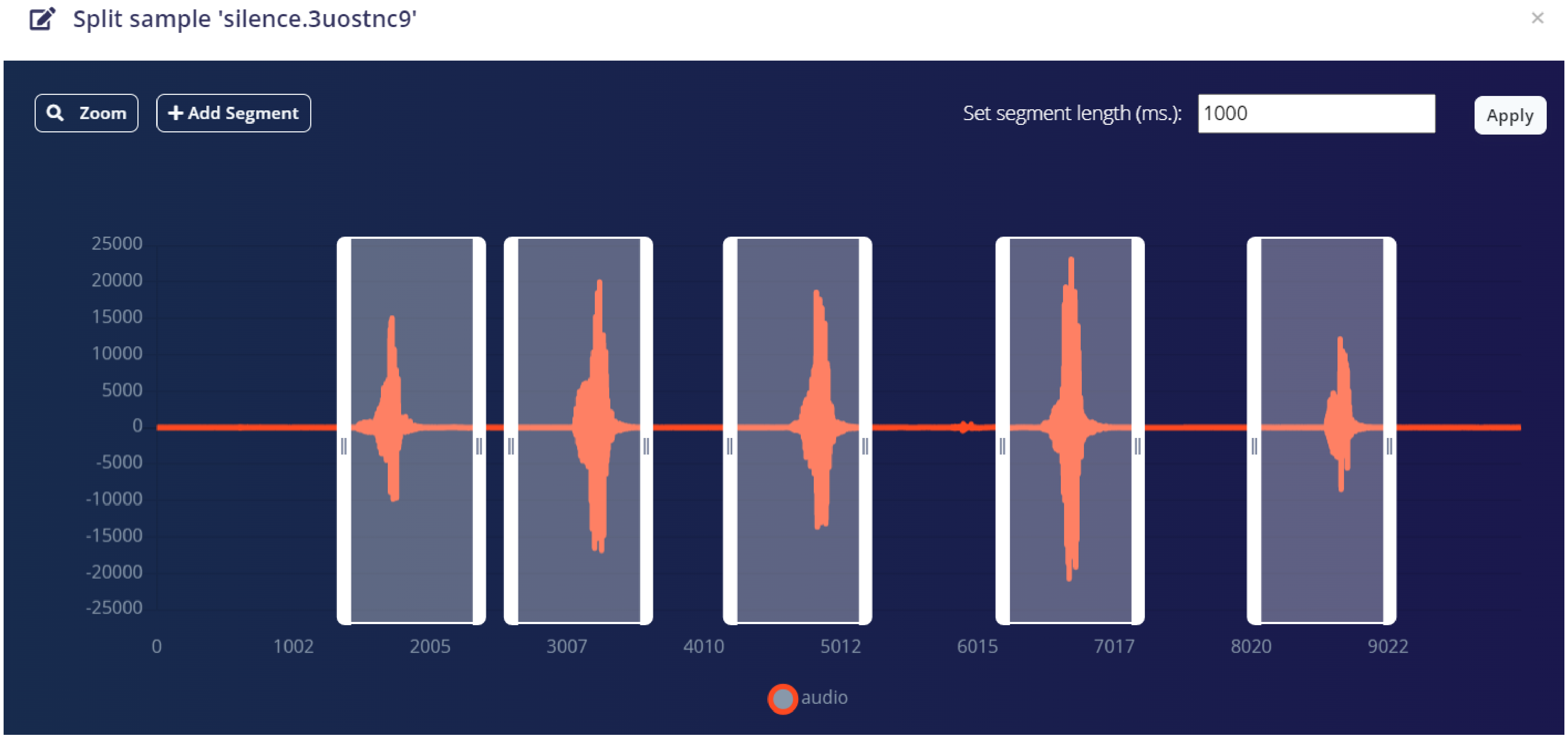
Category: Training



Start recording

Audio captured with current settings: 10s

Data collection with edge impulse



Keyword spotting dataset

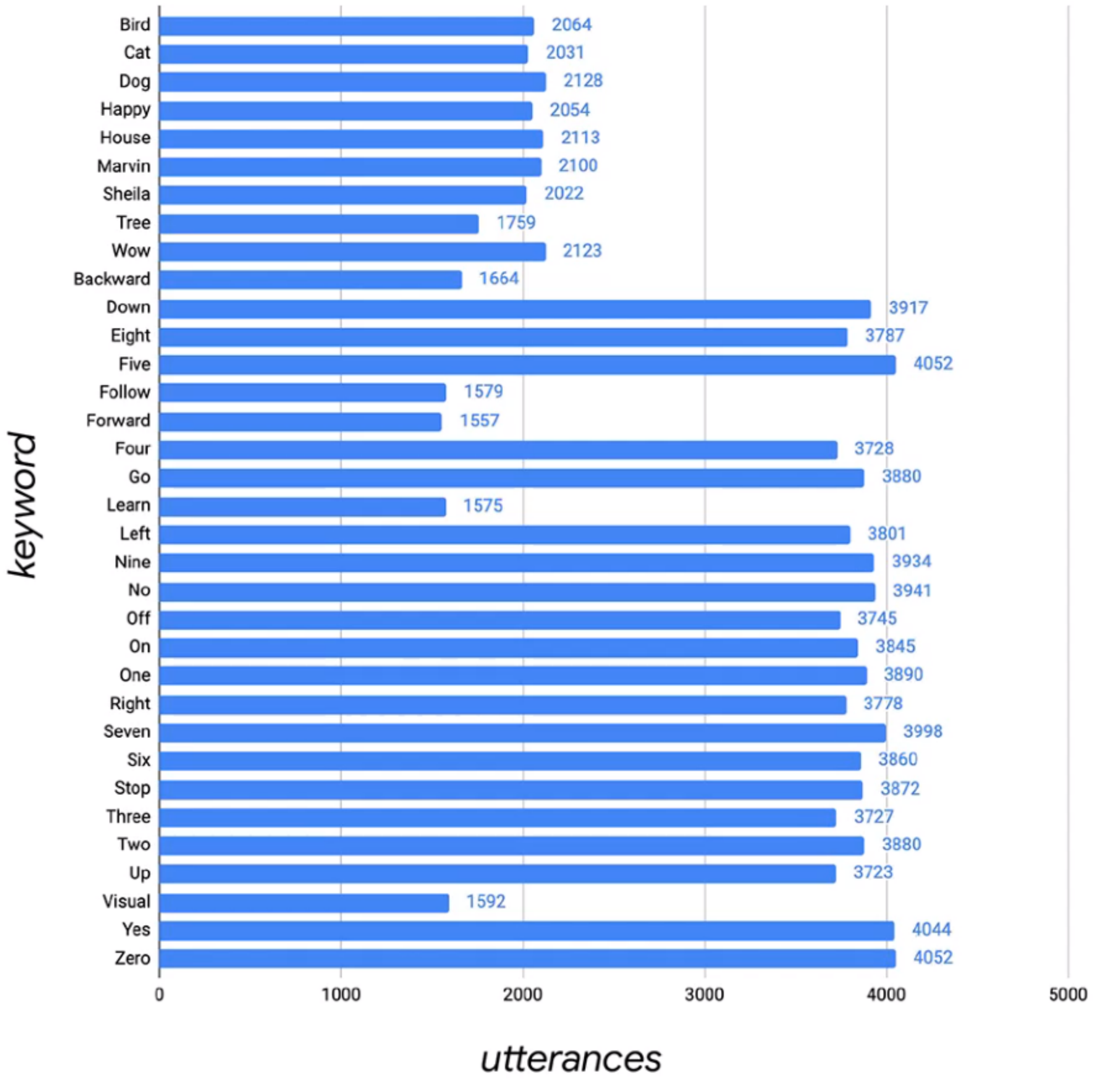
Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition

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

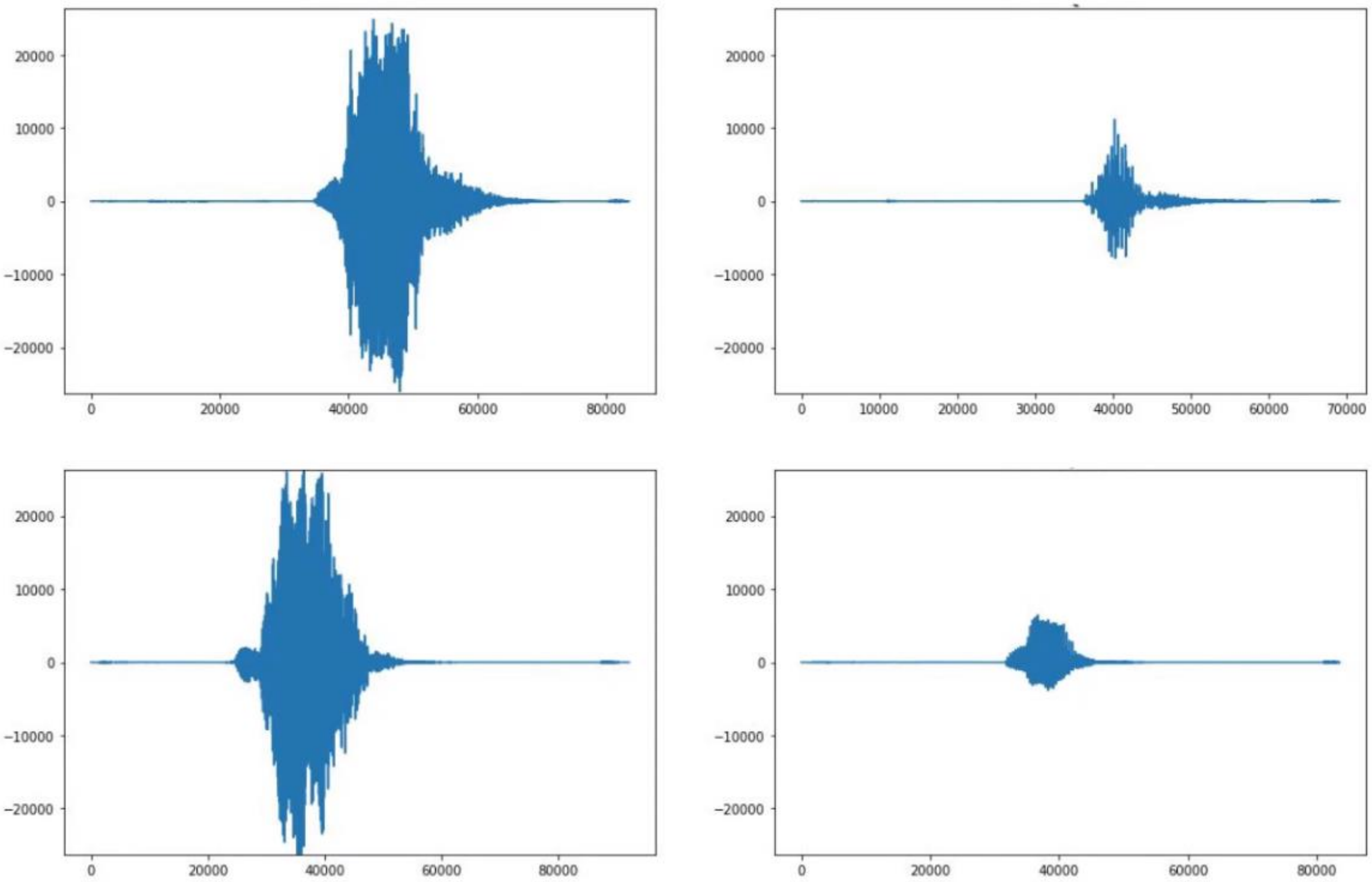
April 2018

- Recorded as individual words not phrases
- 1000-4000 examples for each word
- >2,500 people collecting words

