A Proposed Realtime Psychoacoustic Anomaly Detection On The Edge

Course: Problems in Machine Learning // ECE-551 // Spring 2023

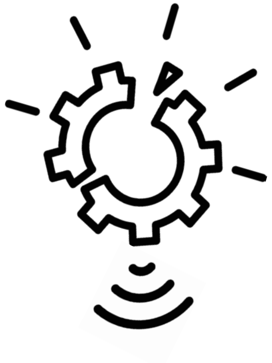
Prepared by: Silas Curfman

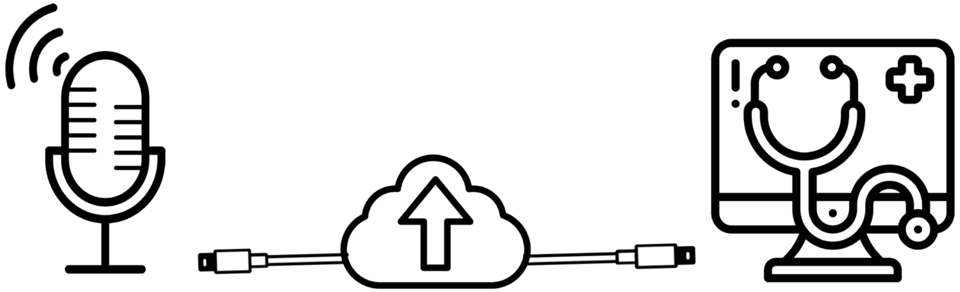
Instructor: Dr. Marios Pattichis

ELECTRICAL & COMPUTER ENGINEERING

Outline

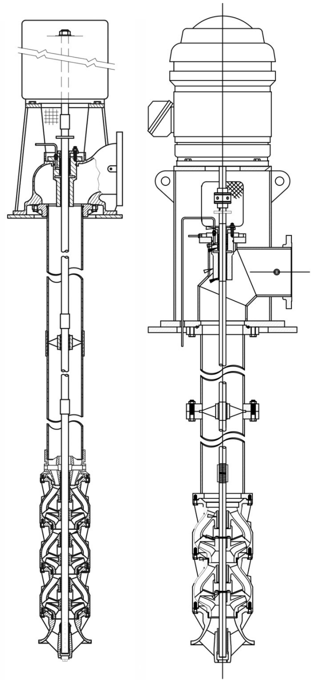
I. Proposed Problem
II. Motivation
III. Background and History
IV. State of the Art
V. Emerging Methods
VI. Demonstration
VII. Future Modifications
VIII. Conclusion
IX. Question & Answer





Case: Vertical Turbine Pump Stations

1,250 HP Pumps, 460 V 3ph Motors,



Multi-Unit Installations, PLC Network Controlled

Very Remote Location, Harsh Conditions, Demanding Uptime, Strict Maintenance Window / Season

Motivation

- Proposed Use Case: Monitoring Large Scale Irrigation in Remote Locations
  - Critical Infrastructure
  - Edge / Remote Locations
  - HITL (Human In The Loop)
  - Very High Economic ROI
  - HVA (Cyber:High Value Asset)

- Cross Domain Case Study
  - Machine Learning
    - Digital Signal Processing
    - Statistical Learning
    - Deep Learning
  - Embedded Systems
    - Low Power IoT
    - Edge / Tiny ML
  - Machine Condition Monitoring
    - Early Warning Failure
    - Preventitive Maintenance
    - Machine Health Tracking

