Ingredients of An Embedded AI Solution

Software as Main course with Hardware as a Garnish and AI as Condiment

AI Hardware – Munich Urban Colab

Andreas Neumeier, Shubham Sharma, Sebastian Prebeck, Andrew Stevens, Vijay Deep Bhatt, Volkan Esen, Wolfgang Ecker

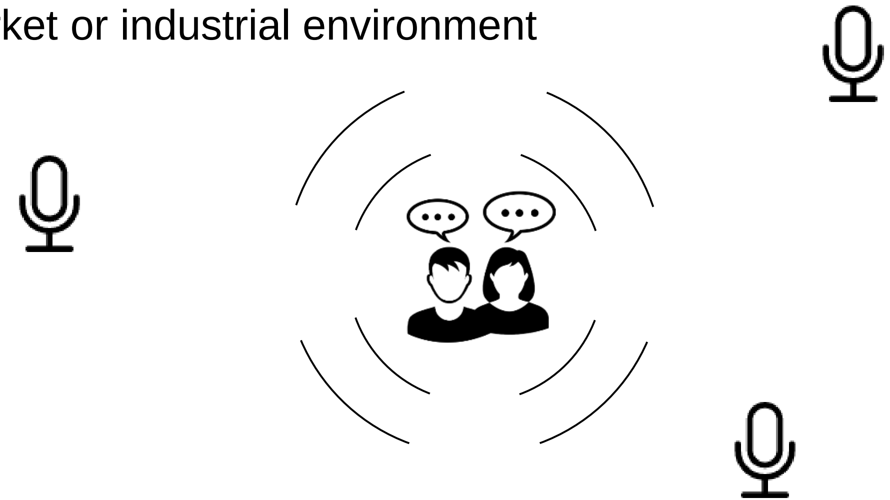
Outline

EdgeAI

- Goal
- The Solution
- First Analysis
- Final Solution
- Summary

Goal

- Audio source localization is an interesting topic nowadays
  - E.g. supermarket or industrial environment

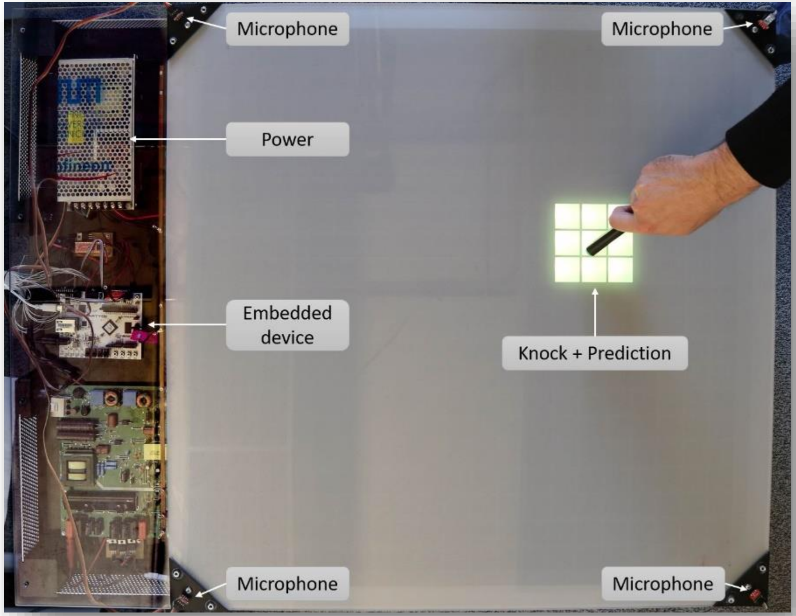


- Why not use ML and an embedded device to do that?
  - Embedded devices are extensively used in industrial environment due to low cost
  - Problem: limited hardware resources