Keywords on the dataset



Raw data may be confusing…

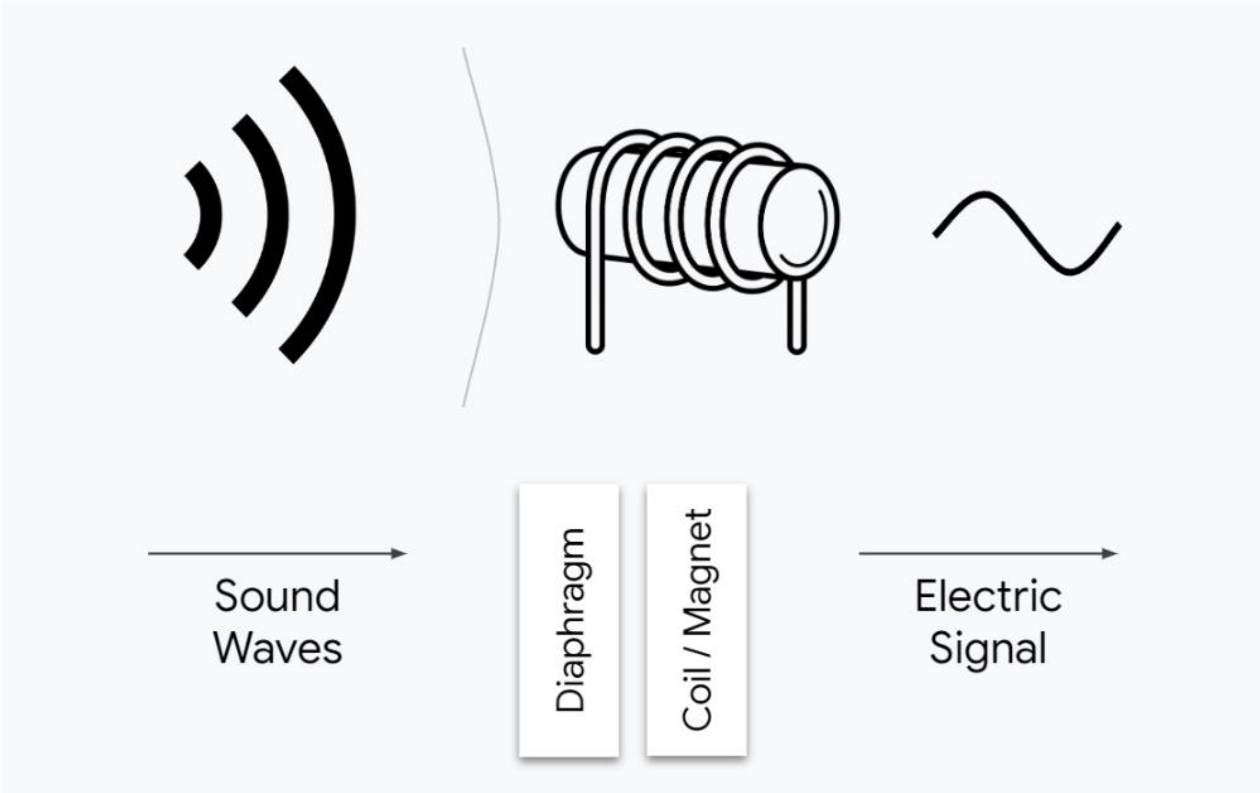


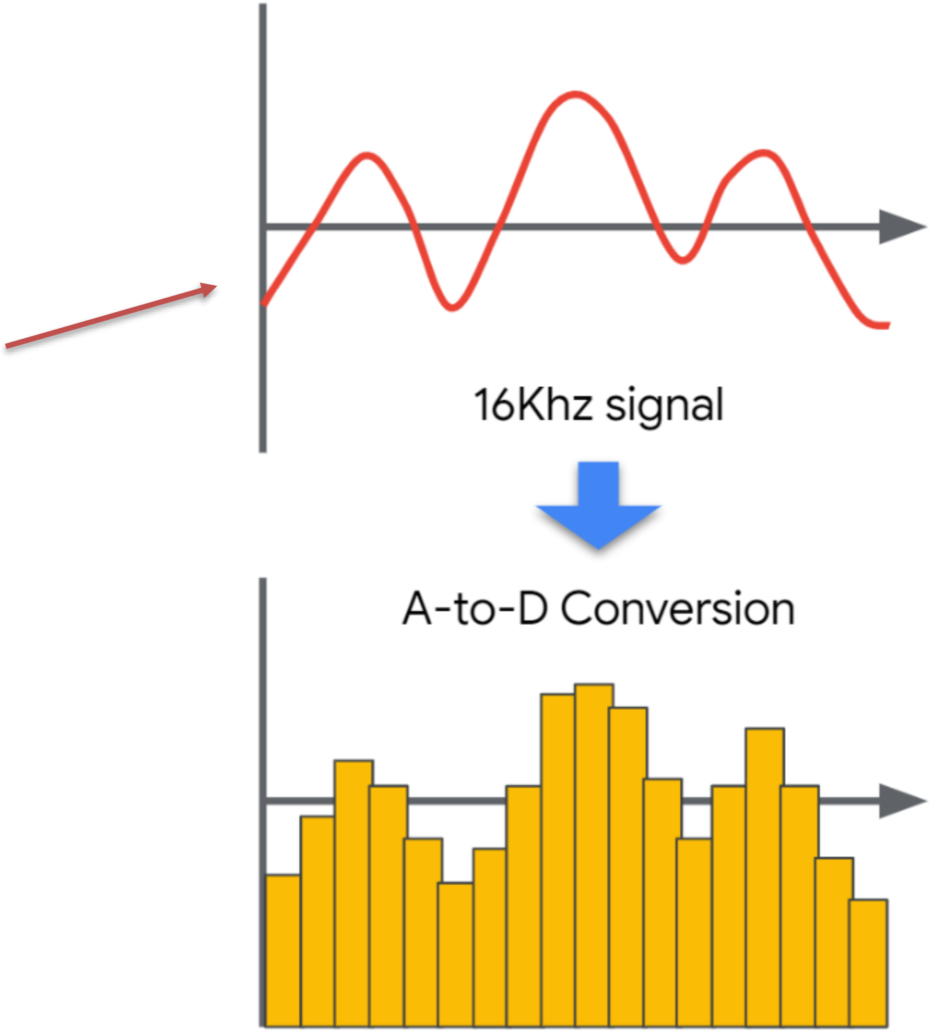
POLITECNICO
MILANO 1863

Pre-processing

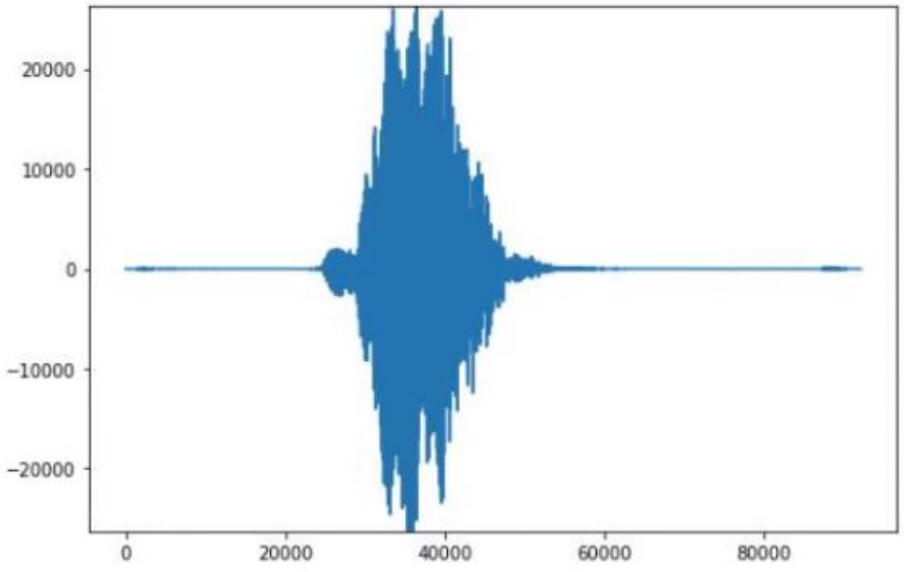
- Collect Data
- Pre-process data
- Design Model
- Train a Model
- Evaluate optimize
- Convert Model
- Deploy Model
- Make inference

