POLITECNICO
MILANO 1863

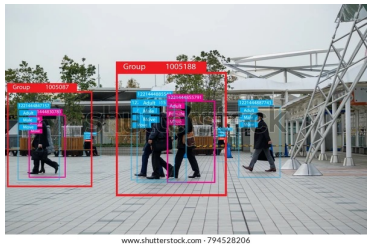
Hardware Architectures for Embedded and Edge AI

Prof Manuel Roveri – manuel.roveri@polimi.it

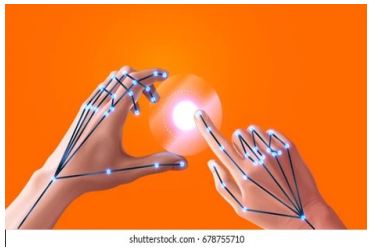
Lecture 3 – Algorithms for Embedded and Edge AI

An overview of an embedded and edge AI system

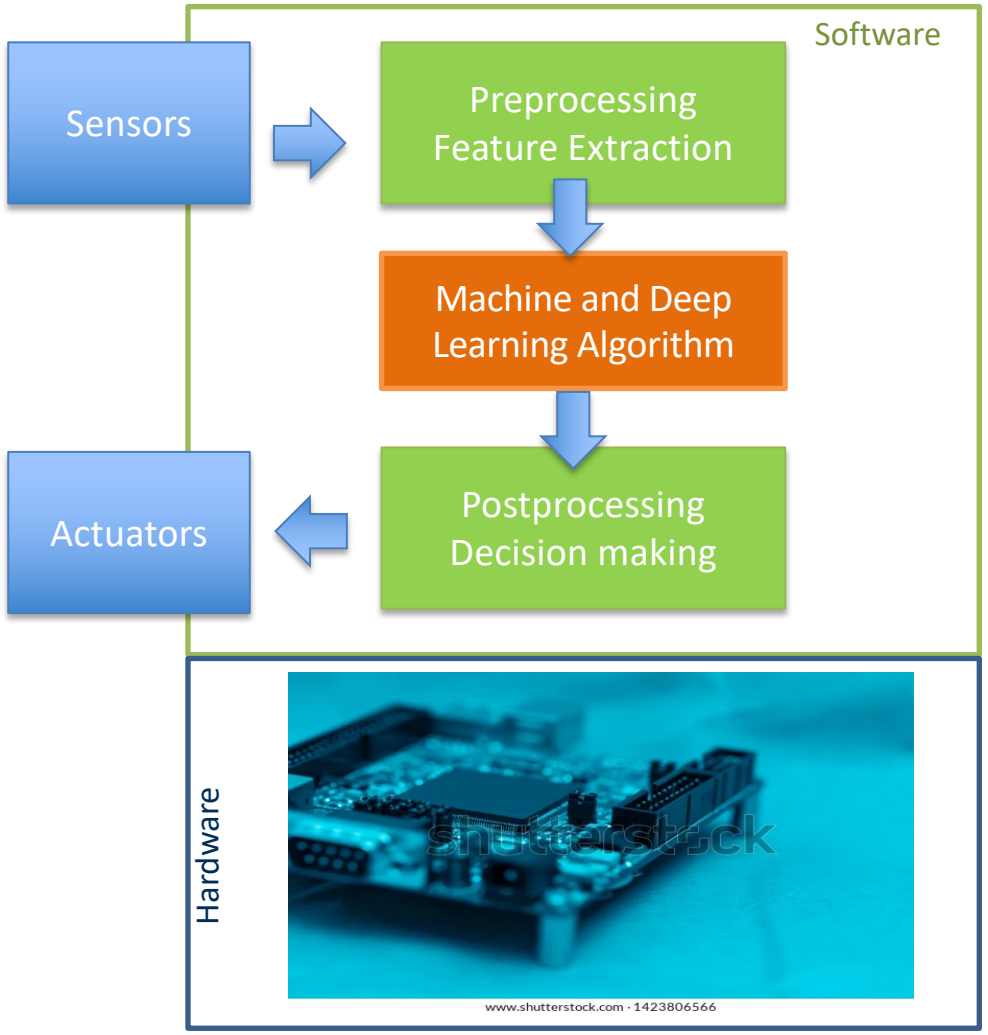
Wake-word detection



Person detection



Gesture recognition

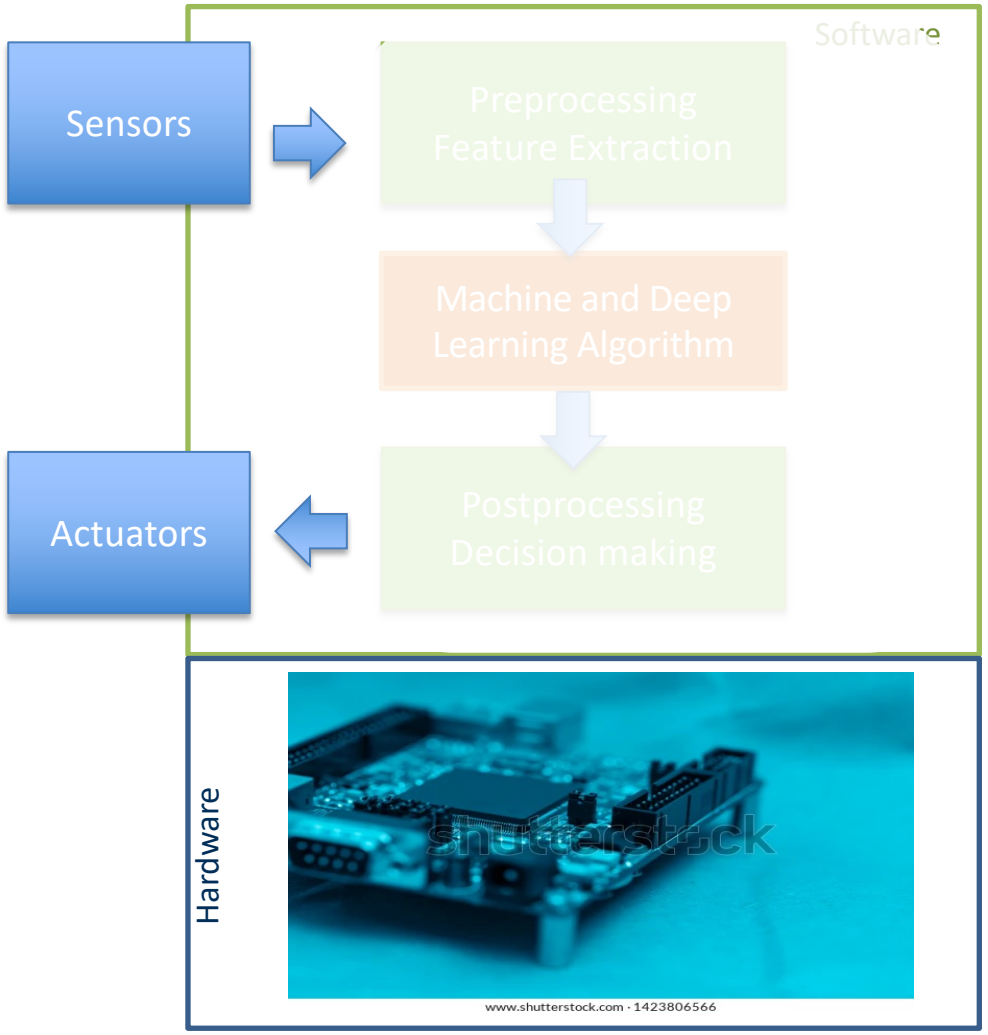


An overview of an embedded and edge AI system

Wake-word detection

Person detection

Gesture recognition



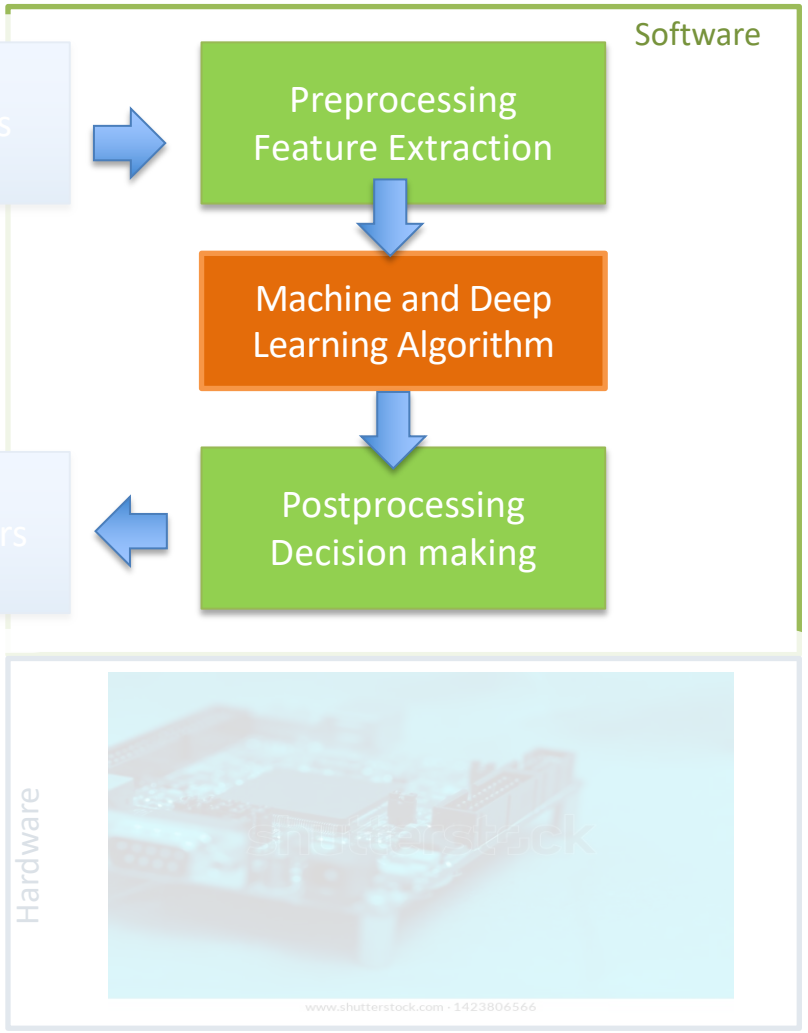
Previous lecture!