VTPS Case: Cross Domain Challenges

ELECTRICAL & COMPUTER ENGINEERING

Internet-of-Things

- Connectivity
- Sensing
- Automation
- Scalable
- ML / AI Capapble
- Low Power

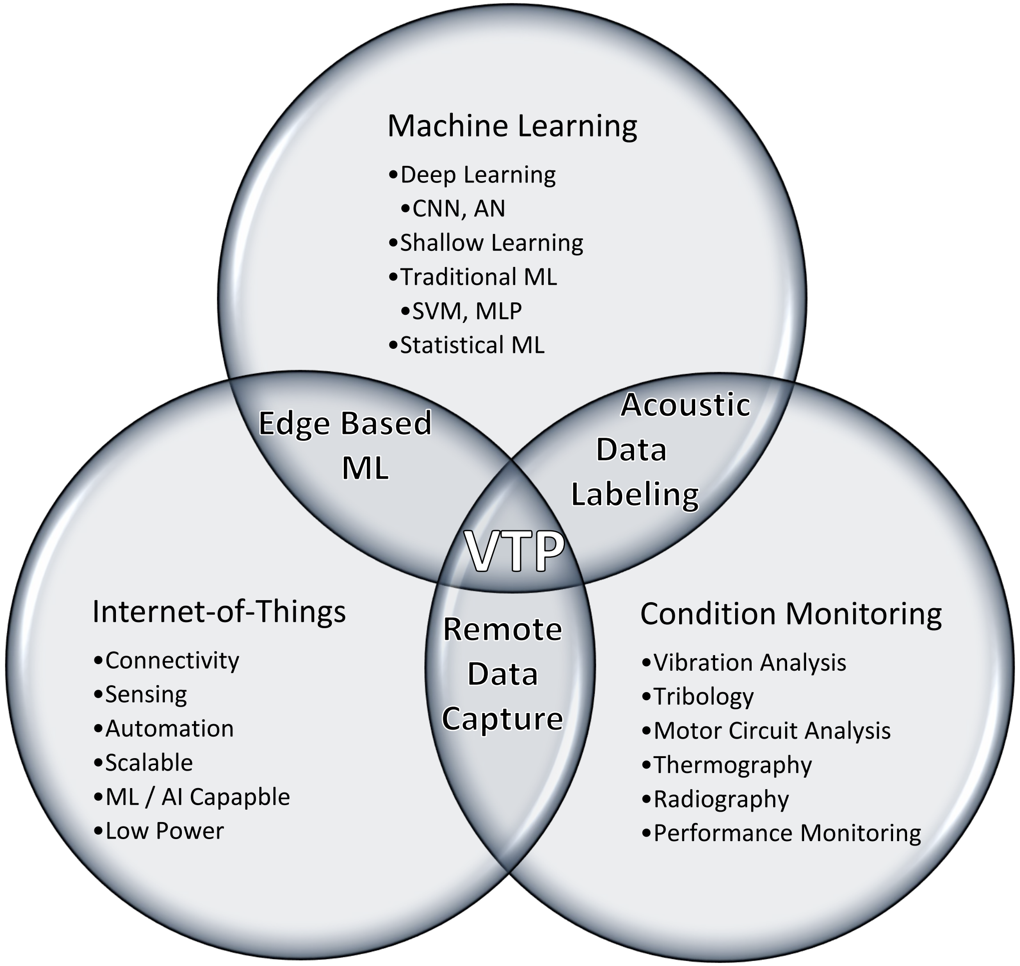
Machine Learning

- Deep Learning
  - CNN, AN
- Shallow Learning
- Traditional ML
  - SVM, MLP
- Statistical ML

Condition Monitoring

- Vibration Analysis
- Tribology
- Motor Circuit Analysis
- Thermography
- Radiography
- Performance Monitoring

VTPS Case: Cross Domain Challenges



Machine Learning + Condition Monitoring

- Human In The Loop Data Labeling, Audible Range (Psychoacoustic)

Condition Monitoring + Embedded Systems

- Remote Sense & Data Capture w/o Interfering With PLC Network

Embedded Systems + Machine Learning

- Train in the Cloud, Deploy on The Edge

Current Knowledge Base:

Many Examples Across 1-2 Domains.

Very Few Across All 3

ML

Citation	Machine-Learning Research	Machine Learning	Embedded Systems	Condition Monitoring
muller2020acoustic	Acoustic anomaly detection for machine sounds based on image transfe	☑		
wissbrock2022discussion	Discussion of Features for Acoustic Anomaly Detection under Industrial D	☑		☑
potovcnik2021condition	Condition classification of heating systems valves based on acoustic feat	☑		☑
stowell2015detection	Detection and classification of acoustic scenes and events	☑		☑
mesaros2017detection	Detection and classification of acoustic scenes and events: Outcome of t	☑		☑
dohi2022description	Description and discussion on DCASE 2022 challenge task 2: Unsupervise	☑		☑
koizumi2019toyadmos	ToyADMOS: A dataset of miniature-machine operating sounds for anoma	☑		☑
han2015deep	Deep compression: Compressing deep neural networks with pruning, trai	☑		
habib2021toward	Toward an automatic quality assessment of voice-based telemedicine cc	☑		
sculley2015hidden	Hidden technical debt in machine learning systems	☑		
shin2023robust	Robust and Lightweight Deep Learning Model for Industrial Fault Diagnos	☑		☑
	or infrared thermal image based machine health monitori	☑		
	ng-based methods for acoustic emission testing: a review	☑		☑
	nomalous sound detection using self-supervised classifica	☑		
	ning approach for locating acoustic emission	☑		☑
	aly detection in additive manufacturing with long short-te	☑		☑
	tificial neural networks to model the acoustic behaviour	☑		☑
	ches for anomaly detection from sound	☑		
	based approach for detecting anomalous audio events	☑	☑	
	nd detection based on interpolation deep neural network	☑		
	nd detection with machine learning: A systematic review	☑		☑