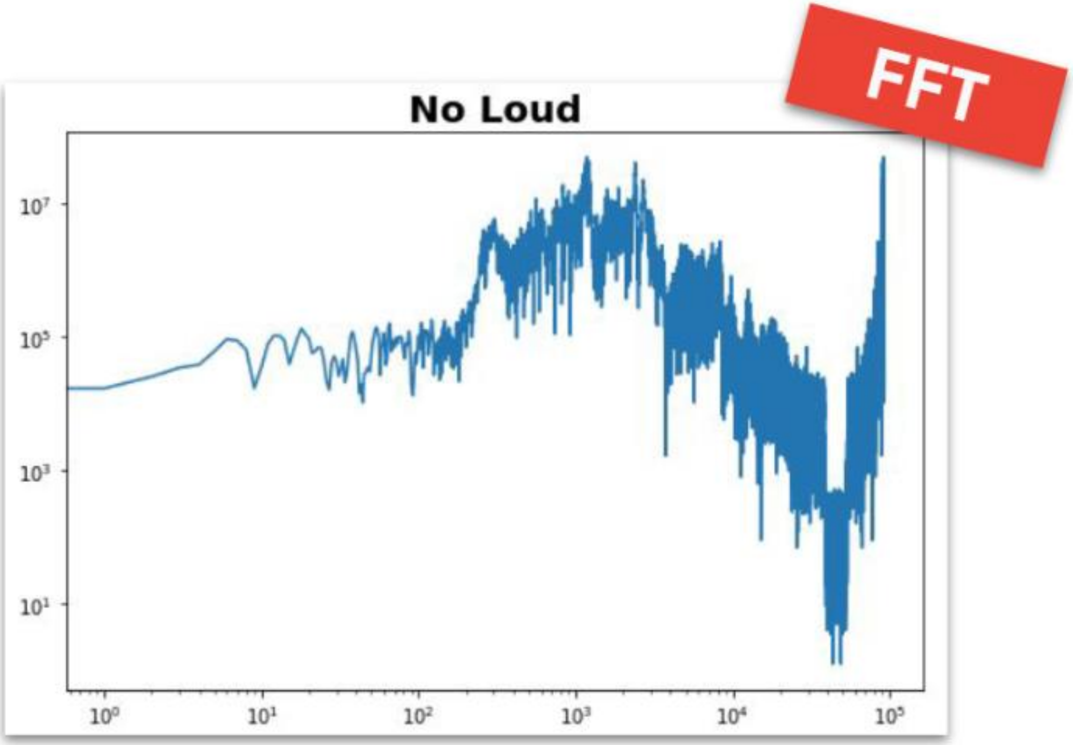
Sensor data


Sound
Waves
Diaphragm
Coil / Magnet
Electric
Signal

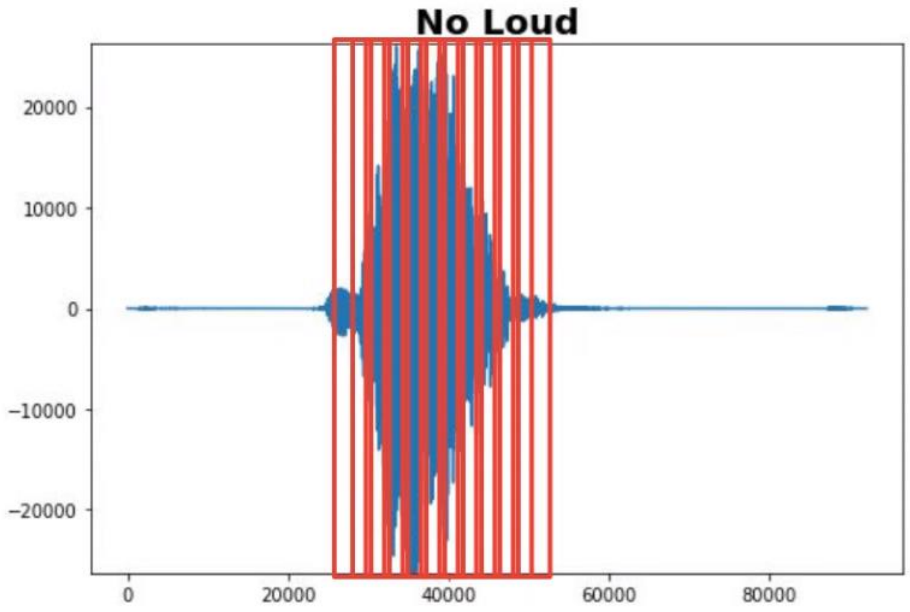

16Khz signal
A-to-D Conversion