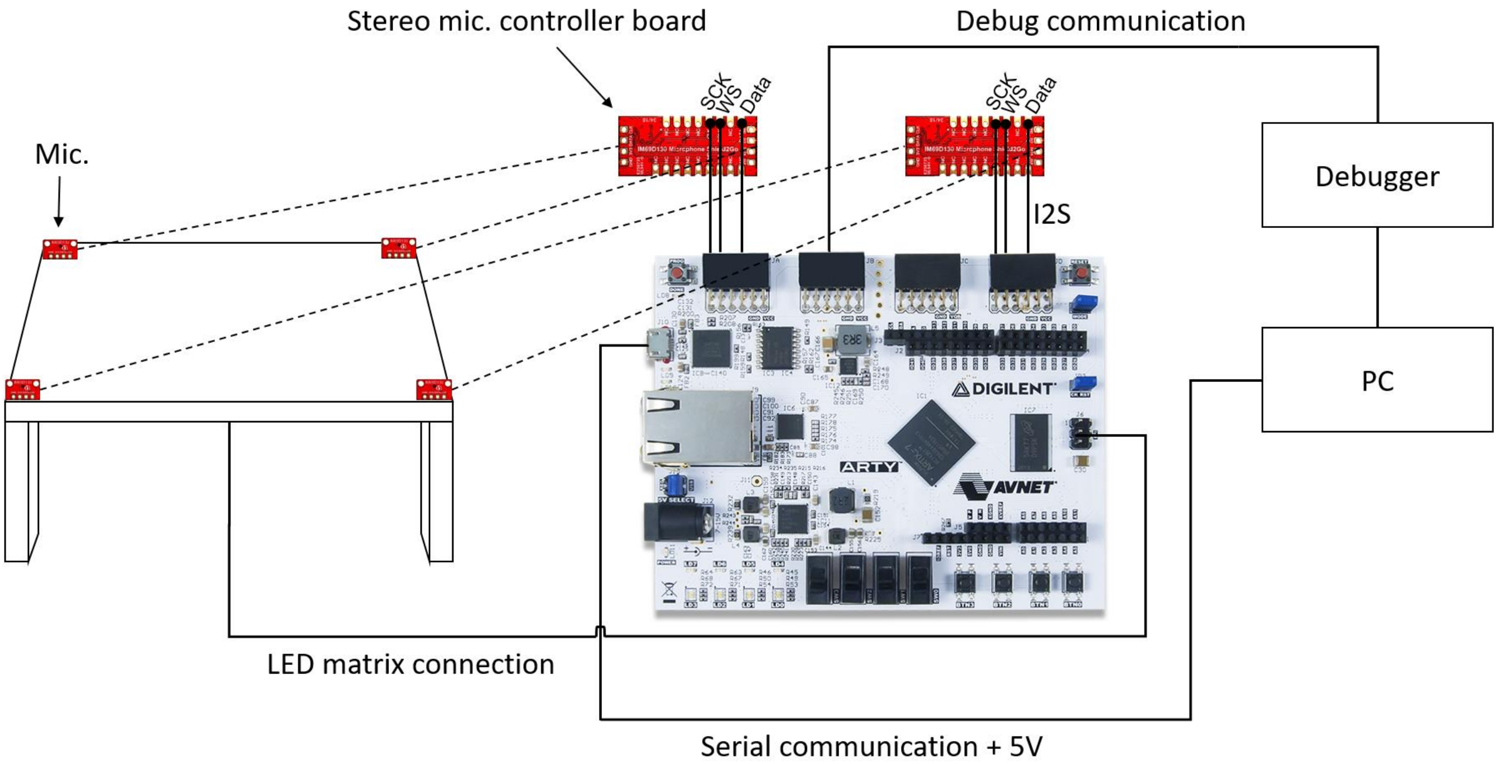
Proof – of -Concept

› Setup:
  - 4 synchronized microphones
  - 1 FPGA running a RISC-V CPU with special ML instructions
    - 64 KBytes RAM
    - 64 KBytes ROM
  - 20 x 20 LED matrix