An overview of an embedded and edge AI system

Wake-word detection

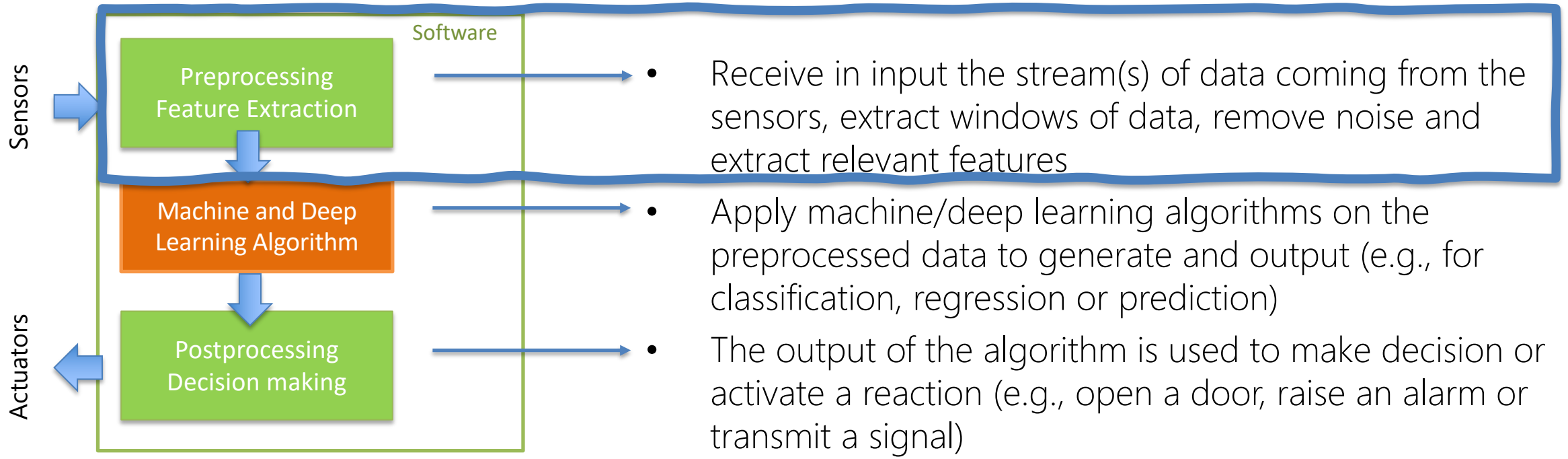
Person detection

Gesture recognition



This lecture!

Algorithms for embedded and edge AI



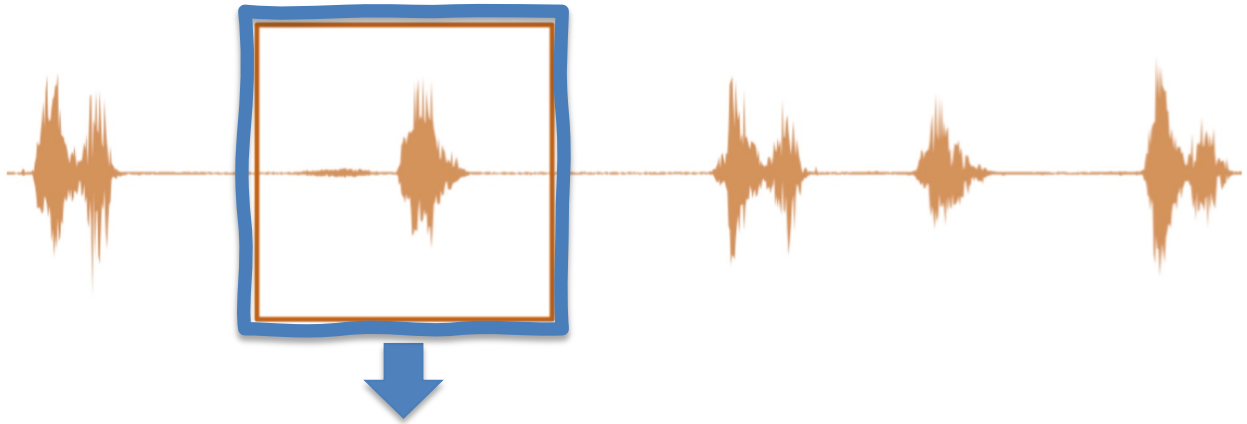
- Receive in input the stream(s) of data coming from the sensors, extract windows of data, remove noise and extract relevant features
- Apply machine/deep learning algorithms on the preprocessed data to generate and output (e.g., for classification, regression or prediction)
- The output of the algorithm is used to make decision or activate a reaction (e.g., open a door, raise an alarm or transmit a signal)

1) Data preprocessing and feature extraction

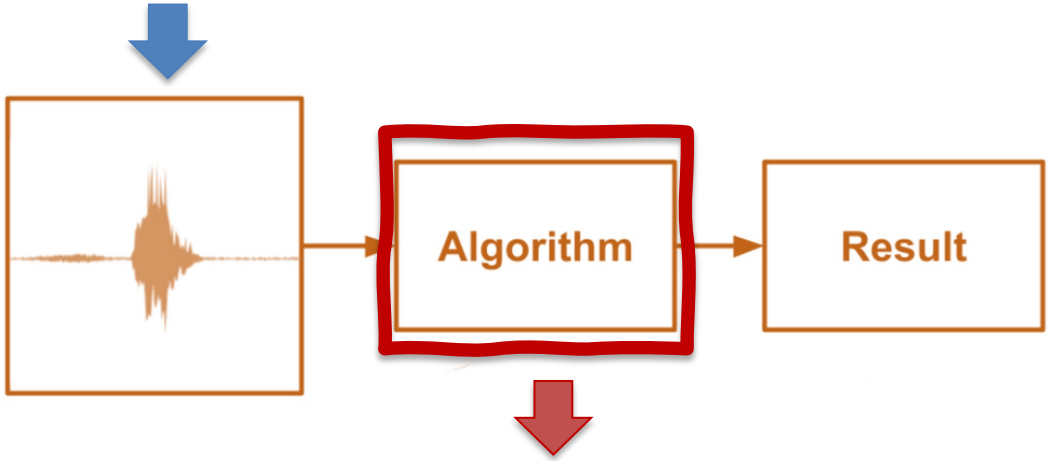
- Goal: transform raw signals into data to be processed by machine and deep learning algorithm
- How to achieve this goal?
  a) Chopping of raw signals into chunks of data
  b) Apply digital signal processing algorithms on chunks of data to remove noise and highlight relevant "portions" of the signal present in the chunks
  c) Extract features from the pre-processed chunks of data

a) Chopping streams of data

Sensors produces streams of data



Windows of data are extracted from the stream (a chunk of data)

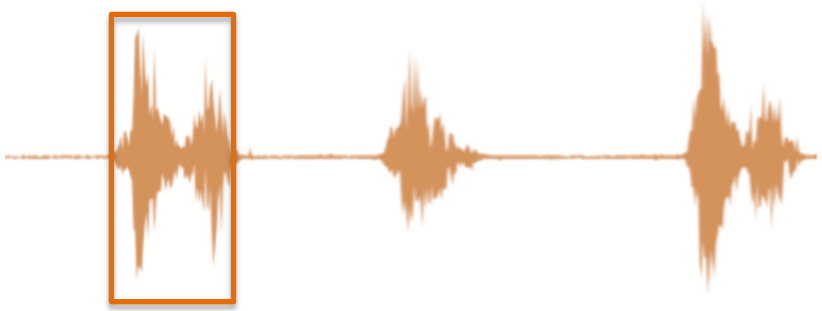


The amount of time required to process a single chunk of data is called ***latency***

The chunk of data is processed by the algorithm generating a result

Latency and window size

- The **latency** defines how fast the embedded system can process the chunk of data
  - The lower the latency, the higher the amount of chunks of data that can be processed in a given amount of time (e.g., a second or a minute)



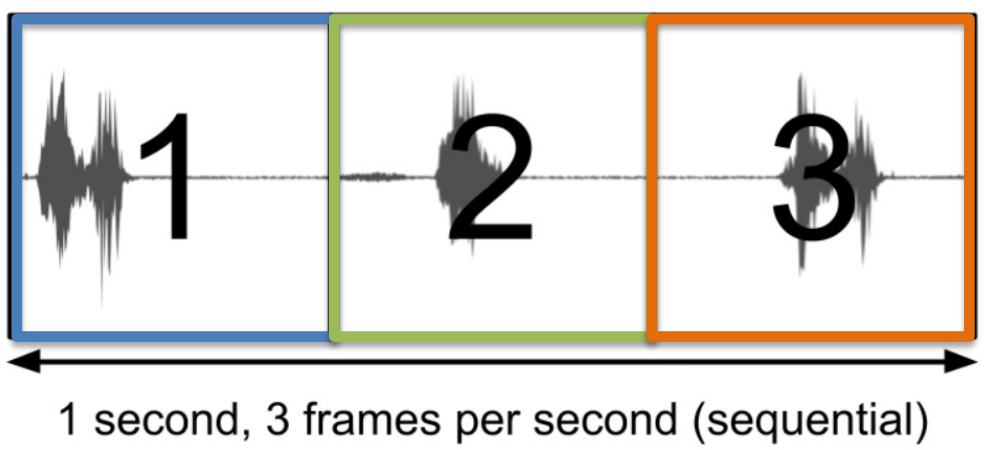
- the larger the window, the higher the latency
- the larger the window, the larger the amount of information that is present (hence better accuracies in the processing :-)

- The design choice of the **algorithm** has a strong effect on the **computational requirement** (the latency) as well as the **memory demand** (see next lectures)

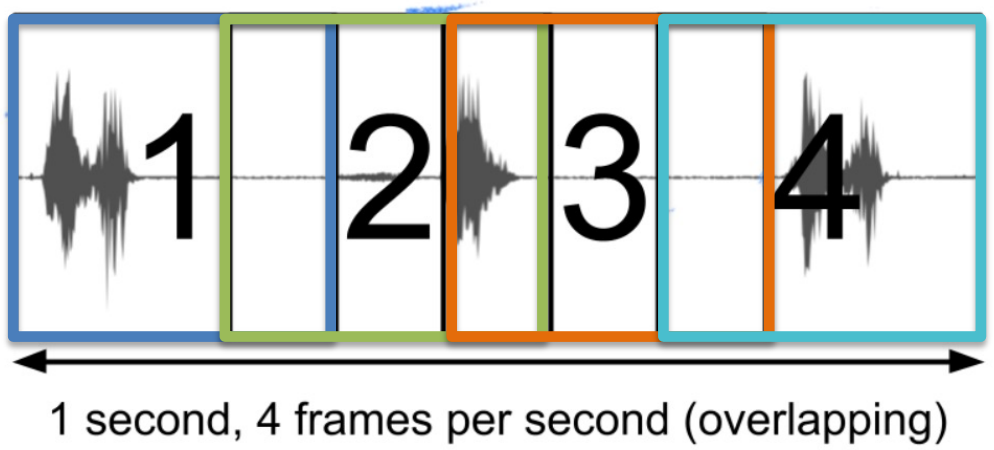
Image taken from [1]