Shifting from Time Domain to Frequency Domain

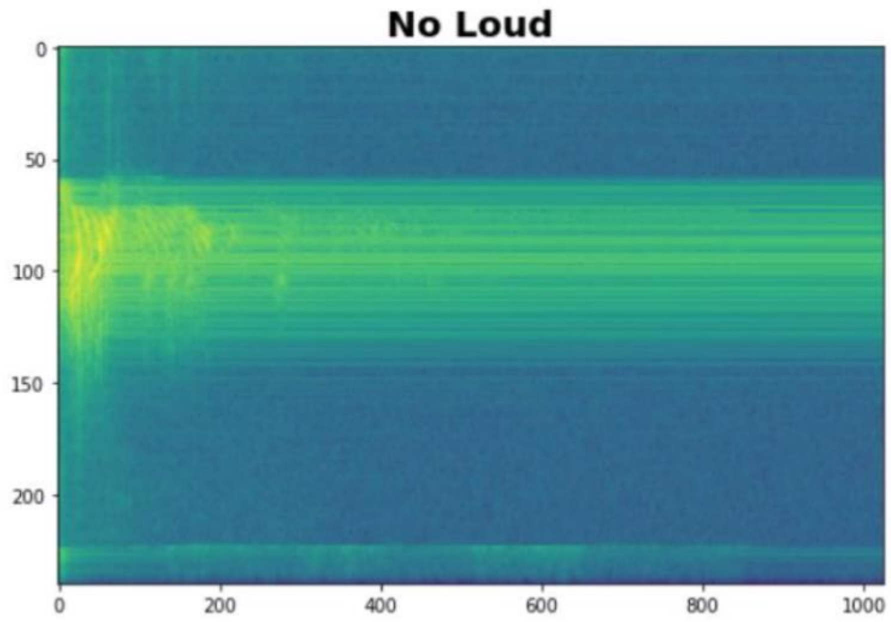




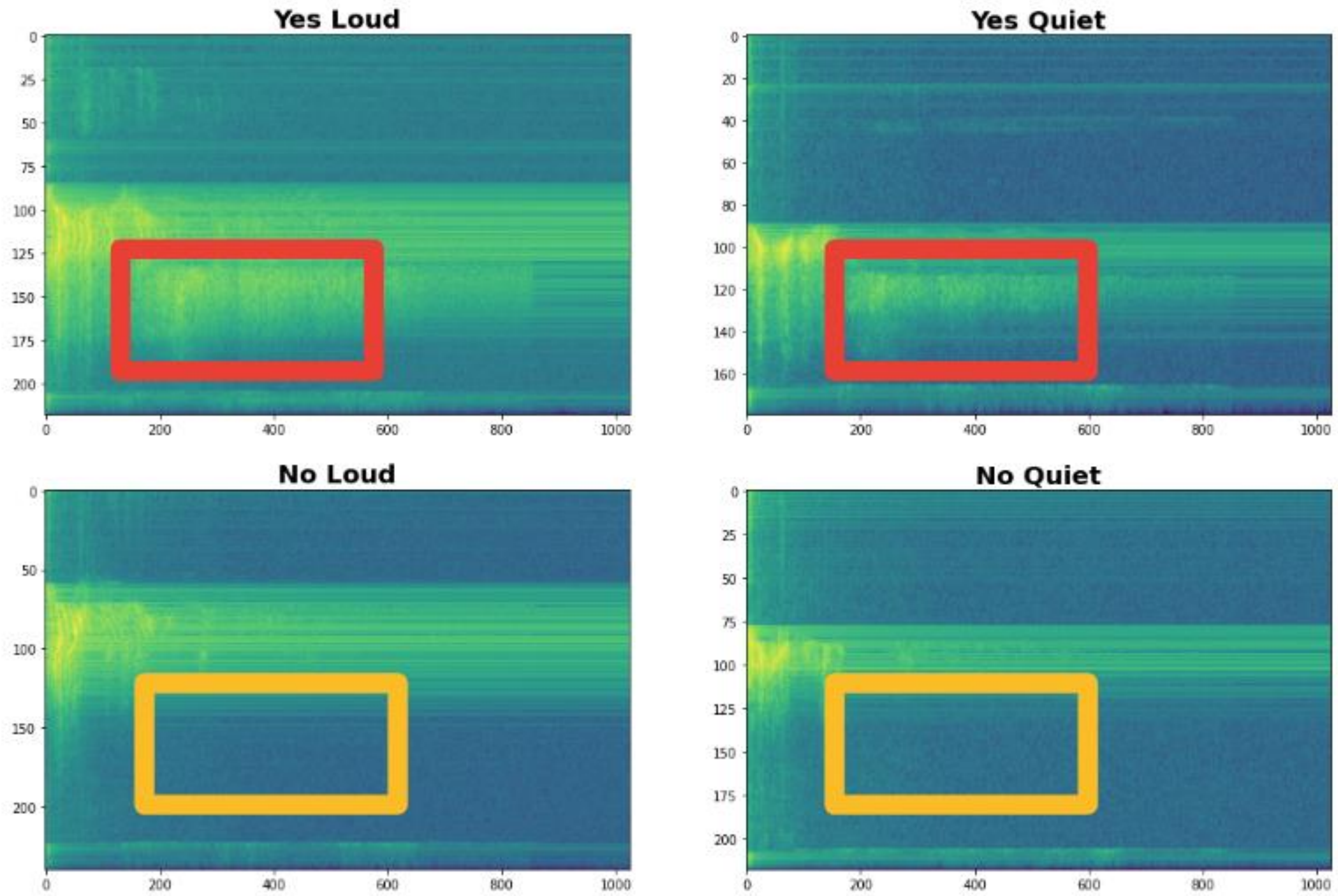
Extracting spectrograms!



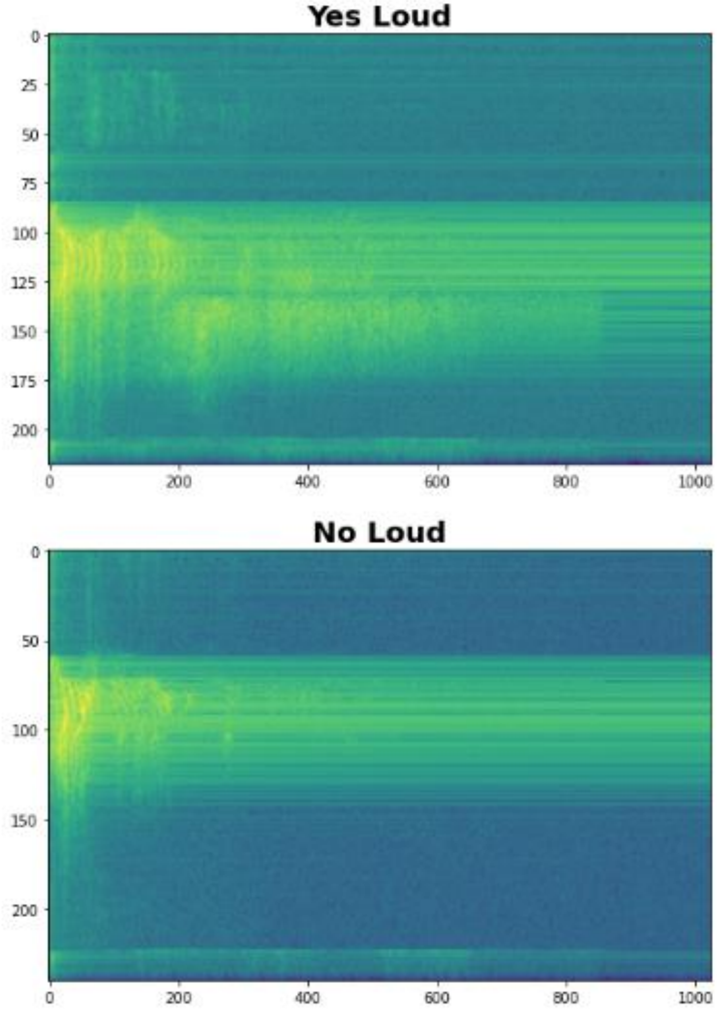
FFT!