CM

Citation	Precision Agriculture | Condition Monitoring	Machine Learning	Embedded Systems	Condition Monitoring
scherer1993irrigation	Irrigation water pumps			☑
purohit2019mimii	MIMII Dataset: Sound dataset for malfunctioning industrial machine inve	☑		☑
mulimani2018acoustic	Acoustic event classification using spectrogram features	☑		☑
schiavello2004abnormal	Abnormal vertical pump suction recirculation problems due to pump-syst			☑
de2022hybrid	A hybrid prototype selection-based deep learning approach for anomaly	☑		☑
thoidis2021semi	Semi-supervised machine condition monitoring by learning deep discrimi	☑		☑
lei2020applications	Applications of machine learning to machine fault diagnosis: A review an	☑		☑
kane2020critical	Critical evaluation and comparison of psychoacoustics, acoustics and vib			☑
serin2020review	Review of tool condition monitoring in machining and opportunities for c			☑
poveda2020comparison	A comparison between psychoacoustic parameters and condition indicat			☑
oxenham2012pitch	Pitch perceptionPitch perceptionPitch perceptionPitch perception			☑
kudelina2021trends	Trends and challenges in intelligent condition monitoring of electrical ma			☑
mithen2012thirst	Thirst: Water and power in the ancient world			☑
tagawa2021acoustic	Acoustic anomaly detection of mechanical failures in noisy real-life facto	☑		☑
gong2018design	Design and implementation of acoustic sensing system for online early fa	☑		☑
wisser2008global	Global irrigation water demand: Variability and uncertainties arising from			☑
mohanty2014machinery	Machinery condition monitoring: Principles and practices			☑
volkovas2006acoustic	Acoustic emission used for detection of crack generation in propellers of			☑
alfayez2005application	The application of acoustic emission for detecting incipient cavitation an			☑
murovec2020psychoacoustic	Psychoacoustic approach for cavitation detection in centrifugal pumps			☑
tsuji2021machine	A machine sound monitoring for predictive maintenance focusing on very		☑	☑
stoney2012development	The development of surface acoustic wave sensors (SAWs) for process m			☑

IoT

Citation	Embedded Systems Research	Machine Learning	Embedded Systems	Condition Monitoring
hervas2016fpga	An FPGA Platform Proposal for real-time Acoustic Event Detection: Optir		☑	☑
zabinski2021fpga	**FPGA-Embedded Anomaly Detection System for Milling Process**	☑	☑	☑
seva2020low	A low energy FPGA platform for real-time event-based control	☑	☑	
taguchi2021fpga	**FPGA Implementation of Support Vector Machine Using Ising Model fc**	☑	☑	☑
thomas2019hierarchical	Hierarchical and distributed machine learning inference beyond the edge	☑	☑	
ren2021tinyol	Tinyol: Tinyml with online-learning on microcontrollers	☑	☑	
karges2022soundscapes	Soundscapes on edge-The real-time machine learning approach for meas		☑	
barksdale2018condition	Condition monitoring of electrical machines with Internet of Things		☑	☑
murshed2021machine	Machine learning at the network edge: A survey	☑	☑	
ray2021review	A review on TinyML: State-of-the-art and prospects	☑	☑	
coady2019remote	Remote acoustic analysis for tool condition monitoring		☑	☑
humphreys2014fpga	FPGA based monitoring platform for condition monitoring in cylindrical g		☑	