Latency and frame rate

- The rate the system is able to acquire and process the data is called **frame rate** (the same holds also for images and videos streaming)



Non-overlapping windows



Overlapping windows

Image taken from [1]

The latency has a strong effect on the effectiveness of the system

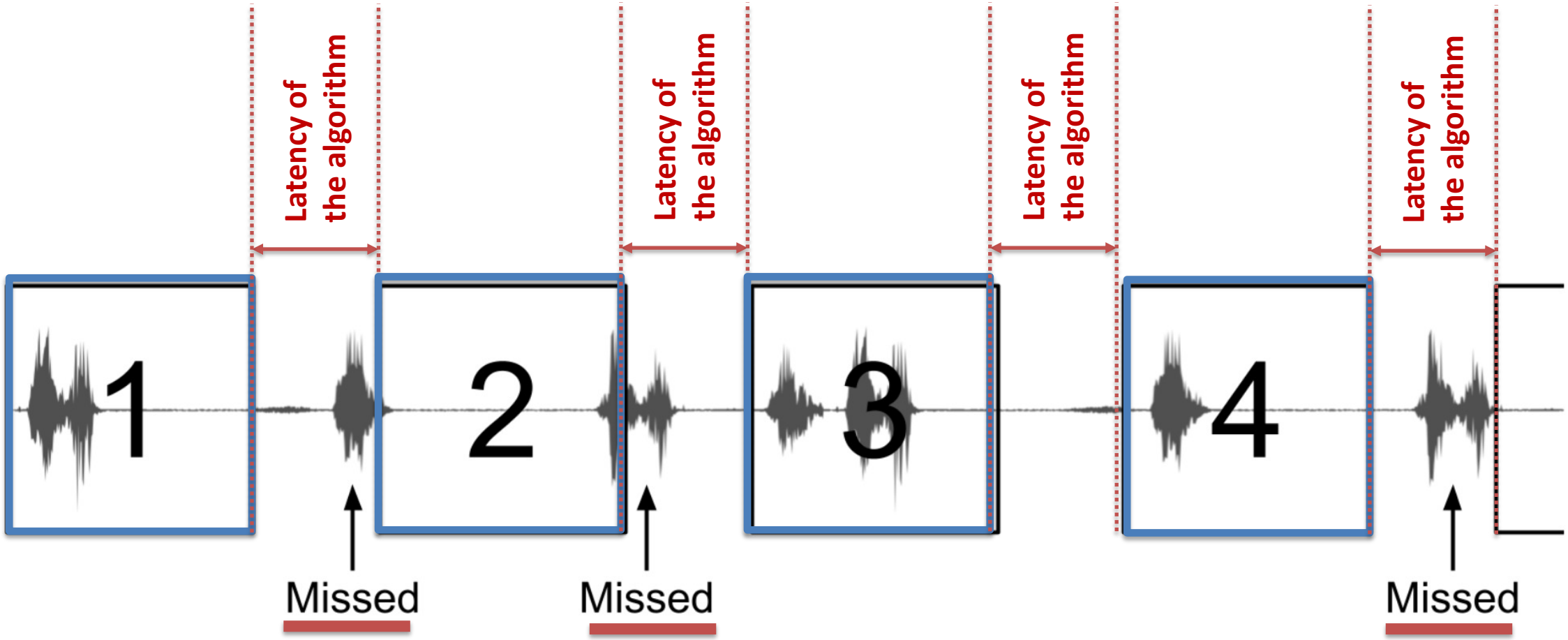


Image taken from [1]

1) Data preprocessing and feature extraction (recap)

- Goal: transform raw signals into data to be processed by machine and deep learning algorithm
- How to achieve this goal?
  a) Chopping of raw signals into chunks of data
  b) Apply digital signal processing algorithms on chunks of data to remove noise and highlight relevant "portions" of the signal present in the chunks
  c) Extract features from the pre-processed chunks of data

Digital processing algorithms for data preprocessing

Reconstruction of missing data

Resampling

Filtering

Digital processing algorithms for data preprocessing

Reconstruction of missing data

- "**Global**" filling methods: When we fill in missing data based on observations about the entire data set.
- "**Local**" filling methods: When we use neighboring data points to estimate the missing value (Forward Fill, Moving average, local Interpolation)
- **Deletion** of affected time periods: When we choose not to use time periods that have missing data at all.

Digital processing algorithms for data preprocessing

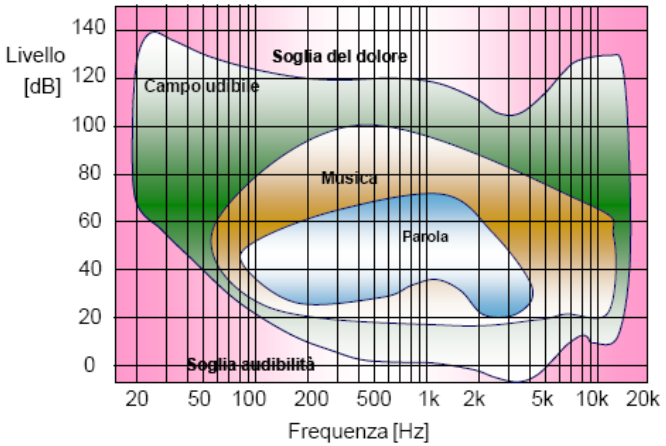
- Dealing with **time series having different sampling frequencies**
- Change the sampling frequency:
  - **Upsampling**: increasing the timestamp frequency (replicating, interpolating)
  - **Downsampling**: decreasing the timestamp frequency (subsampling)
- We do not change the actual rate
- Mind the aliasing (see Nyquist)

Resampling

- Dealing with **images** requiring to change the **spatial resolution** (pixel per image)
  - Downsample to **reduce** the resolution
  - Interpolation to **increase** the resolution
- Dealing with images requiring to **change the shape**:
  - Cropping
  - Resizing

Digital processing algorithms for data preprocessing

- "**Low pass** filter" to keep low frequencies (mind the cutoff frequency and frequency response)
- "**High pass** filter" to keep high frequencies (mind the cutoff frequency and frequency response)
- "**Band pass** filter" to keep low frequencies (mind the band frequency and frequency response)



E.g., for a speech recognition application the frequencies of interest range from 100Hz to 4KHz (i.e., the band of human speech).

We could band-pass the frequencies of interest and discard the rest

Filtering

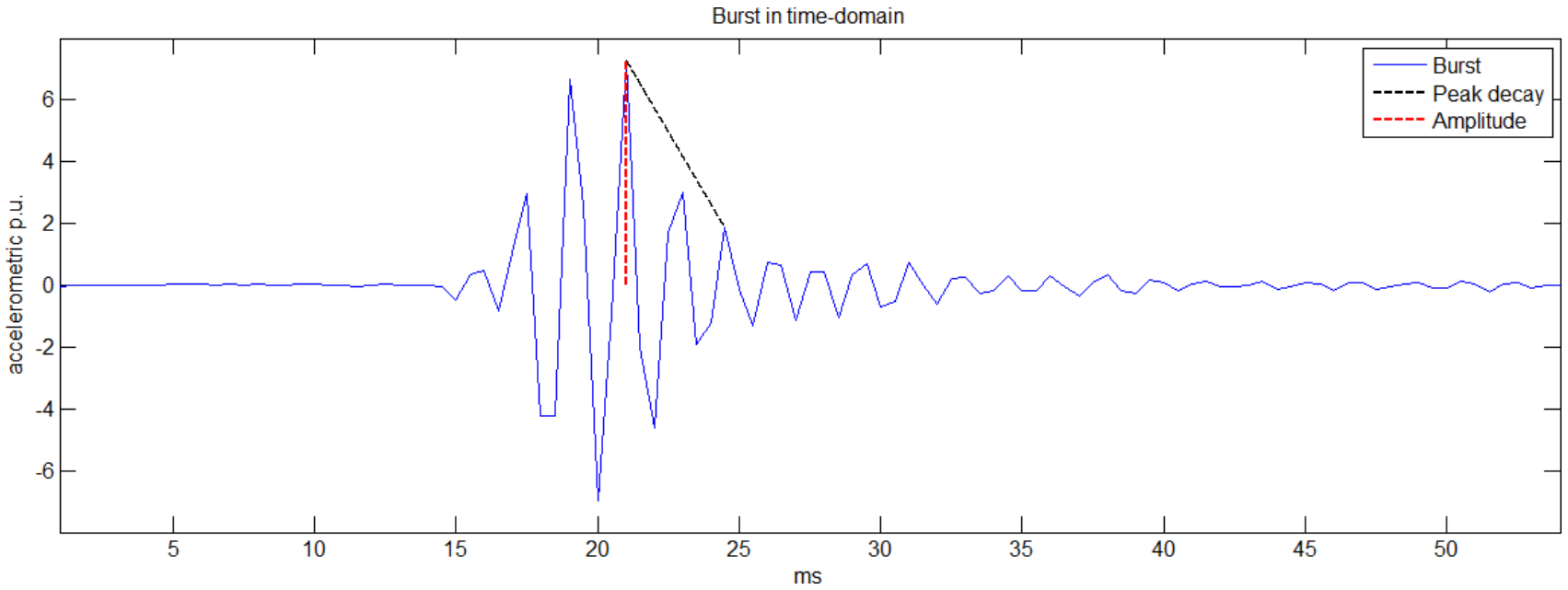
Many embedded and edge processors provide hardware support for the fast and energy-efficient computation of filtering operations

1) Data preprocessing and feature extraction (recap)

- Goal: transform raw signals into data to be processed by machine and deep learning algorithm
- How to achieve this goal?
  a) Chopping of raw signals into chunks of data
  b) Apply digital signal processing algorithms on chunks of data to remove noise and highlight relevant "portions" of the signal present in the chunks
  c) Extract features from the pre-processed chunks of data