› Use the setup for
  - Data collection
  - Proof-of-Concept



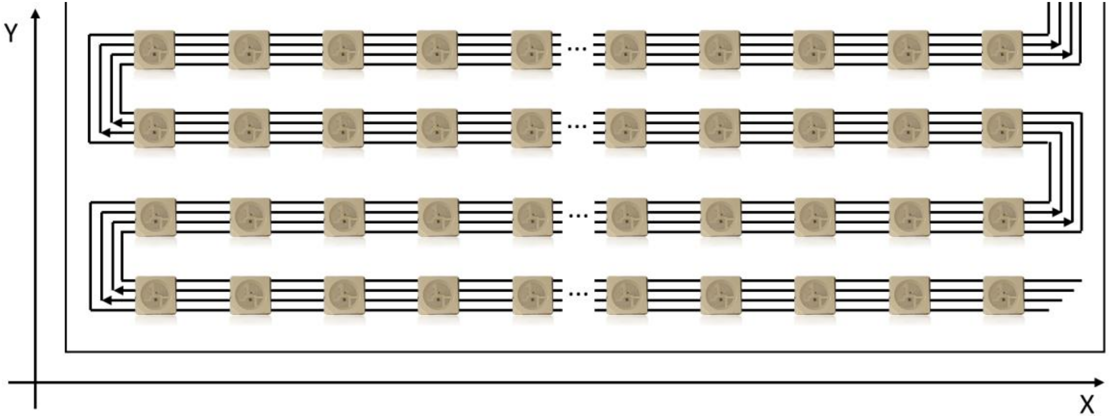
Setup



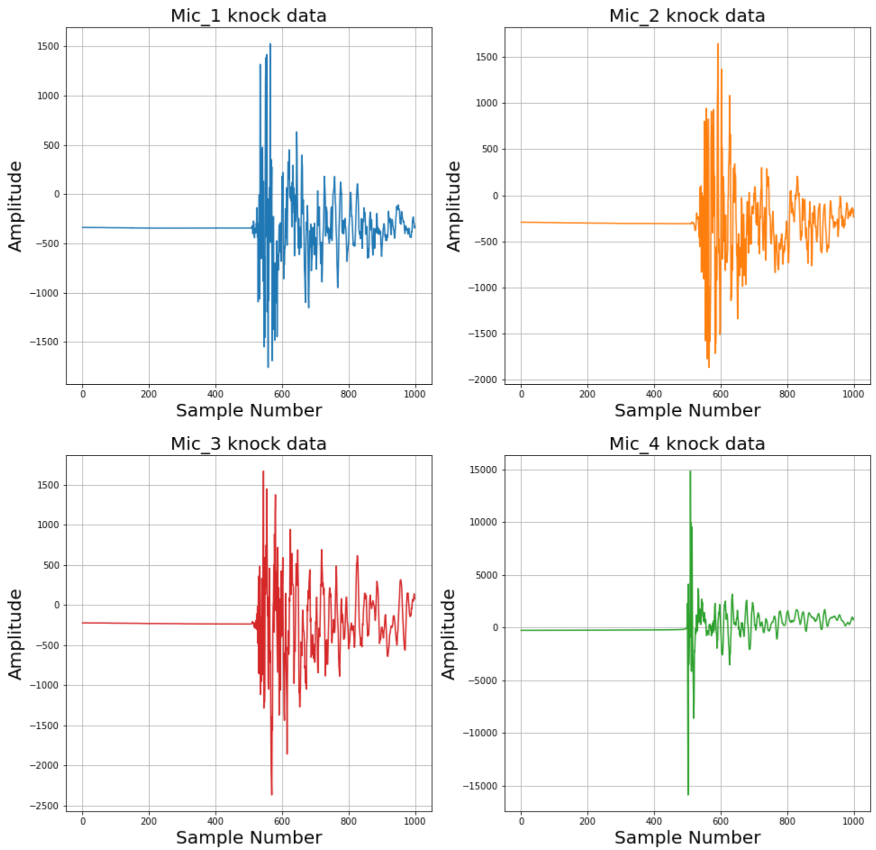
Stereo mic. controller board
Debug communication
Mic.
SCK
WS
Data
SCK
WS
Data
I2S
Debugger
PC
LED matrix connection
Serial communication + 5V

Capturing the dataset

- Coordinate system with origin at the left bottom of the LED matrix pad
- Capturing process
  - Random permutation of all LEDs is created
  - A LED lights up
  - A knock is recorded
  - The audio data is sent to the PC for further processing



Signal characteristics



- Plot of sample audio data corresponding to position (20,1)
- High Amplitude for Mic 4
- Less Echo for Mic 4