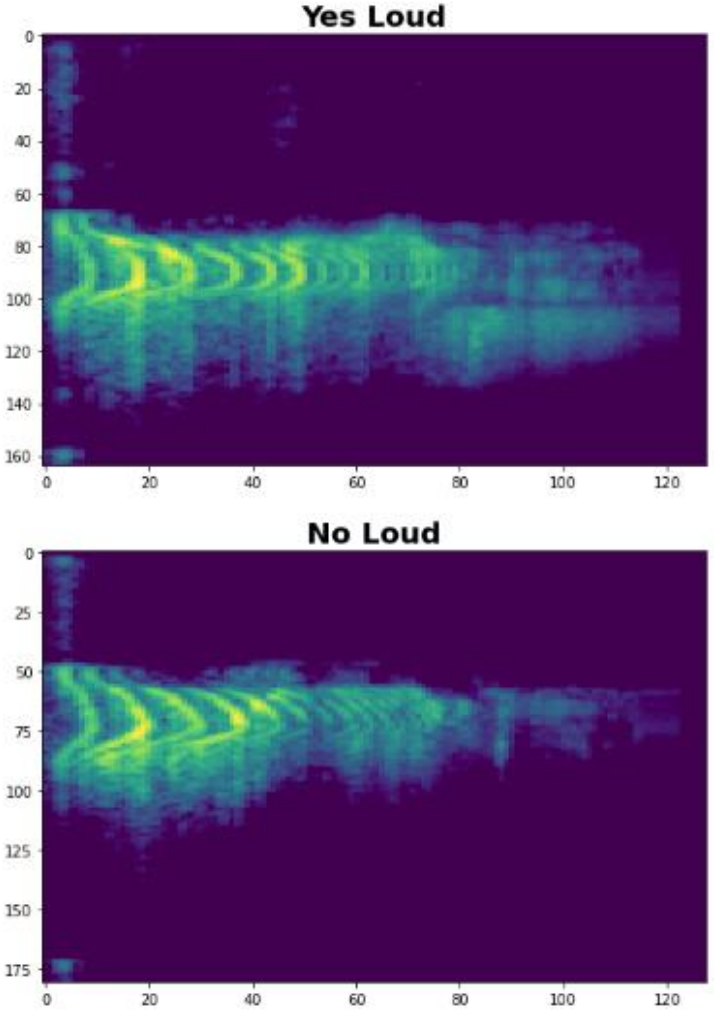
Better!