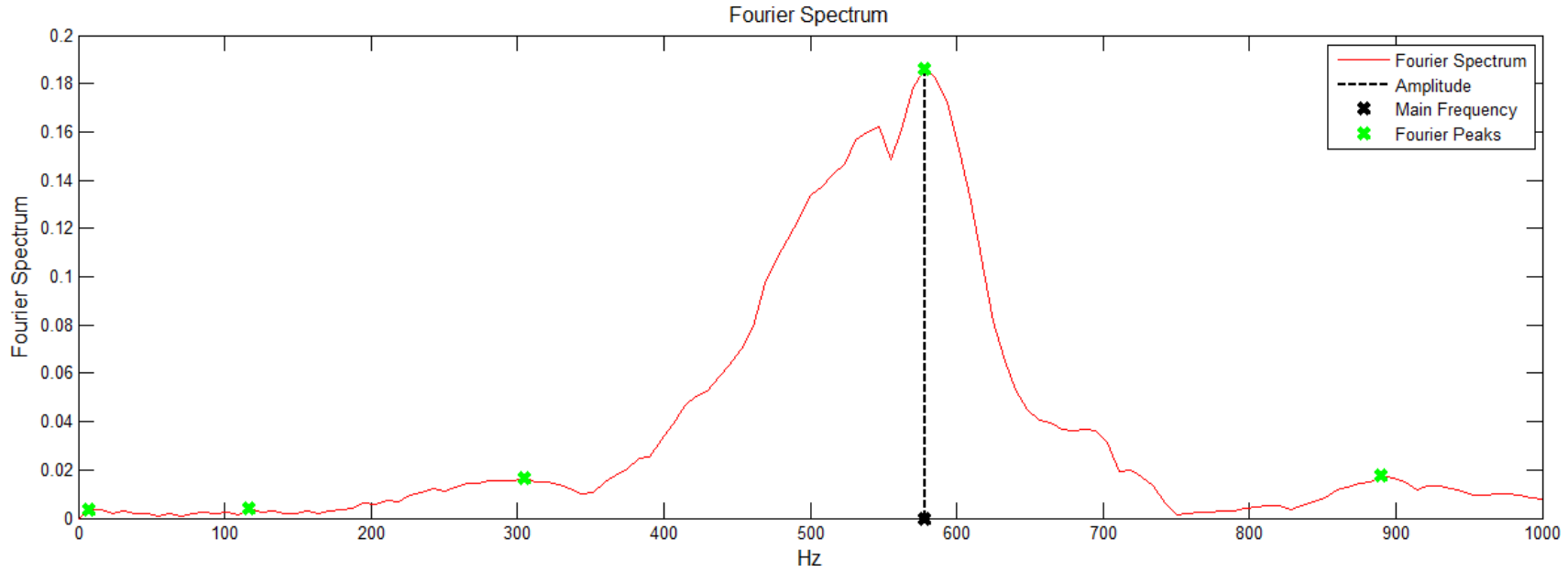
Feature Extraction in the time domain

- Mean (before PCA)
- PCA eigenvalue
- Max amplitude
- SNR
- Peak decay
- Energy



Feature Extraction in the frequency domain

- Max amplitude
- Main frequency
- Variance of peaks



Feature Extraction in the time-frequency domain: the spectrogram

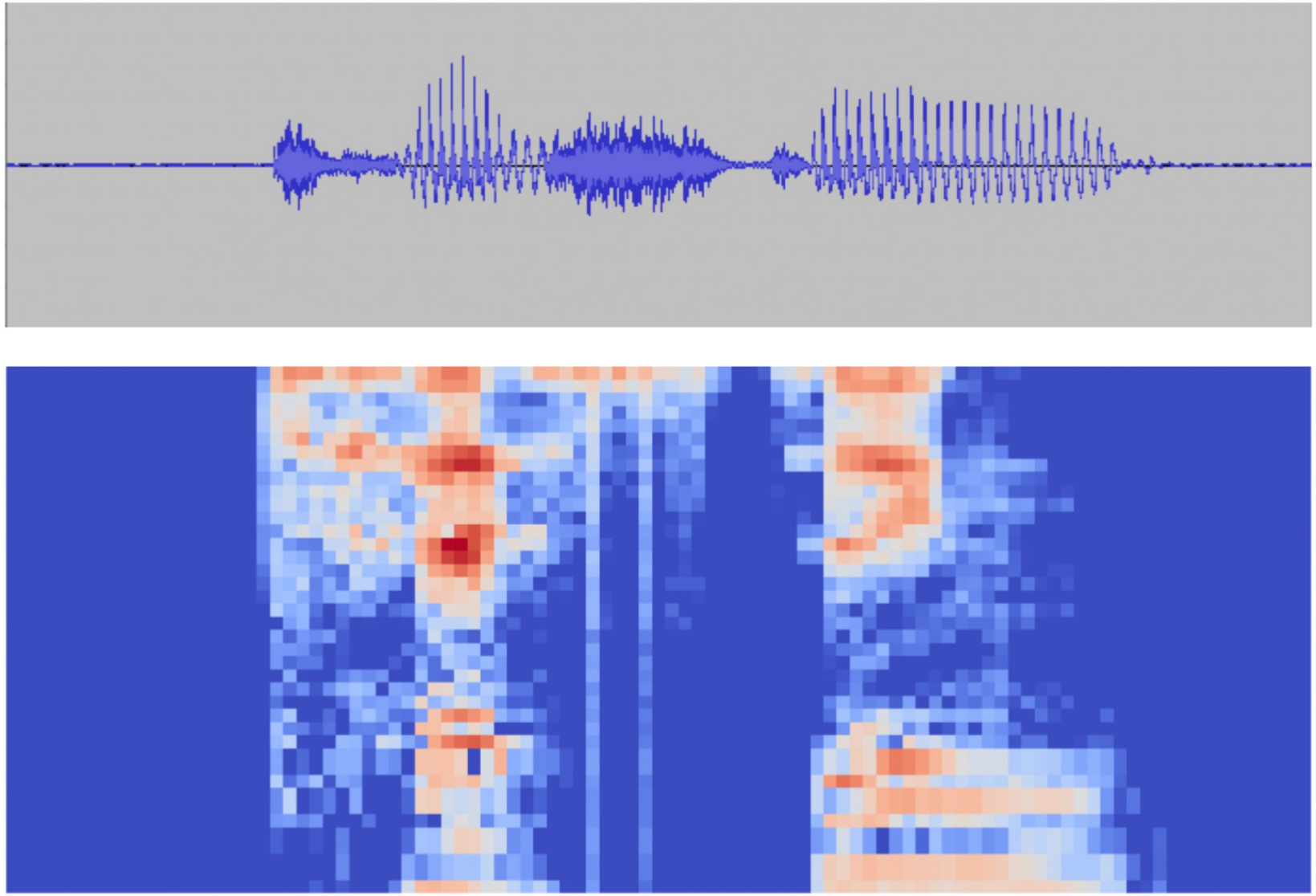
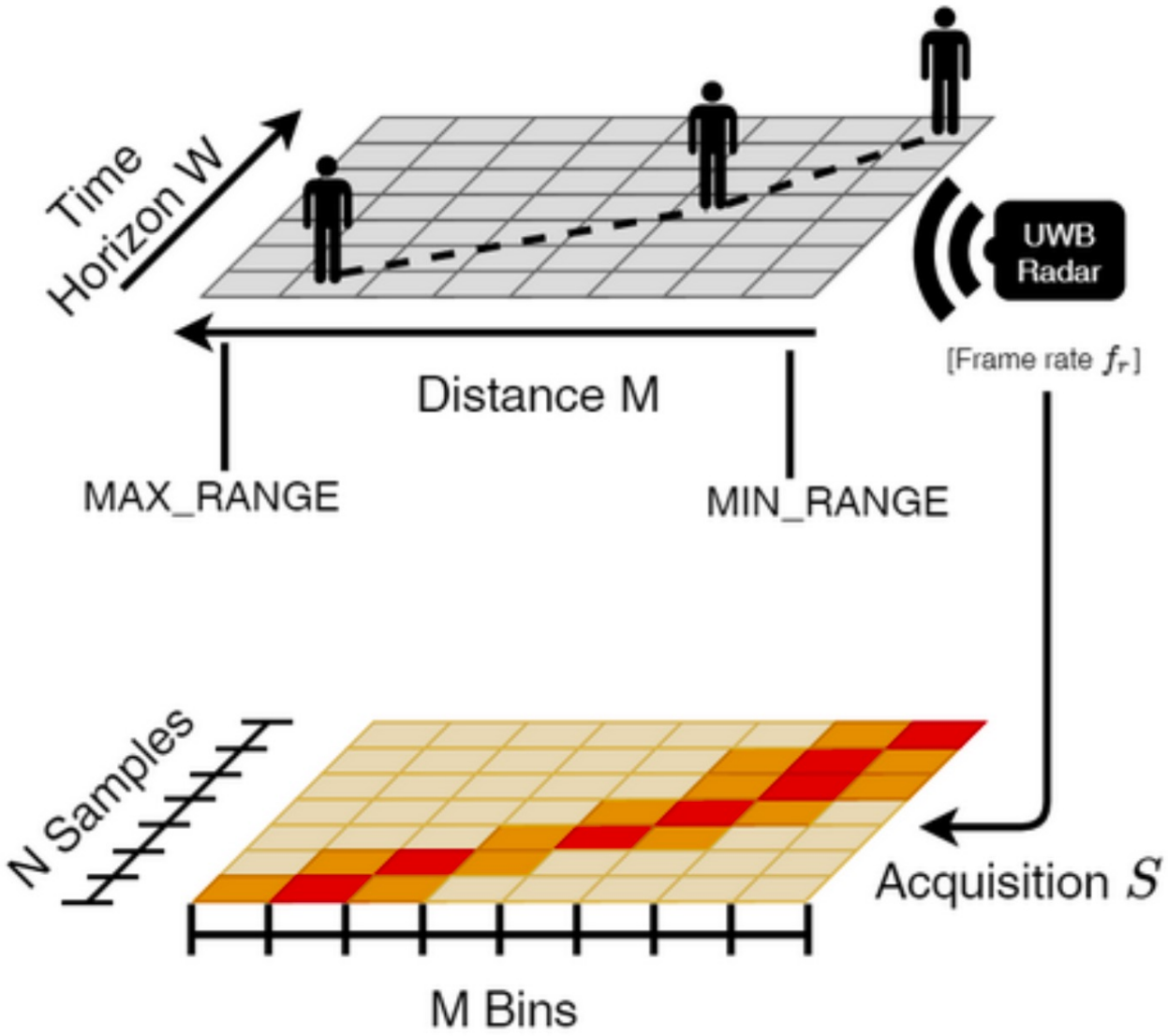