Emerging Concepts / Models

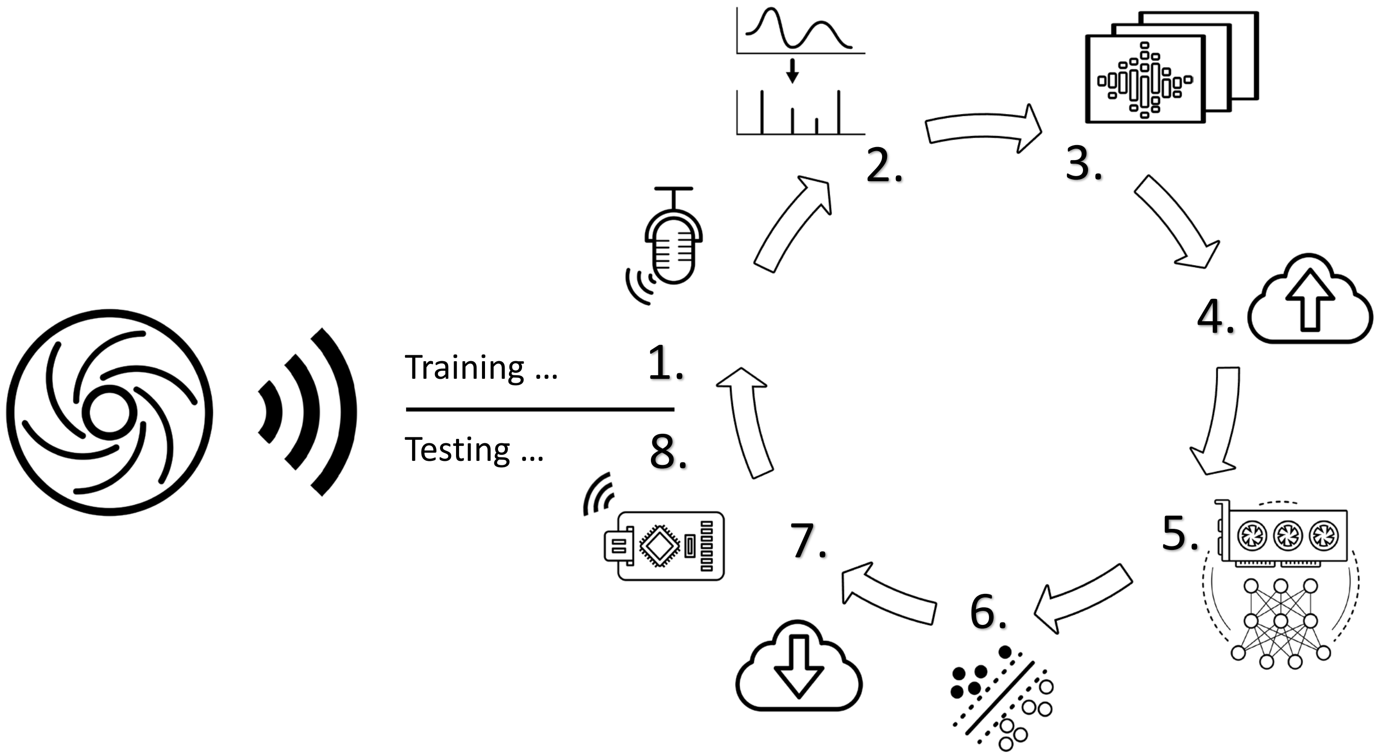
- Image Transfer Learning + Acoustic Anomaly Detection
  - [muller2020acoustic], [wissbrock2022discussion]
- Transfering Deep Learning Models to Low Power IoT Devices
  - [ren2021tinyol], [ray2021review]
- Human Recognizable Audio Frequency, Acoustic Condition Monitoring
  - [mulimani2018acoustic], [kane2020critical],[tagawa2021acoustic]

Problem Statement

Propose a Means of Remotely Monitoring Vertical Turbine Pump Stations That…

- Can Properly Recognize Predicted Running Conditions,
- Can Label Priorly Unknown Anomaly Conditions,
- Can Be Deployed Without Physically Modifying the Pumps,
- And Do So Reliably Despite Harsh Conditions And Limited Network Availability.

Proposed Approach:

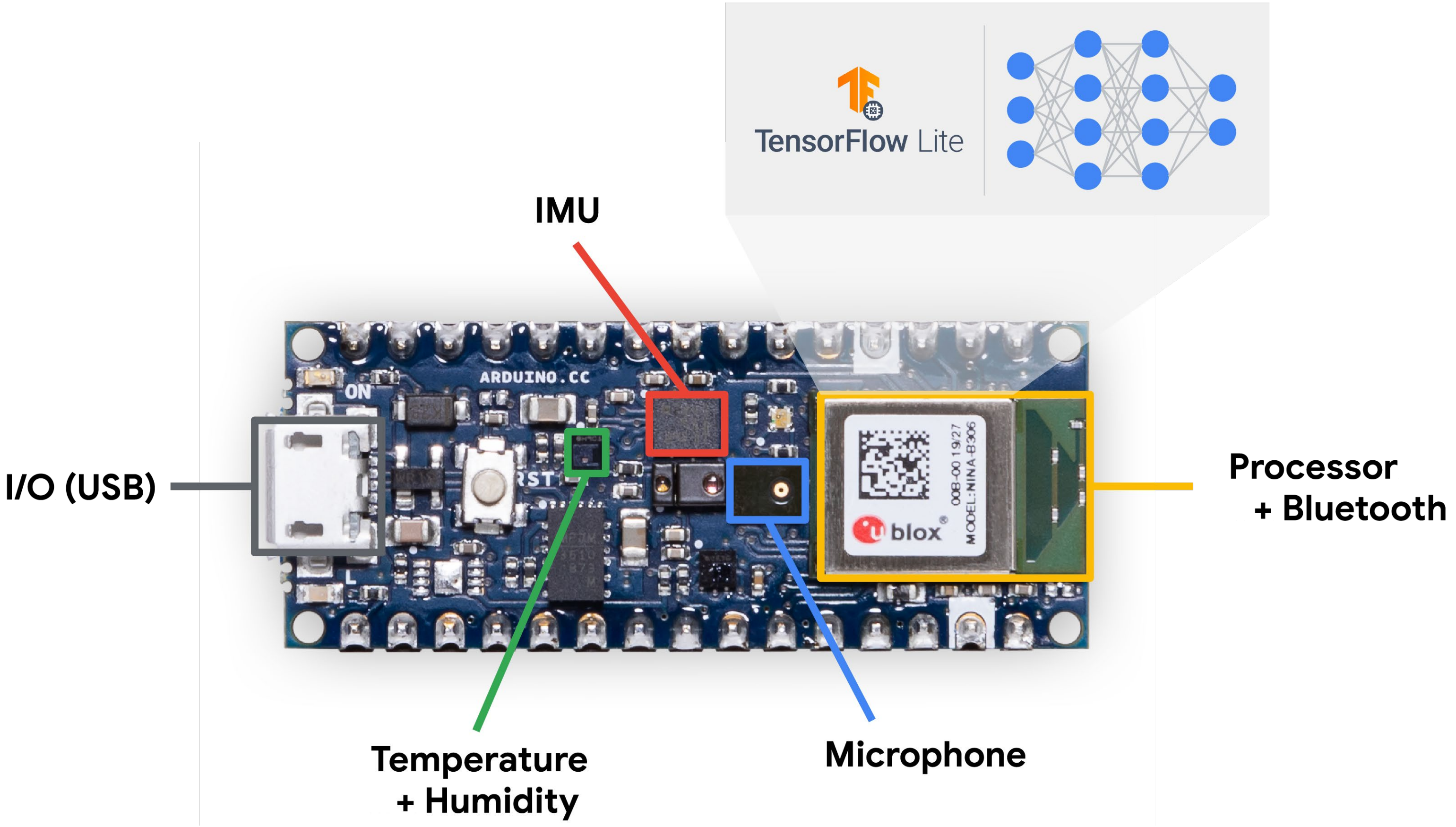


1. Initial Data Collection / Ingest
2. DSP / Time Domain -> Freq Domain
3. Freq Domain - > Image Representation
4. Upload to Deep Learning Infrastructure
5. Deep Learning Model / Analysis
6. Convert Model Inference to Firmware
7. Download Firmware to Edge, IoT Device
8. Test Embedded Neural Net on Live Device

Define Hardware Scope:

Targe Specification:

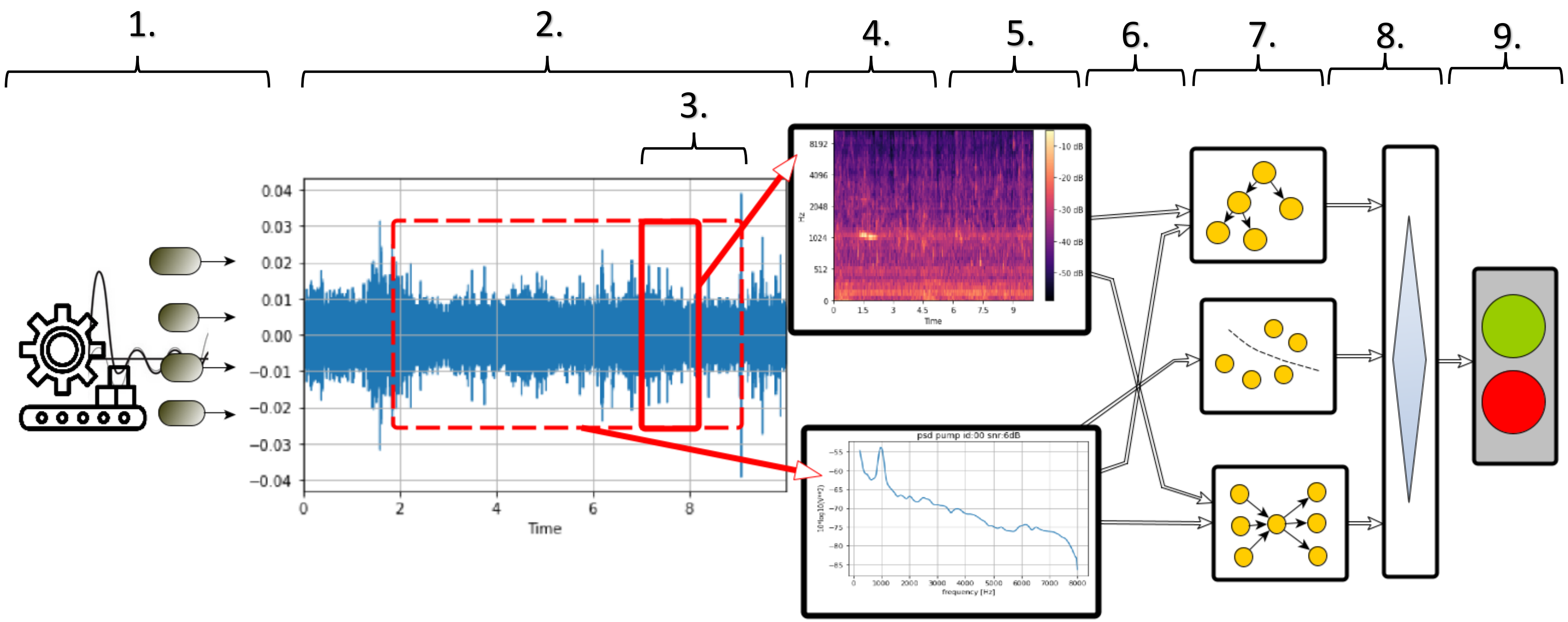
1. Capable of running TensorFlow Lite
2. Onboard Microphone
3. 32 bit (i.e. Arm Cortex M0+ … M4)
4. Networkable (i.e. BLE)
5. Min 1MB Flash