Preprocessing is essential

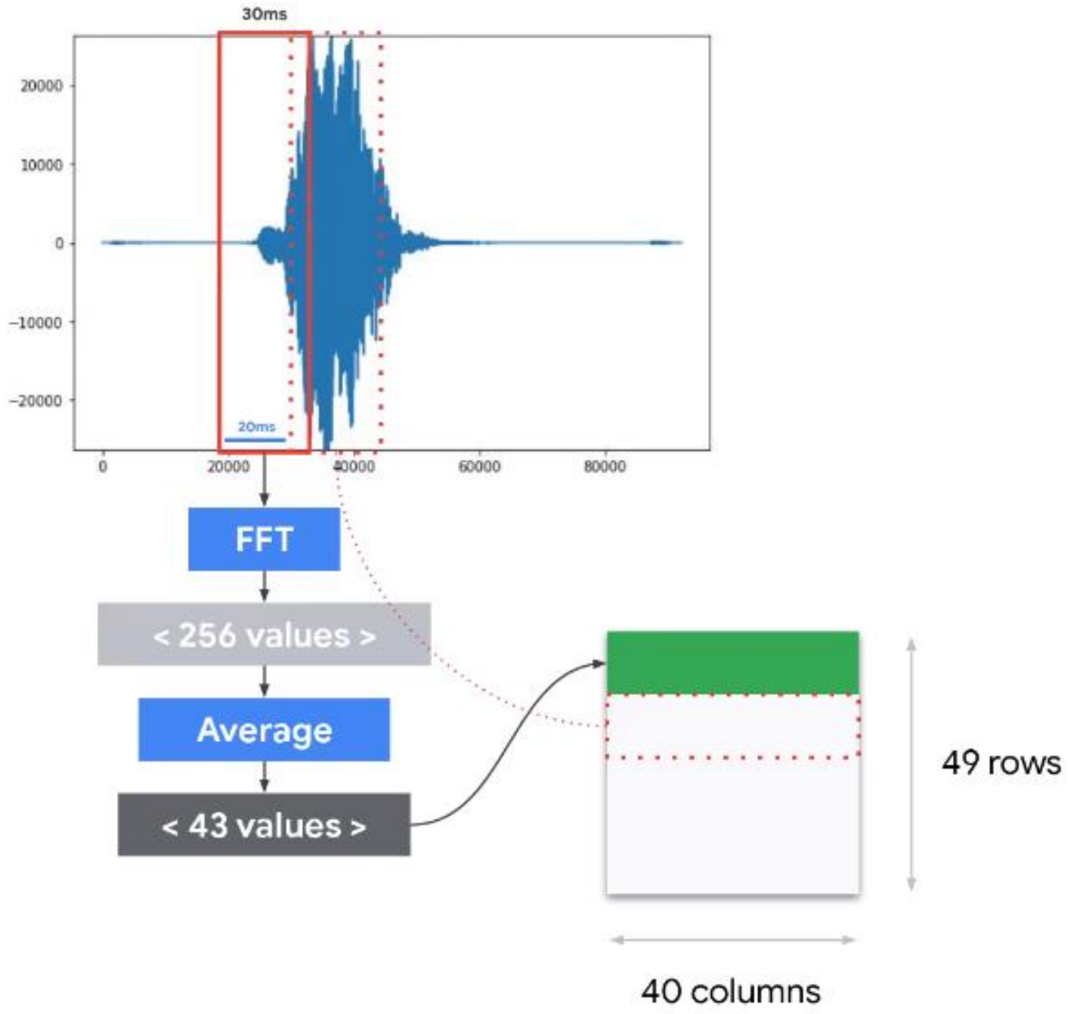


FFT preprocessing



Mel filters preprocessing

Pre-processing can help with dimensionality reduction

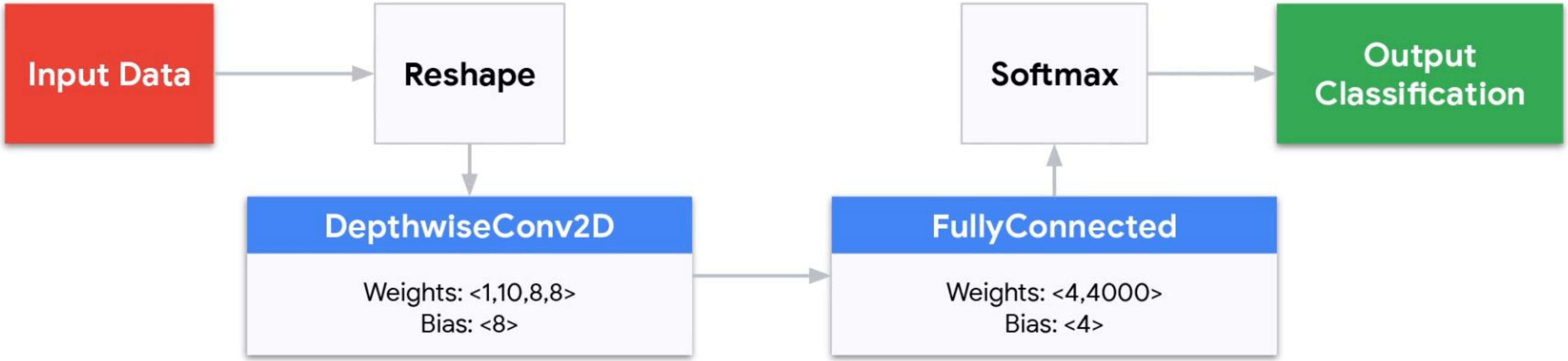


For 1 second inputs, collected at 16KHz, we go from 16000 values to 40*49 = 1960 values

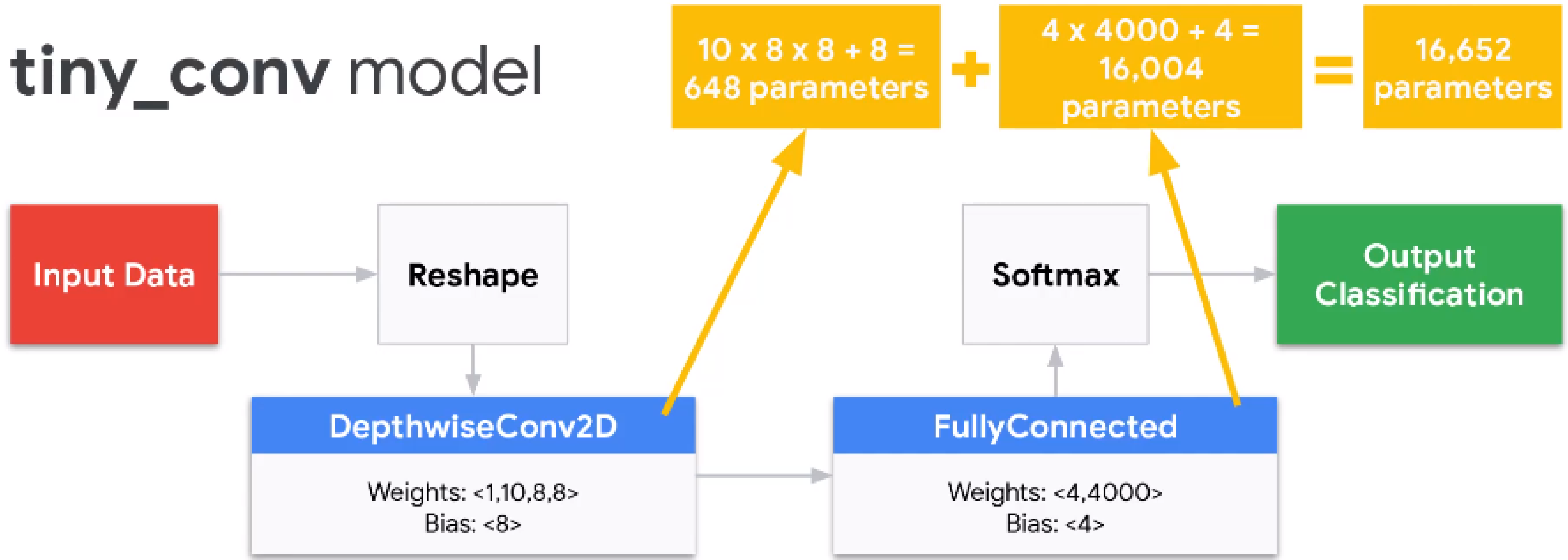
POLITECNICO
MILANO 1863

Designing and training a model

Tiny_conv model



In terms of weights?



For floats that's <70Kb
Quantized that's <17Kb

Let's try a pre-trained network for recognizing «yes» from «no» from «other»

COLAB:

https://colab.research.google.com/drive/1YH6vXlDzzCRZOT-sLx50TNAE-LhVkcOK?usp=sharing

Train your own!

COLAB:

Feature extraction:

https://colab.research.google.com/drive/10pxaPTL0QAhlu4L7U2DAqzcwuiKAe8nC?usp=sharing

Training:

https://colab.research.google.com/drive/1j3mGVMuoQRT-TWRgmyqxb-AeVVlMcwbL?usp=sharing

Additional data and code:

https://drive.google.com/drive/folders/1_L3HJjCn536UmYnzBdzY7d2-_N9LMyXQ?usp=sharing

POLITECNICO MILANO 1863

Evaluating a model

Collect Data → Pre-process data → Design Model → Train a Model → Evaluate optimize → Convert Model → Deploy Model → Make inference