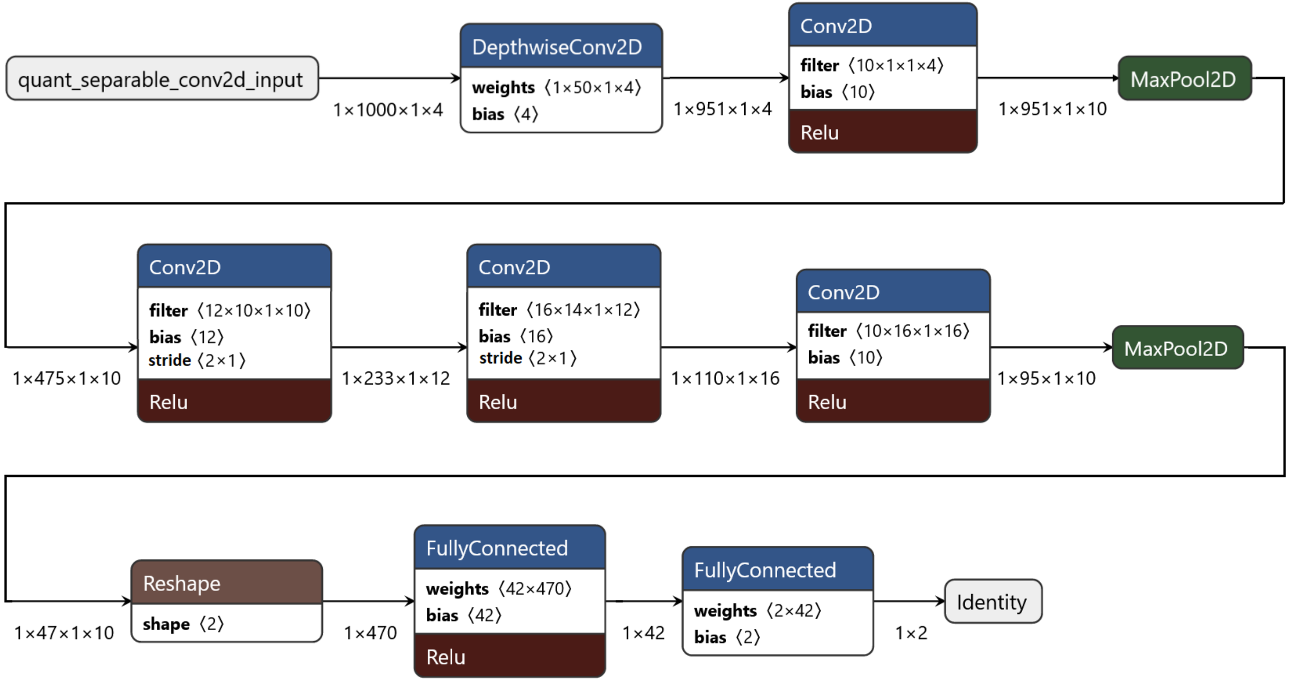
Outline

EdgeAI

- Goal
- The Solution
- First Analysis
- Final Solution
- Summary

AI model

ROM	51 KB (78%)
RAM	17 KB (27%)
Instructions	13 Mio
Time	325 ms
Avg. Error	1.25 cells



AI model architecture

Convolutional
- Well understood & easy to compute
- Supported in ML frameworks for edge devices (e.g. TFLite)
- Best accuracy

Depthwise convolutional
- Adds to the concept of convolution
- Reduces the amount of expensive multiplications

Temporal convolutional
- Causal convolutions
- Brings time dependency into CNNs
- Not supported well in TFLite

How to fit an AI model in 64KB RAM and 64KB ROM (1)

- Reduce model complexity
  - Use stride where possible
  - Use max pooling if strides don’t perform well
  - Replace multiplication heavy convolutions with depthwise convolutions
  - Analyze the model in regards to under- and overfitting

- Use quantization
  - 32 Bit float if no quantization is used
  - 16 Bit int quantization: 50% memory reduction
  - 8 Bit int quantization: 75% memory reduction

- Do scalable pruning

	ROM usage	**RAM usage**
Final AI model	51KB (78%)	17KB (27%)

How to fit an AI model in 64KB RAM and 64KB ROM (2)

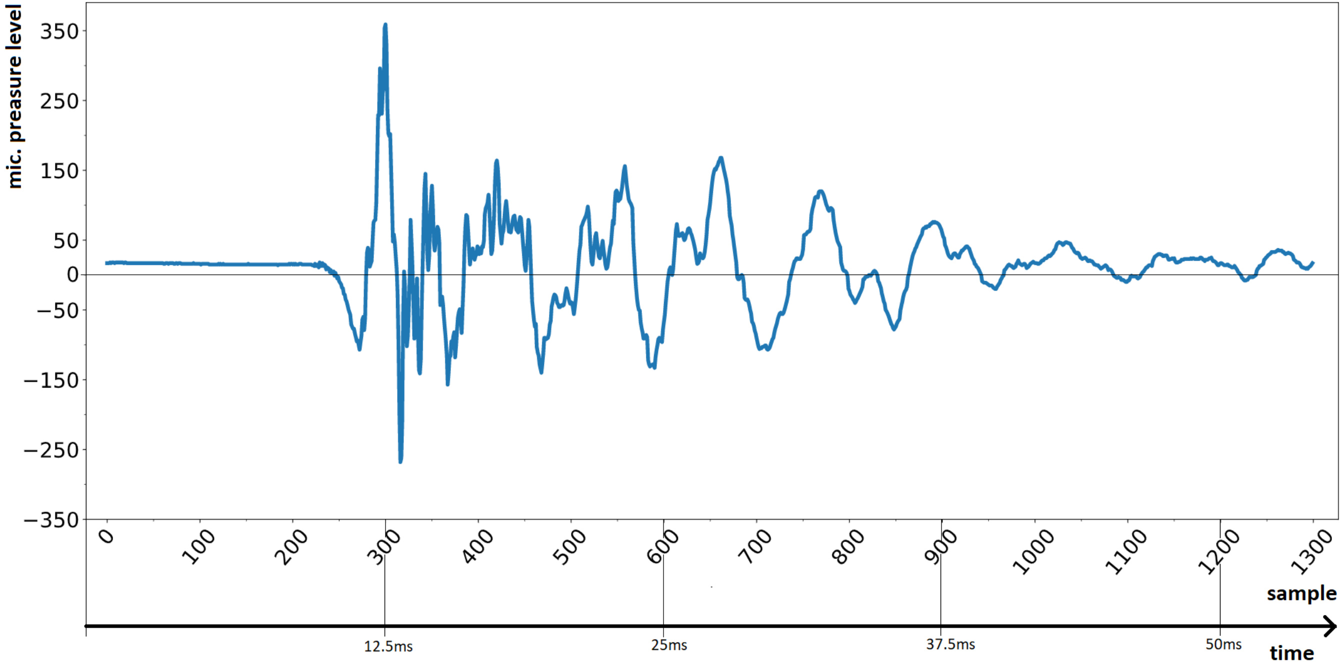
- Tensor Flow Light Interpreter is with several 100k binary code size way to big
- Use Tensor Flow Lite Micro
- Compile Model
  - Apply compression techniques to weights
  - Use pre-interpreter to identify required functions and re-implement to support compression
  - Generate Skeleton
  - Merge minimum number of required functions with Skeleton and Compile
  - Consider compression in hardware accelerator
- Use TVM in future