Image taken from [1]

Feature Extraction in the radar domain



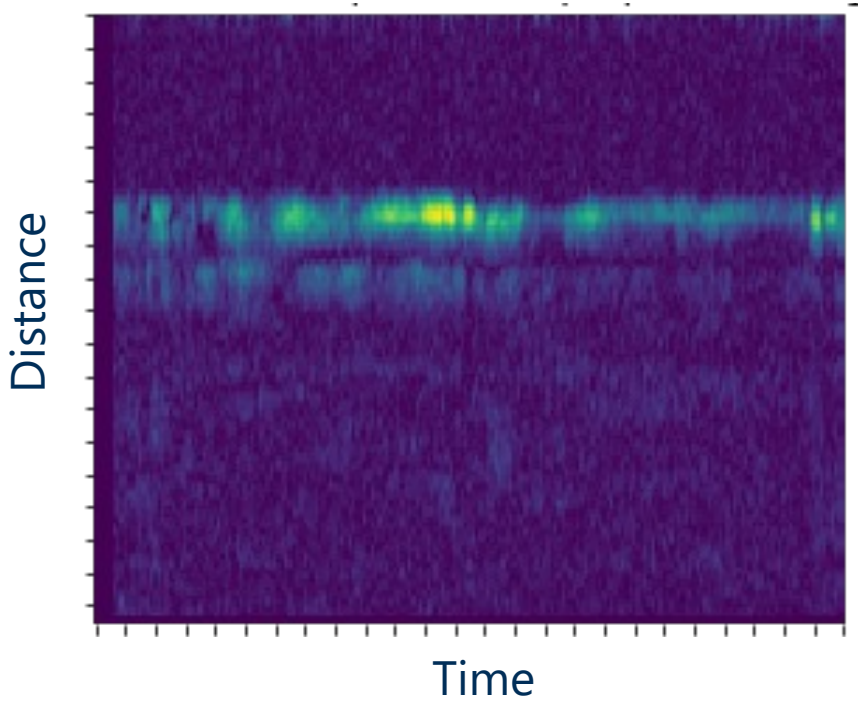


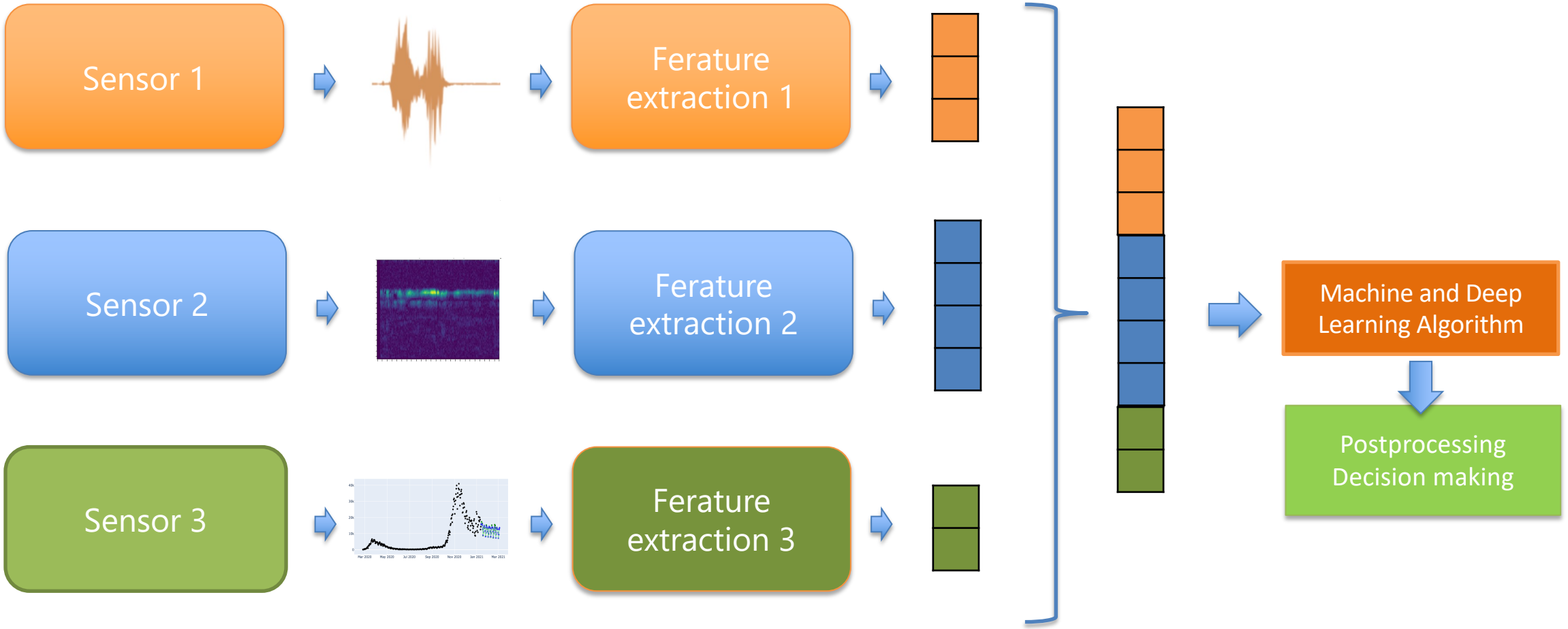
Image feature detection

- Edge detection: highlight boundaries
- Corner detection: find corners in shape
- Blob detection: highlight areas with similar content
- Ridge detection: highlight curves

OpenCV for feature detection (and other image processing tasks) **on SoC**

OpenMV for feature detection **on High-End MCUs**

Combining features and sensors: sensor fusion



Combining features and sensors: normalization and standardization

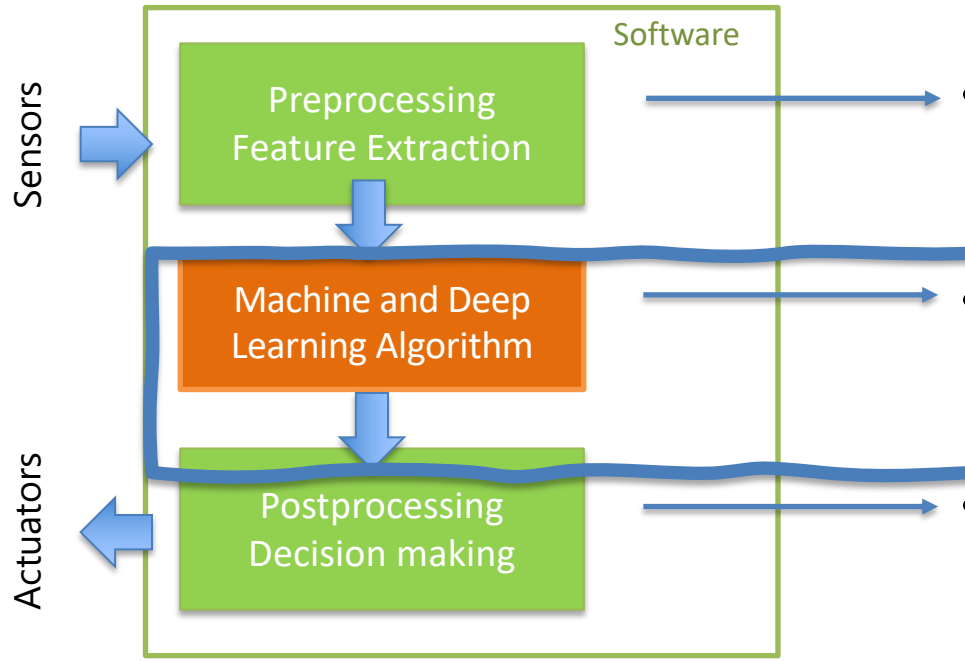
- Features with **different scales** might underfit the training of machine learning
- Min-Max Normalization in the range [0,1]:

$$x_i^{normalized} = \frac{x_i - \underset{i}{\mathrm{MIN}}(x_i)}{\underset{i}{\mathrm{MAX}}(x_i) - \underset{i}{\mathrm{MIN}}(x_i)}$$

- Standardization (removing mean and dividing by the std):

$$x_i^{standardized} = \frac{x_i - \mathrm{MEAN}(x_i)}{STD(x_i)}$$

Algorithms for embedded and edge AI



- Receive in input the stream(s) of data coming from the sensors, extract windows of data, remove noise and extract relevant features
- Apply machine/deep learning algorithms on the preprocessed data to generate and output (e.g., for classification, regression or prediction)
- The output of the algorithm is used to make decision or activate a reaction (e.g., open a door, raise an alarm or transmit a signal)

Algorithms for embedded and edge AI

Machine and deep learning algorithms by **functionality**

Machine and deep learning algorithms by **implementation**

Machine and deep learning algorithms by functionality

- Classification
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Prediction
- Feature reduction

Machine and deep learning algorithms by functionality

- **Classification**
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Prediction
- Feature reduction

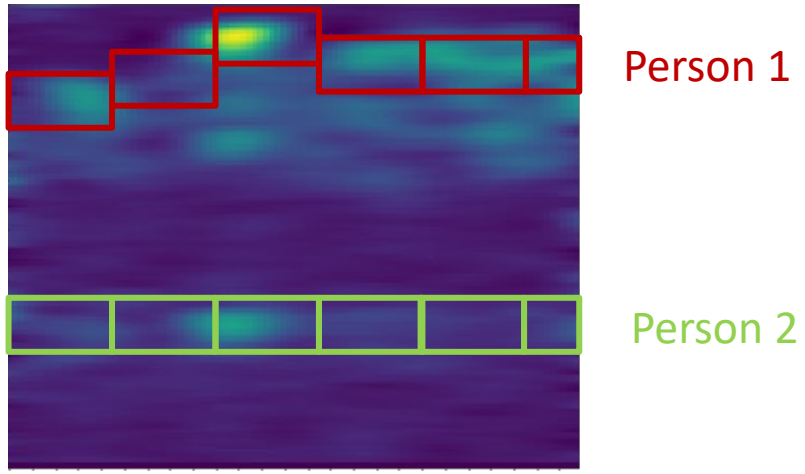
- Classify micro-acoustic emissions of a rock face
- Recognize bird vocalizations
- Classify the behavior of a patient in a retirement house by analyzing radar data