Arduino Nano BLE 33 Sense:

1. Runs TensorFlow Lite
2. MEMS Microphone
3. Arm Cortex M4F / 64 MHz + FPU
4. Bluetooth Low Energy
5. 1MB Flash / 256 KB Ram

End to End Data Pipeline

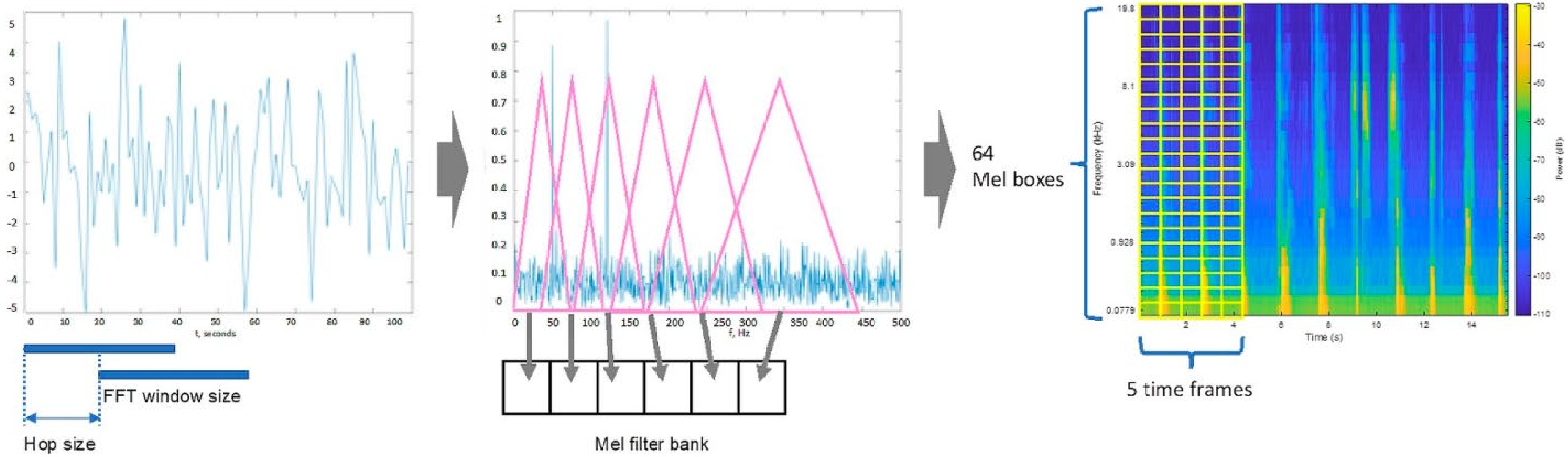


1. Record Audio In Fixed Duration .Wav Files (I.E. 5s-10s)
2. Apply Windowing To .Wav Files, Reduce Durations, Specify Overlap.
3. Split New Smaller .Wav Files Into Train / Test Groups
4. Convert Time Domain Waveforms To Frequency Domain, FFT / STFT
5. Convert Freq Domain To Image Representation (Mel, Log-mel, MFCC)
6. Run Image Based Neural Network Classification On “Audio” Images Using Training Data
7. Test & Retrain NN Model With Testing Data
8. Once NN Model Is Trained, Optimize For Tensorflow Lite & Export As Firmware
9. Install Firmware On MCU And Test Live