Outline

EdgeAI

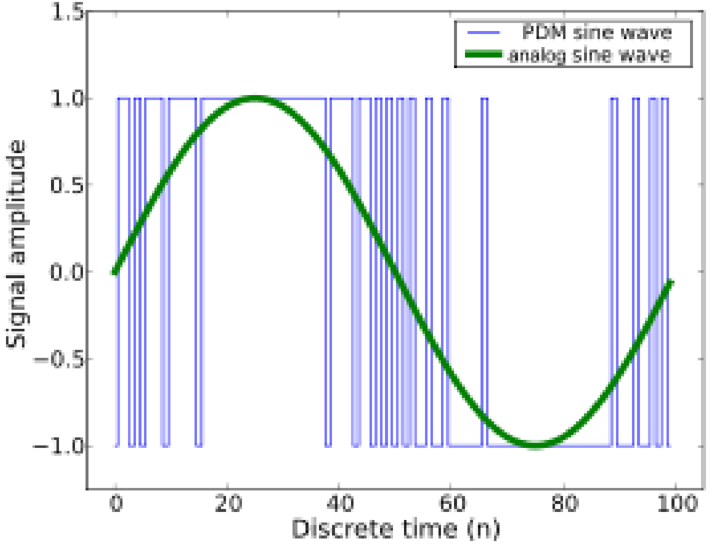
- Goal
- The Solution
- First Analysis
- Final Solution
- Summary

Accuracy was lousy



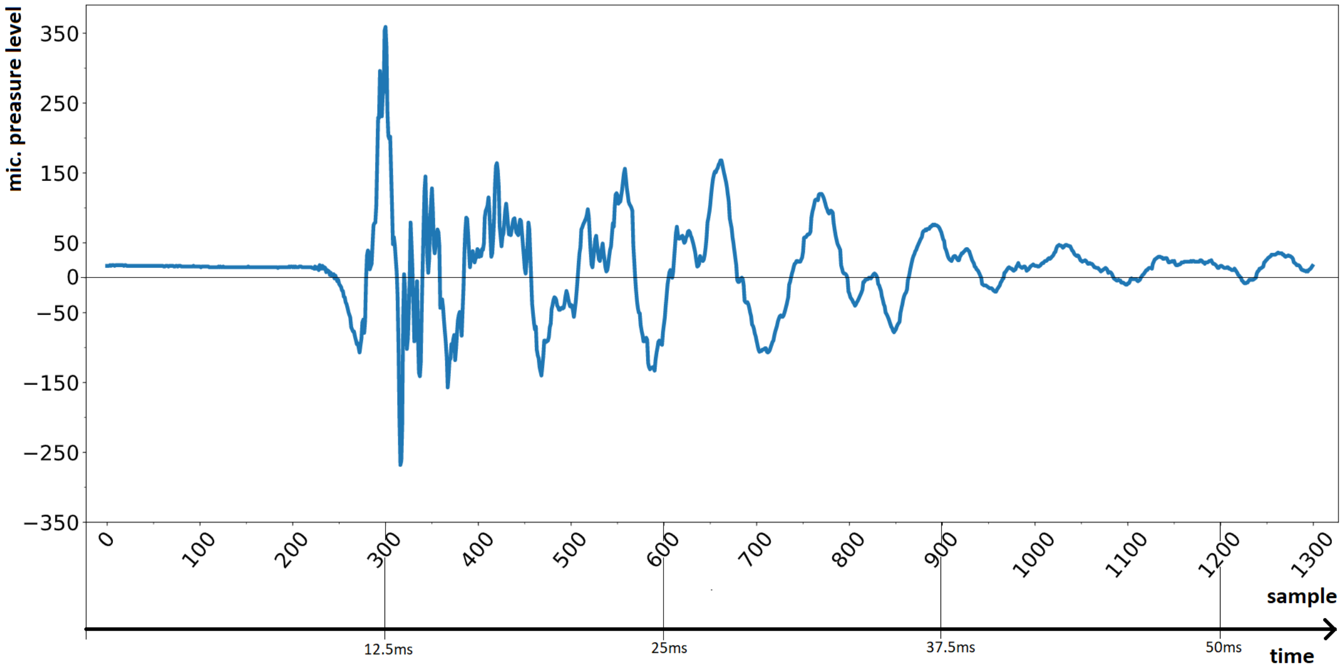
Accuracy was lousy
Analog Signal Analysis showed noisy data signal