Image taken from [1]

Machine and deep learning algorithms by functionality

- Classification
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Prediction
- Feature reduction

- Measure the level of a liquid in cup
- Estimate the number of people in the room by analyzing radar data



Machine and deep learning algorithms by functionality

- Classification
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Prediction
- Feature reduction

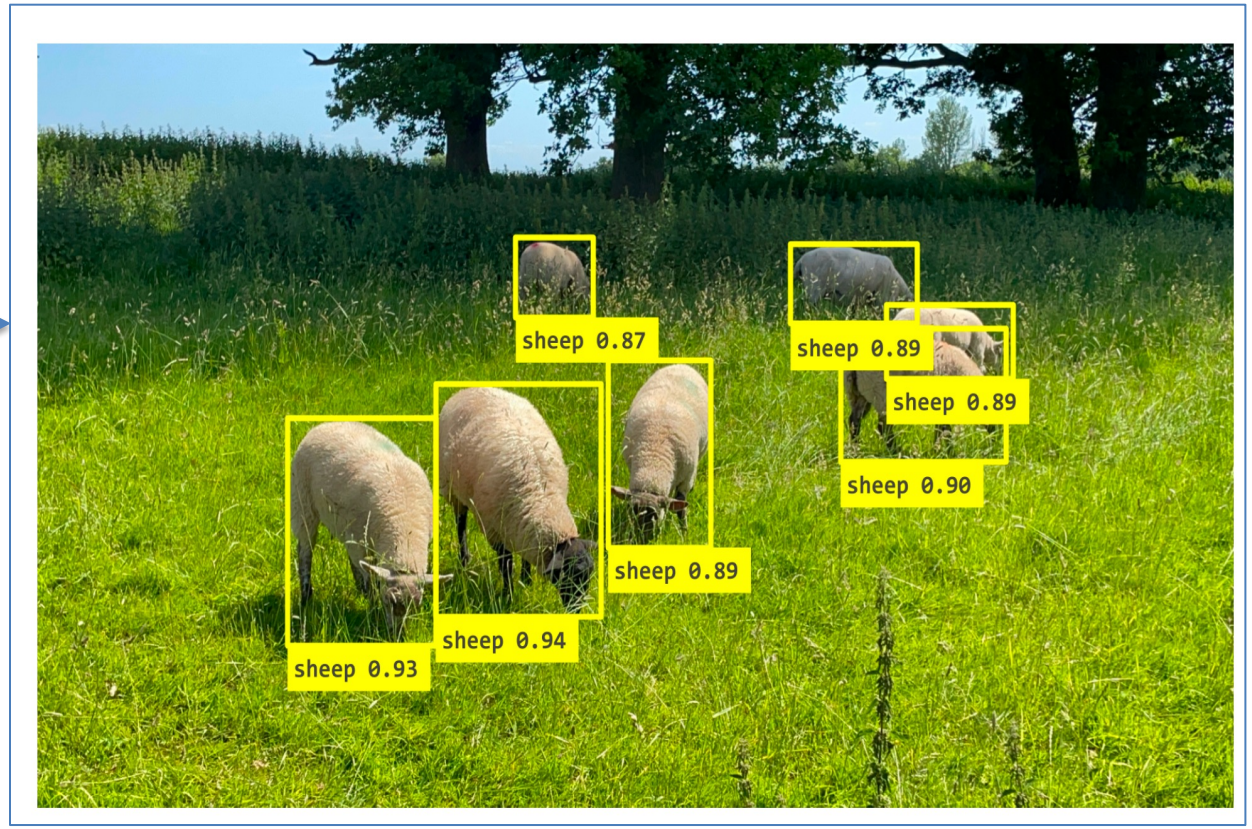


Image taken from [1]

Machine and deep learning algorithms by functionality

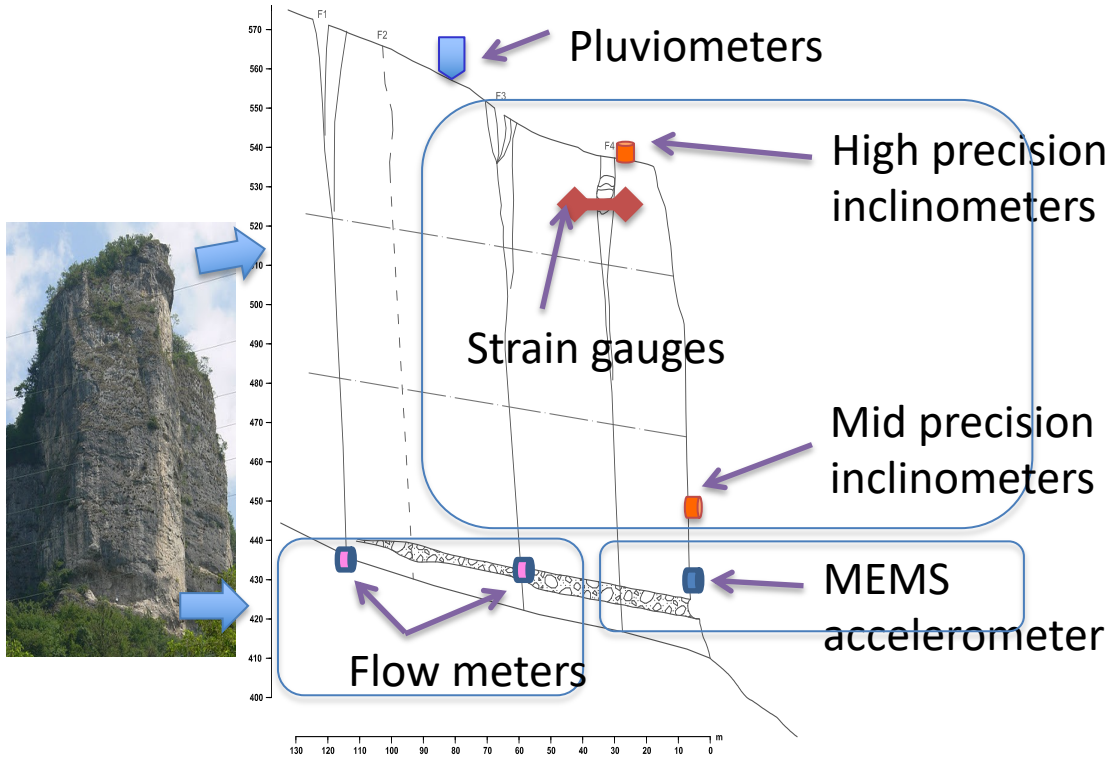
- Classification
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Prediction
- Feature reduction

Image taken from [1]

Machine and deep learning algorithms by functionality

- Classification
- Regression
- Object detection
- Segmentation
- **Anomaly detection**
- Prediction
- Feature reduction

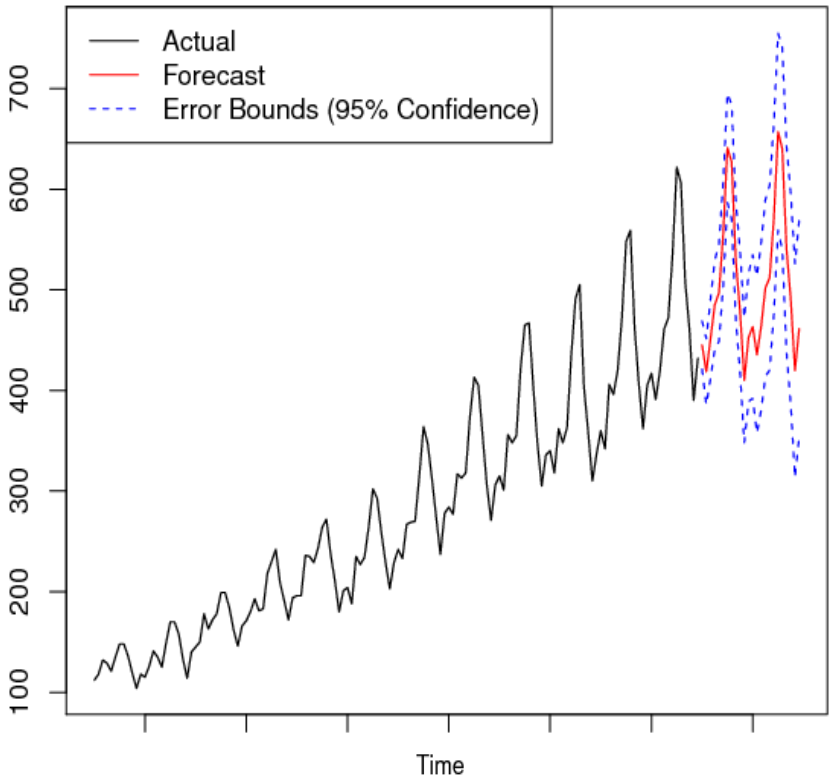
- Detect persons in a conveyer belt
- Detect micro-acoustic emissions
- Detect enlargement of fractures



Machine and deep learning algorithms by functionality

- Classification
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Prediction
- Feature reduction

- Predict the next value of a time series (e.g., temperature, humidity, power consumption, etc.)