Data Ingest: FFT, Filter Banks, MFCC

Tune DSP Looking for best combination of…

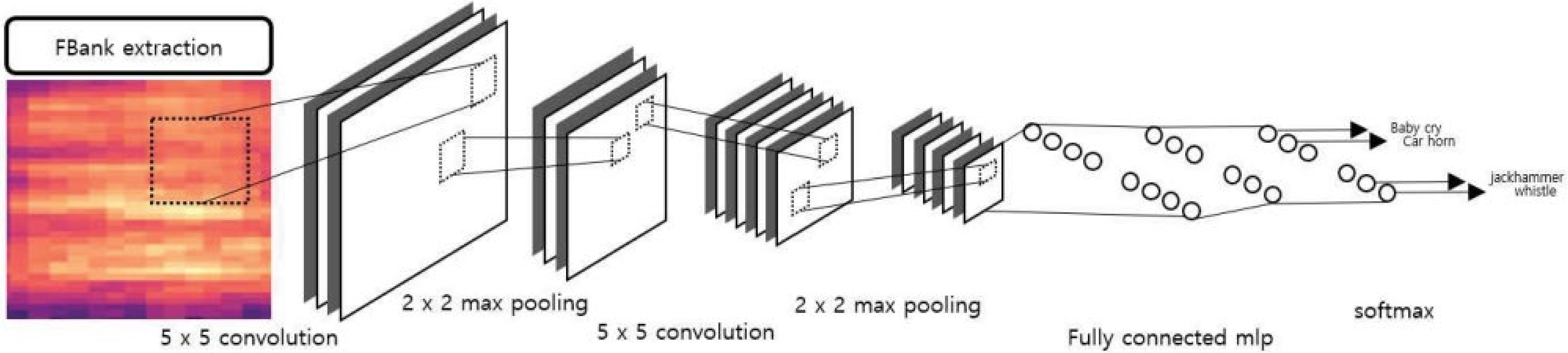
- Clear visual distinctions between class samples
- Recording windows small enough for IoT device performance
- Recorindg windows large enough to capture class features
- Log based frequency spread to match human perception (human perception = labels)



Feature Extraction: CNN

Tune NN Classifier for…

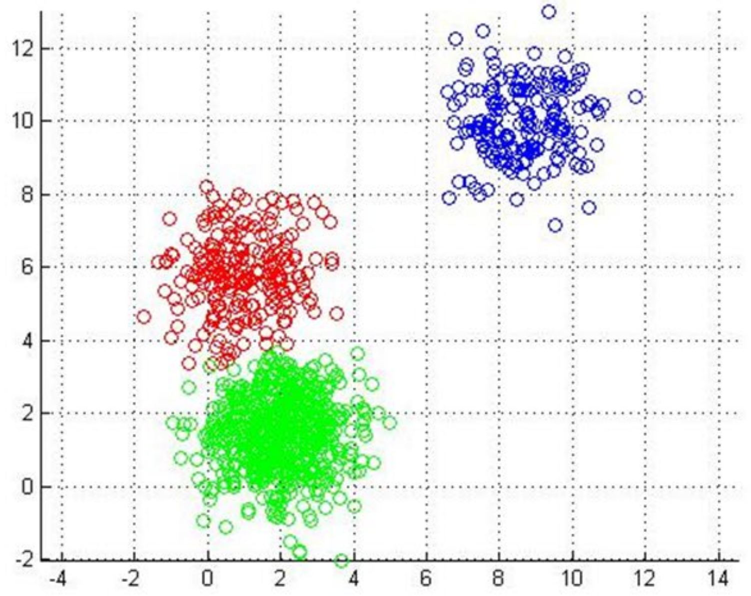
- Good distinction between labeled classes,
- Minimal processing overhead,
- Generate as few feature parameters as possible, will impact anomaly detection.