- I^2S Data Line Signal was partially corrupted
  - Replaced by PDM



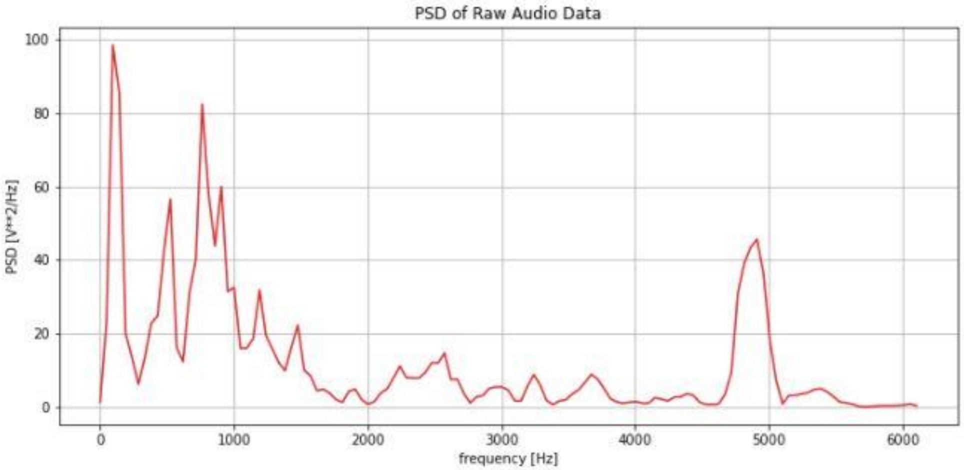
Source:
https://en.wikipedia.org/wiki/Pulse-density_modulation

Accuracy was less lousy but still not good

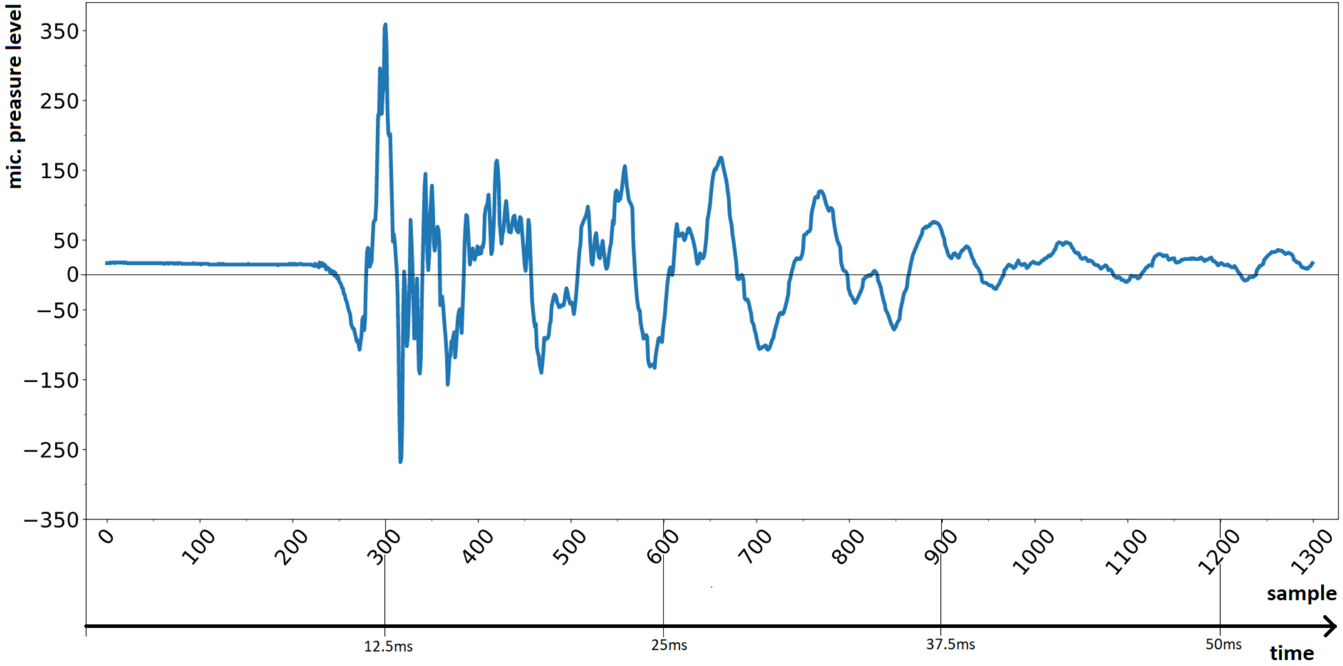


Accuracy was less lousy but still not good
Moving to Power Spectral Density

- Time Series of input values (sound) was transformed to power spectral density
- Result, i.e. nose localization, got worse