Machine and deep learning algorithms by functionality

- Classification
- Regression
- Object detection
- Segmentation
- Anomaly detection
- Prediction
- Feature reduction

- Compress or extract features from audio chunks to reduce the size

Algorithms for embedded and edge AI

Machine and deep learning algorithms by **functionality**

Machine and deep learning algorithms by **implementation**

Machine and deep learning algorithms by implementation

- Conditional logic
- Machine Learning
- Deep Learning

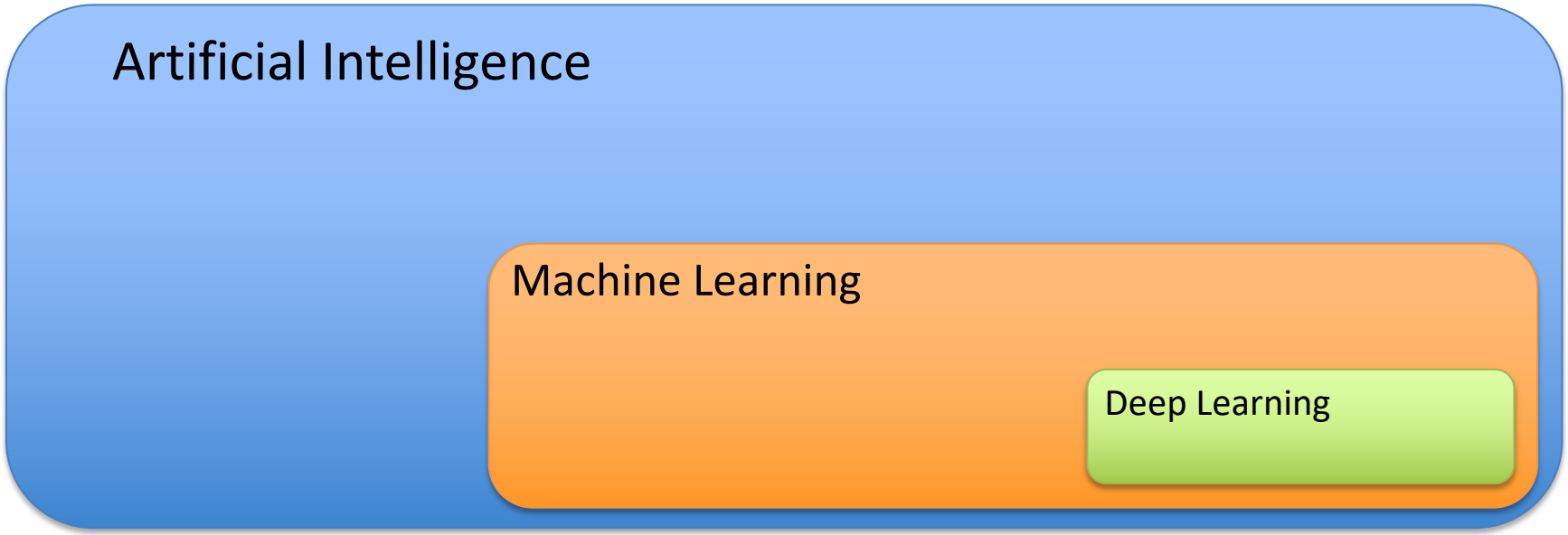
Conditional logic

```
current_speed = 10 # In meters per second
distance_from_wall = 50 # In meters
seconds_to_stop = 3 # The minimum time in seconds required to stop the car
safety_buffer = 1 # The safety margin in seconds before hitting the brakes

# Calculate how long we've got before we hit the wall
seconds_until_crash = distance_from_wall / current_speed

# Make sure we apply the brakes if we're likely to crash soon
if (seconds_until_crash < seconds_to_stop + safety_buffer):
    applyBrakes();
}
```

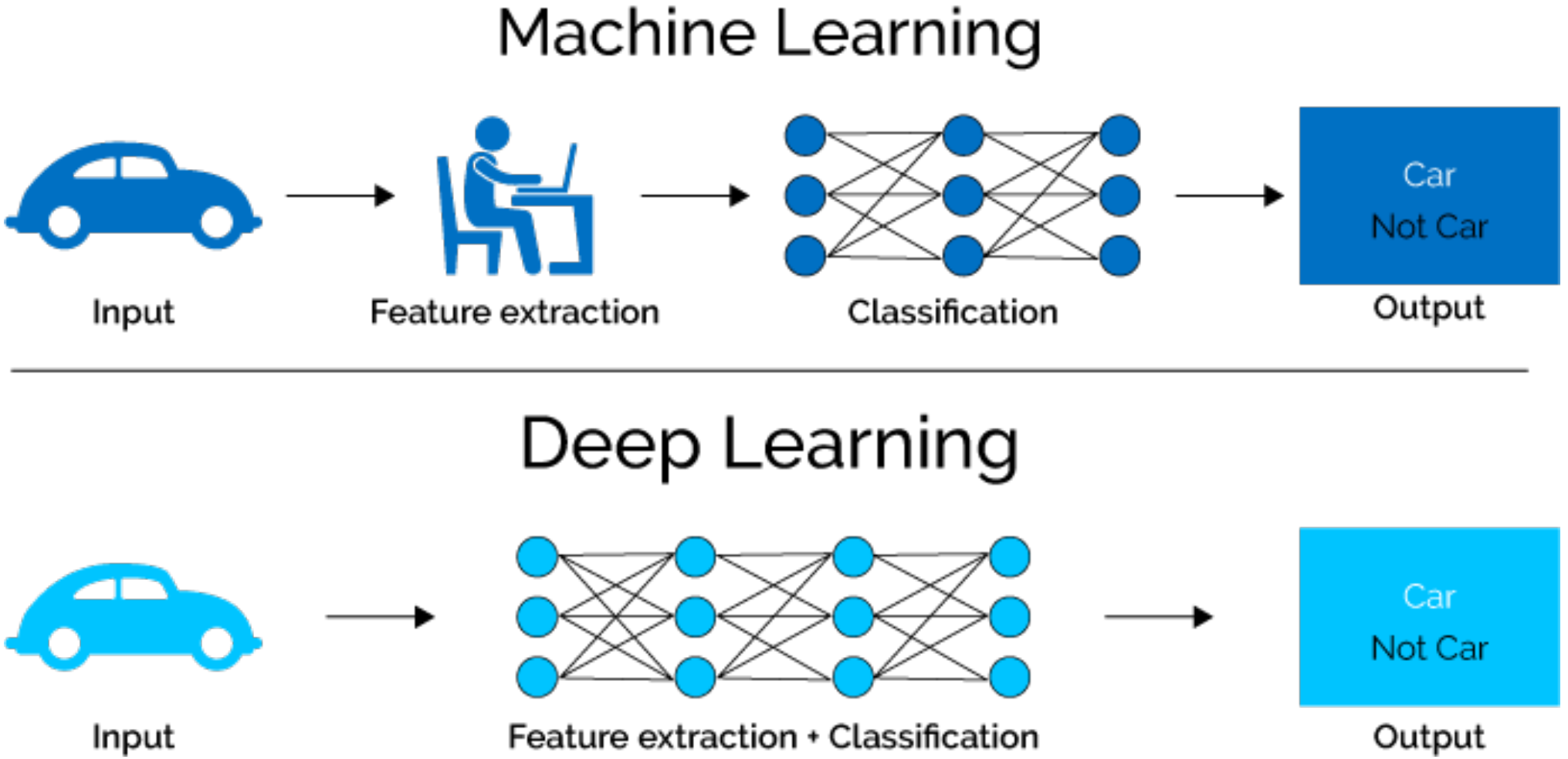
- Deterministic
- Efficient
- Do not require training data

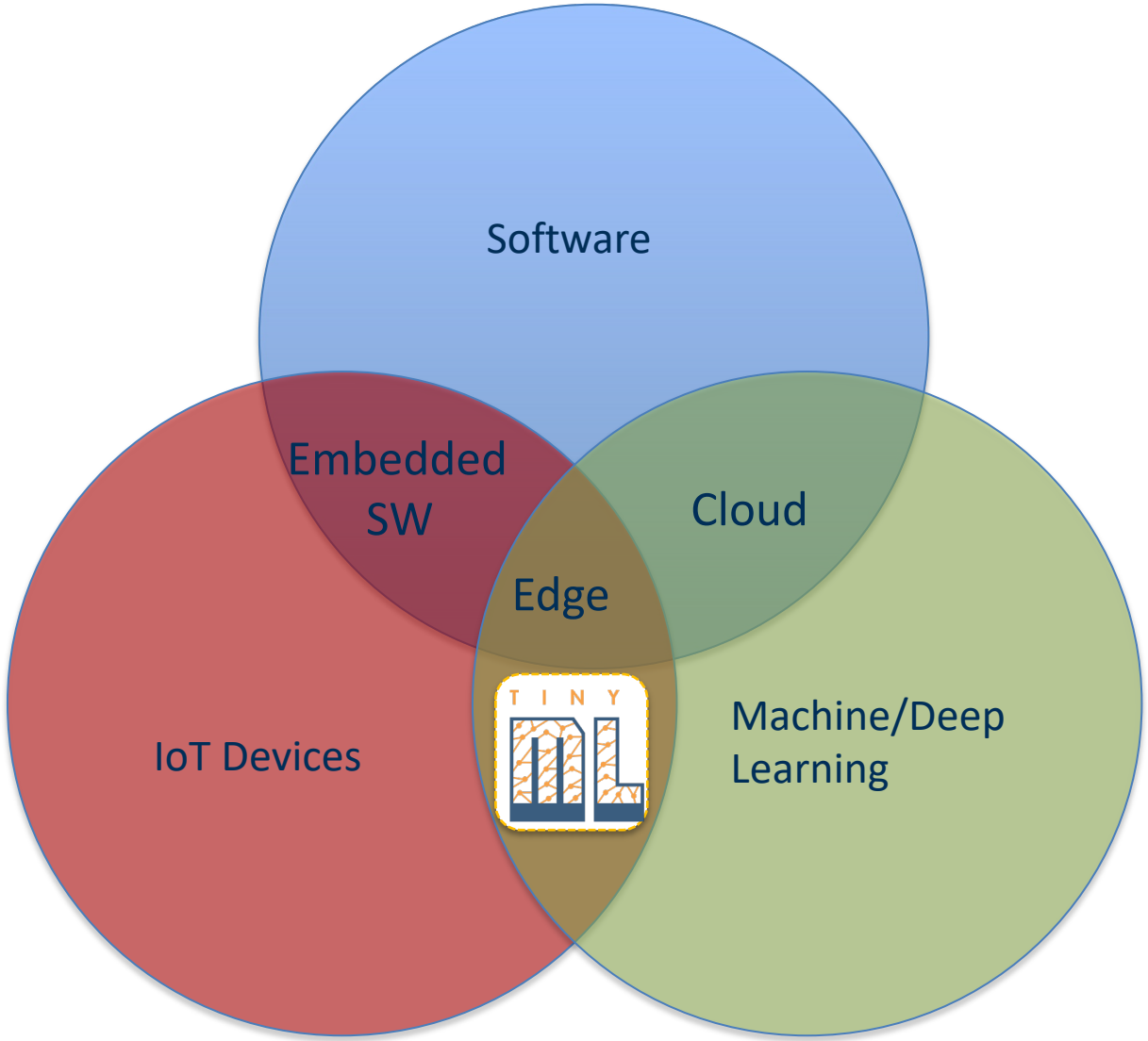
Image taken from [1]

Machine Learning and Deep Learning (and AI)