Metrics during training

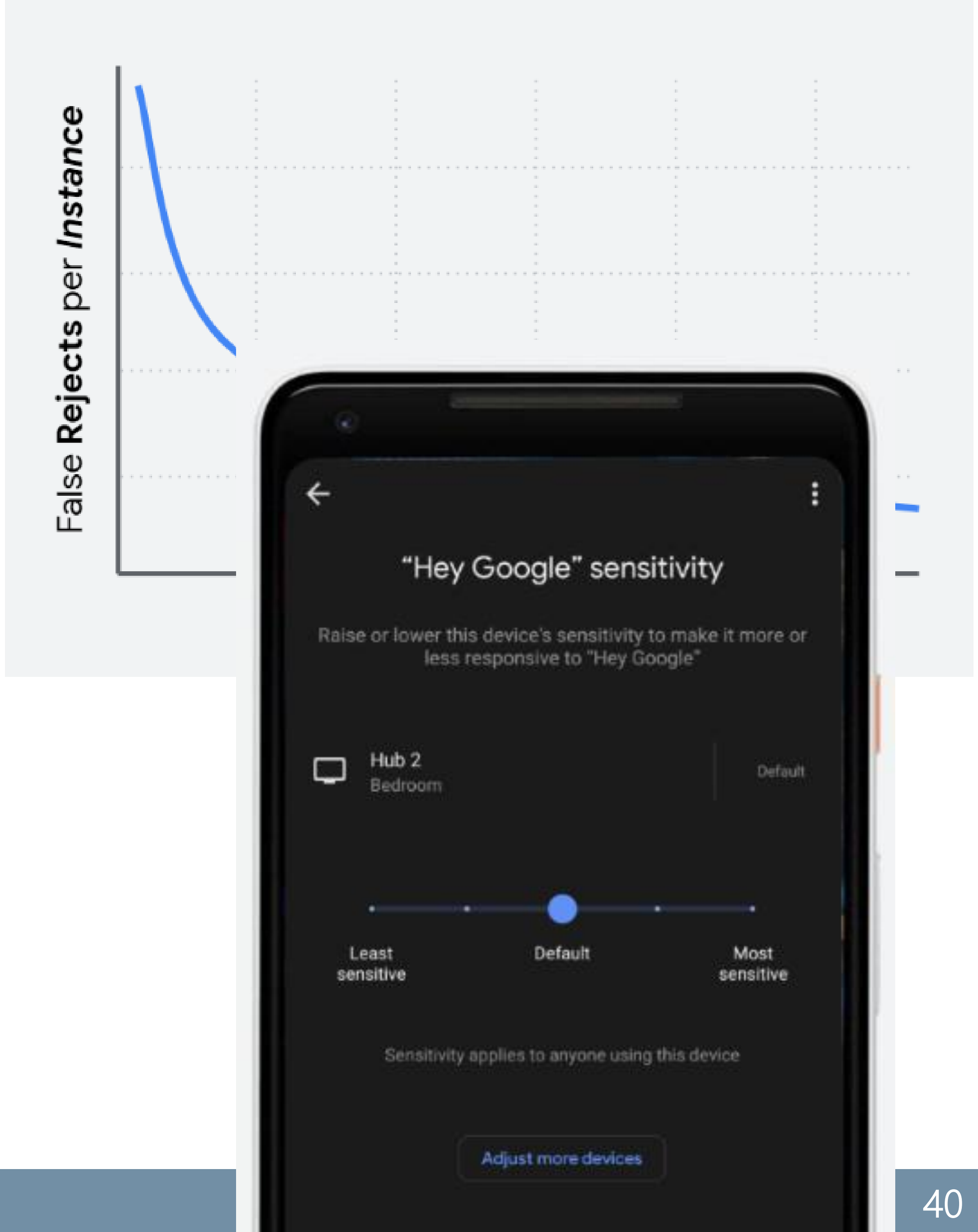
- Since during training we control the composition of the datasets, accuracy is a good metric during this phase of the algorithm.
- In general, it's good to plot the confusion matrix to control also false positive and false negative rates.
- Accuracy, precision recall and F1 score can be taken in consideration

	Actual Class: 1	Actual Class: 0
Predicted Class: 1	tp	fp
Predicted Class: 0	fn	tn

$$Acc = \frac{tp + tn}{N} \qquad Pre = \frac{tp}{tp + fp} \qquad \text{Recall: } Rec = \frac{tp}{tp + fn} \qquad F1 = \frac{2 \cdot Pre \cdot Rec}{Pre + Rec}$$

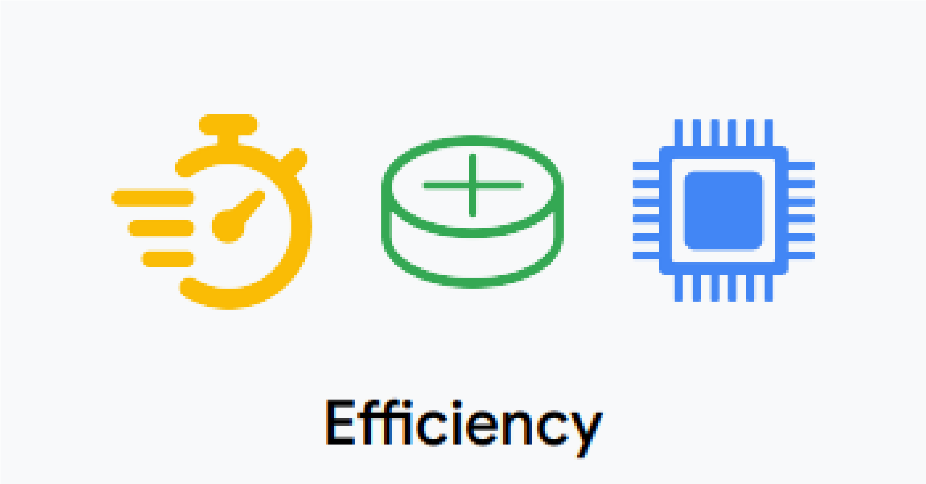
Metrics during use

- With many algorithms, it's possible to trade off false positive for false negatives.
- With multiple models, we can draw a receiver-operator curve (ROC) and select an operating point.



Other metrics: efficiency

- Latency:
  - Model must be fast enough to keep up with the speech input
  - The model must run fast enough to be responsive to the end user
  - But it must run efficiently on a small processor TinyML
- Memory Usage:
  - Need to be resource aware
  - Less compute
  - Less memory
  - Use quantization



Appendix

Credits and reference

- “TinyML: Machine Learning with TensorFlow Lite on Arduino and Ultra-Low-Power Microcontrollers”, Daniel Situnayake, Pete Warden, O'Reilly Media, Inc.
- Online course:
  - https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning
- A lot more material on TinyML:
  - http://tinyml.seas.harvard.edu/

- Special thanks to Gioele Mombelli for letting me use the code he developed for his master thesis

- Colab to better understand pre-processing and spectrograms generations:
  - https://colab.research.google.com/github/tinyMLx/colabs/blob/master/3-5-10-SpectrogramsMFCCs.ipynb