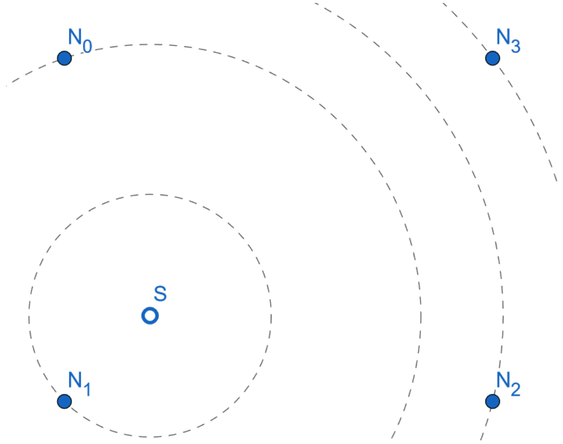


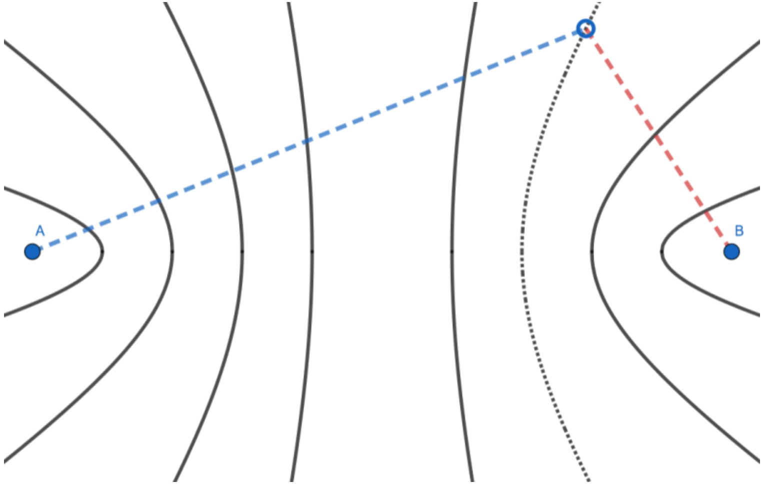
Accuracy was less lousy but still not good



Accuracy was less lousy but still not good
Non ML-Solution: Time Difference of Arrival (TDoA)

- Circles can not be used for localization => use hyperbolas
  - Time Difference of Arrival can be measured

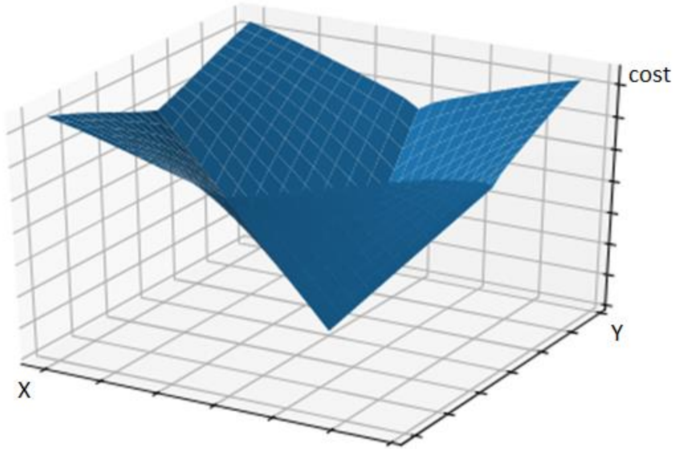




Time Difference of Arrival (TDoA)

- A cost function can be created
  - Mic. positions are known
  - Time Difference of Arrival is known

- Computation costly but lest accurate than ML



Machine learning (ML) vs. Conventional

- ML provides better accuracy
  - Possible reason: Echo noise from the PVC surface of the LED matrix pad

Distance in cells	0	1	2	3	4	5
ML (hit rate)	24%	78%	91%	96%	99%	99%
Conventional (hit rate)	1%	10%	20%	34%	64%	81%