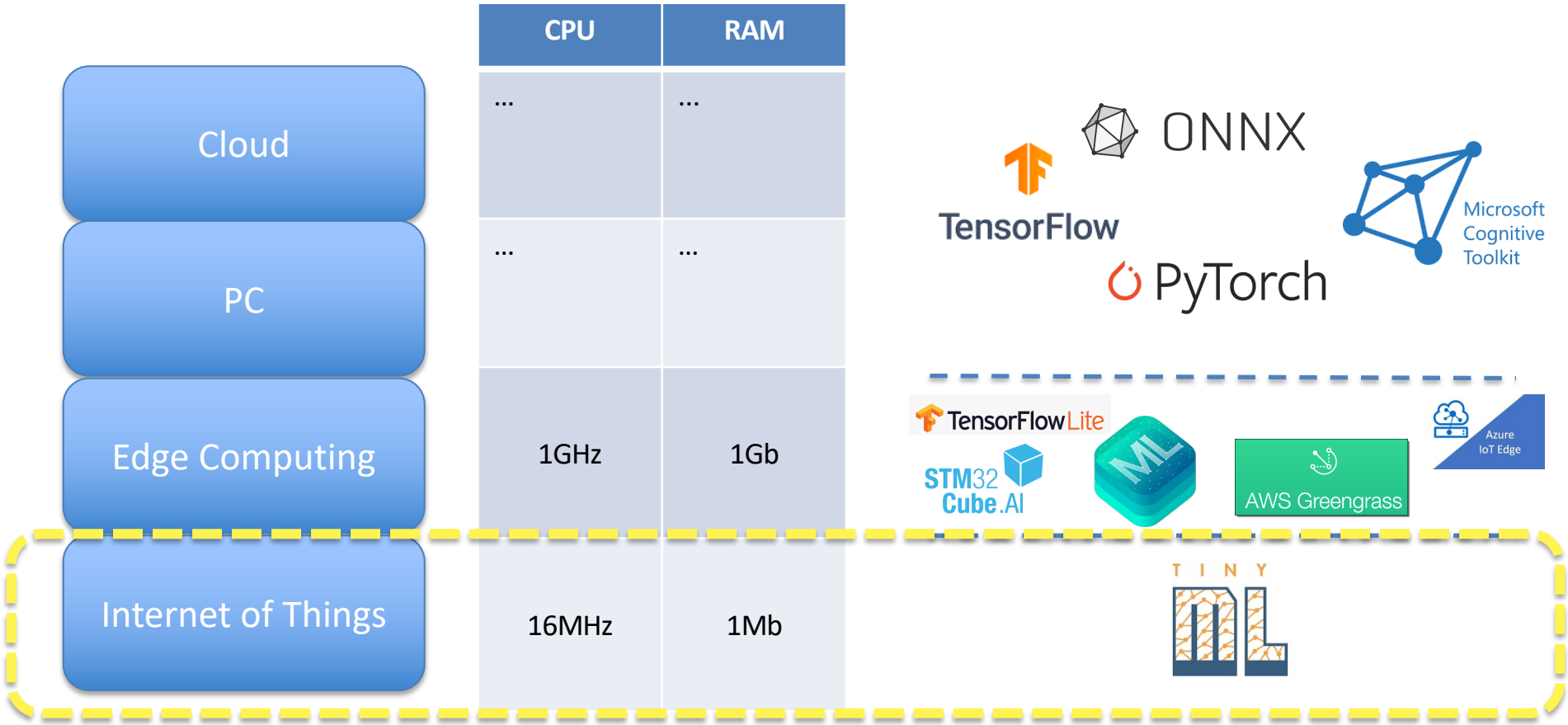


Machine Learning and Deep Learning (this time w/o AI)




Software
Embedded SW
Cloud
Edge
IoT Devices
Machine/Deep Learning
TINY ML

Heterogeneous hardware platforms for ML/DL



How does TinyML work?

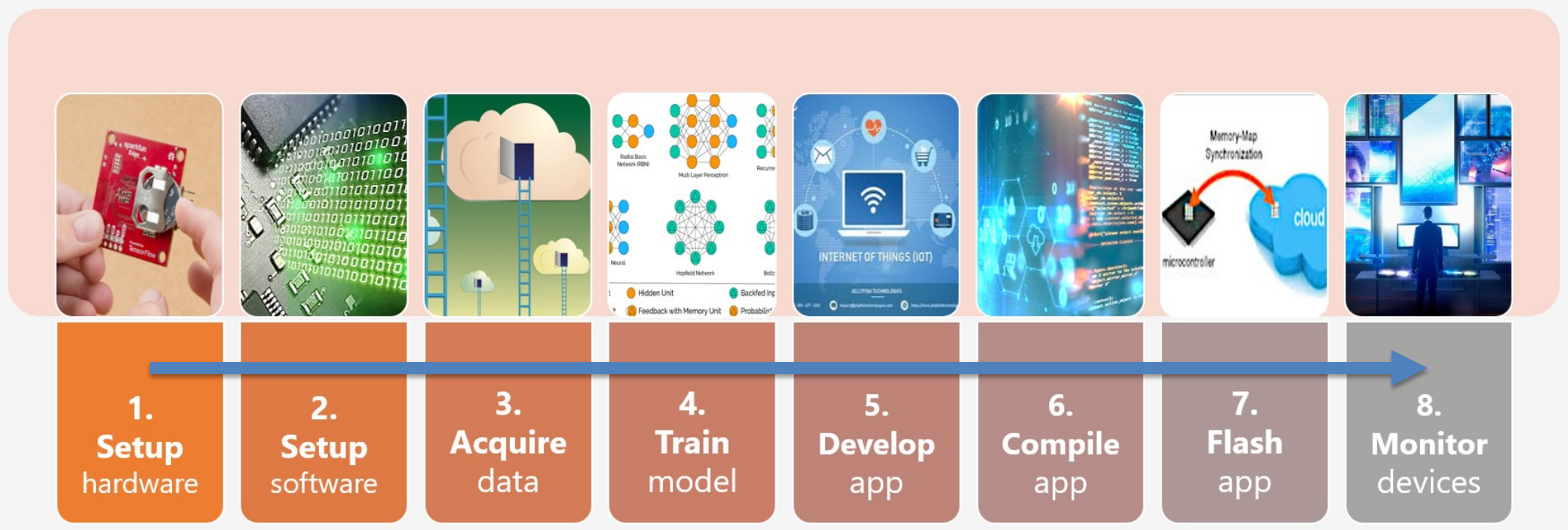


Image from Medium.com “Introducing cAInvas for TinyML devices”, July 2020

To train or not to train…. this is the problem with TinyML

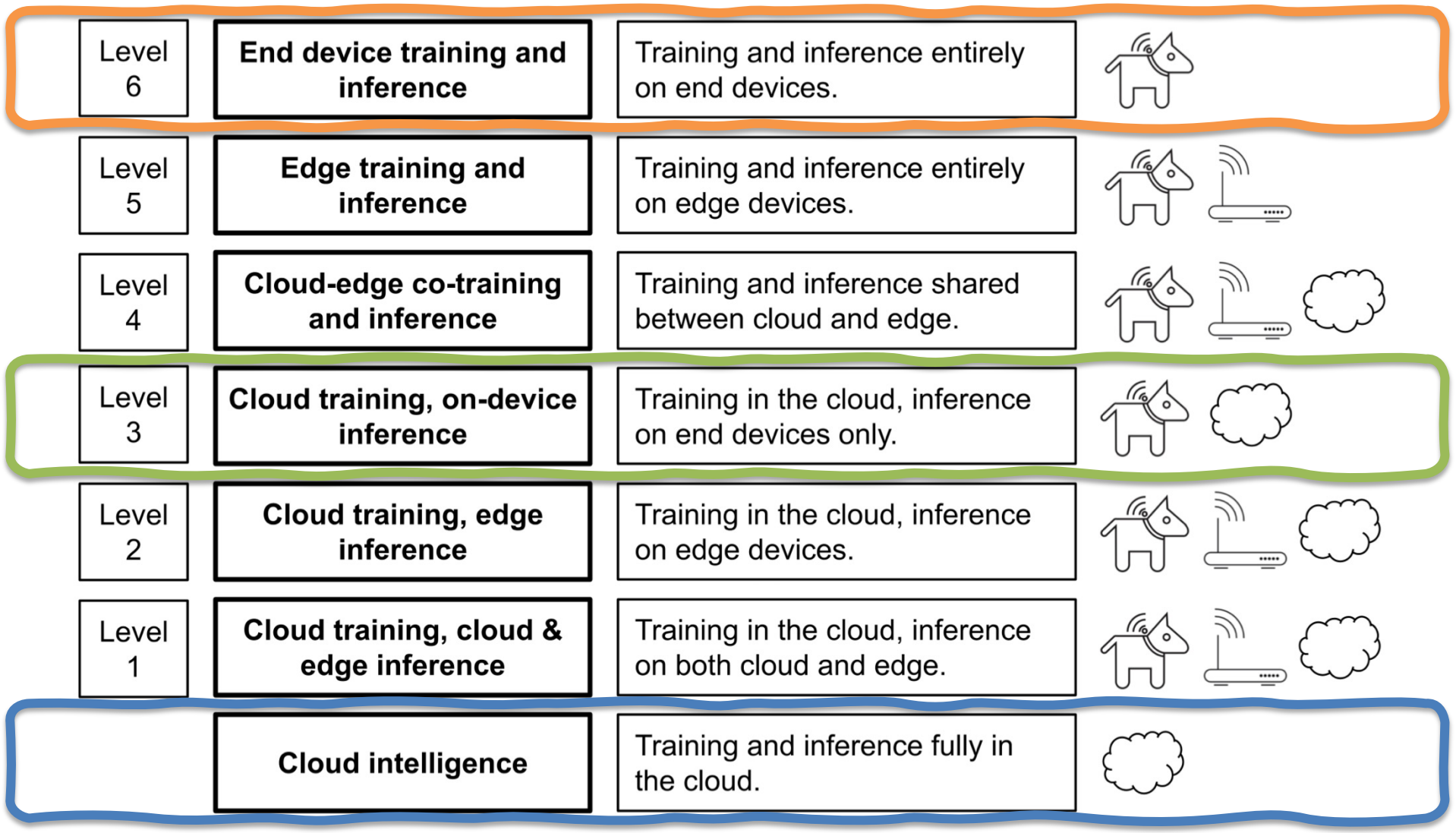


Image taken from [1]

In the next two lectures we will see how to design machine and deep learning solutions for TinyML