- Conventional is faster
  - Neural networks are computational heavy

	Instructions	Execution time
ML	~13 Mio	~325 ms
Conventional	~2,1 Mio	~53 ms

Outline

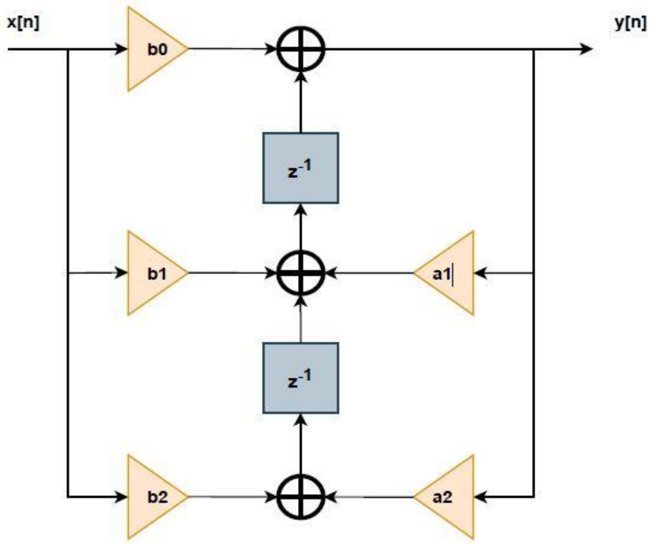
EdgeAI

- Goal
- The Solution
- First Analysis
- Final Solution
- Summary

A Mixed NN and DSP solution
Applying a High Pass Filter on the Inputs (1)

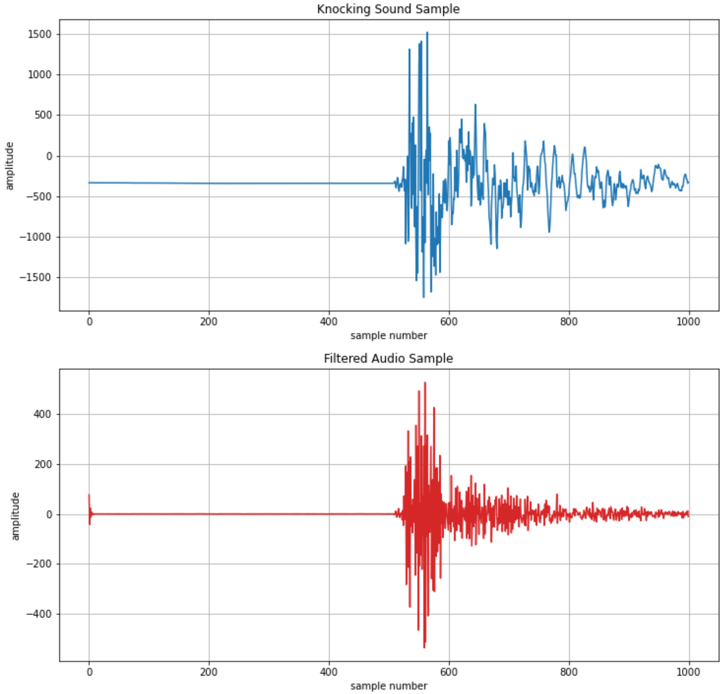
- Each Biquad stage implements a second order filter using the difference equation:
  - $b0 * x[n] + d1$
  - $d1 = b1 * x[n] + a1 * y[n] + d2$
  - $d2 = b2 * x[n] + a2 * y[n]$

- Coefficients calculated using Octave.
  - b0 = 0.3045
  - b1 = -0.3045
  - b2 = 0
  - a1 = -0.3910
  - a2 = 0



A High Pass Filter Biquad Cascade II T Filter with Direct Form II Transposed Structure

A Mixed NN and DSP solution
Applying a High Pass Filter on the Inputs (2)



A Mixed NN and DSP solution
High Pass Filter substantially improved the results

Model Type	Mean Cell Distance	0 Cell	1 Cell	2 Cell	3 Cell	4 Cell
CNN without HP Filter	1.8	5.83%	48.33%	84.16%	94.16%	97.48%
CNN with HP Filter	1.4	10.10%	75.10%	91.66%	97.49%	99.99%

Cumulative Accuracy comparison for CNN model with the usage of High Pass Filter in place v/s raw audio signal. A total of 1696 training audio *datapoints*, trained for 50 epochs and learning rate of 0.01 were used. The cell results are based on 120 test *datapoints*

Open Question: Why could‘t this be achieved already with CNNs?

Outline

EdgeAI

- Goal
- The Solution
- First Analysis
- Final Solution
- Summary

Software as Main course with Hardware as a Garnish and AI as Condiment

- Findings on ML application
  - *The data driven approach with ML was able to deal with body noise – it "detected" that effect*
  - *The best solution could be achieved with a mixed ML/DSP solution*

- Most effort went into SW development
  - *Software Infrastructure for Micro including runtime system, driver, application code*
  - *ML compiler extension*
- Second most effort went in HW development
  - *Setting up the system (!), building the SoC, building the AI accelerator*
- Least effort went into ML
  - *Most hereof was data collection and running experiments*