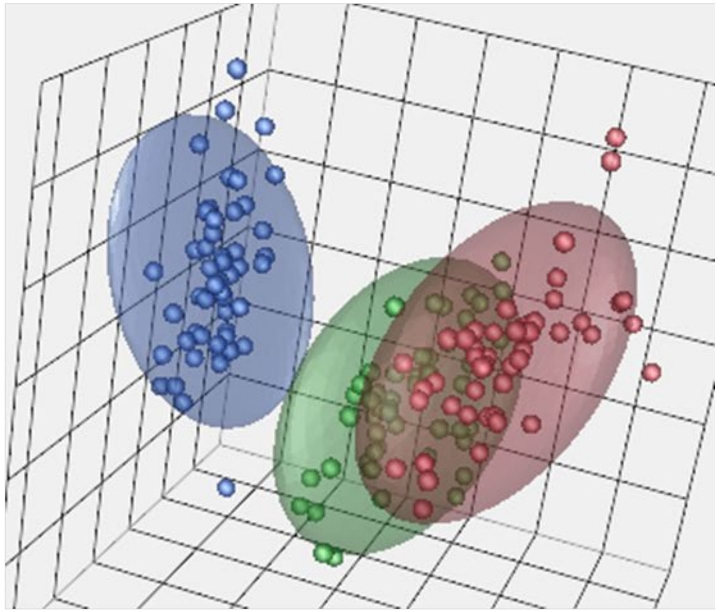


Anomaly Detection: K-Means

Anomaly Detection is …

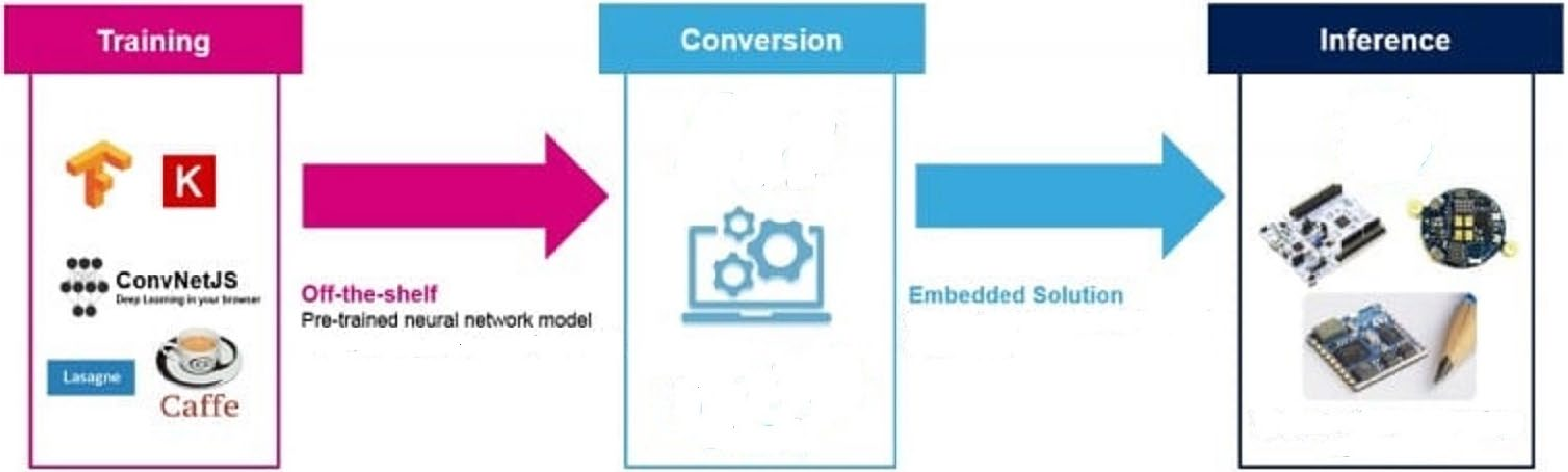
- Essentially a One Class Classifier,
- Very Sensitive to High Dimensionality (keep feature generation low)
- Difficult due to extremely little training data, lack of "known unknowns"
- Try: K-NN, LOF, K-Means, SVM. Lots of trial and error.





Model Inference (Transfer to MCU):

- Heavy lifting done in cloud via DNN / TensorFlow
- Finished Model Optimized to run on TensorFlow Lite
- Flash Finished Model to Device, if possible.
- If not enough resources, reduce model size, i.e. smaller recording windows or lower bit rate. Retrain and attempt to flash again.



Live Demo, Pre-Deployment:

ELECTRICAL & COMPUTER ENGINEERING

Model Simulation & Selection

	Initial Ingest (Raw Data Set)		
	Analog Freq	Samp Rate	Duration
Attempt 1	20Hz - 20 KHz	48KHz / 16bit	10 sec
Attempt 2	20Hz - 20 KHz	48KHz / 16bit	10 sec
Attempt 3	20Hz - 20 KHz	48KHz / 16bit	10 sec

Preprocessing Time Domain (waveform)		
Freq	Window (length)	Window (slide)
16 KHz	2000 ms	1000 ms
44 KHz	2000 ms	2000 ms
16 KHz	1000 ms	500 ms

	Preprocessing Freq Domain (FFT / MEL)						
	Frame Length	Frame Stride	Filter Number	FFT length	Low Band	High Band	Features Generated
Attempt 1	0.05	0.05	32	256	300 Hz	0 Hz	16,000
Attempt 2	0.075	0.075	NA*	128	NA*	NA*	88,200
Attempt 3	0.05	0.05	32	256	300 Hz	0 Hz	16,000

	As Deployed Embedded Model			
	Analog Freq	Samp Rate	Duration	FFT Slide
Attempt 1	20Hz - 20 KHz	16 KHz / 16bit	2000 ms	1000 ms
Attempt 2	20Hz - 20 KHz	44 KHz / 16bit	2000 ms	1000 ms
Attempt 3	20Hz - 20 KHz	16 KHz / 16bit	1000 ms	500 ms

Deploy to IoT
Pass / Fail
Fail
Fail
Pass

Observations:

- Keep Window Size To 1 Sec @16 Khz Or 2 Sec @ 8 Khz
- Spectrogram Created 5-6 Times More Features During Auto Encoding Than MFE (Mel-scale Frequency Extraction)
- Both MFE And Spectrogram Created Too Many Feature Parameters For Use In Anomaly Detection.
- Try Again With MFCC?
- Consider 2 Iot Devices: (1) For Status Classification (Idle, Speed1, Speed2, Etc) & (1) For Only Anomaly Detection

Live Demo, Model Architecture:

Model Simulation & Selection

ELECTRICAL & COMPUTER ENGINEERING

Model Architecture

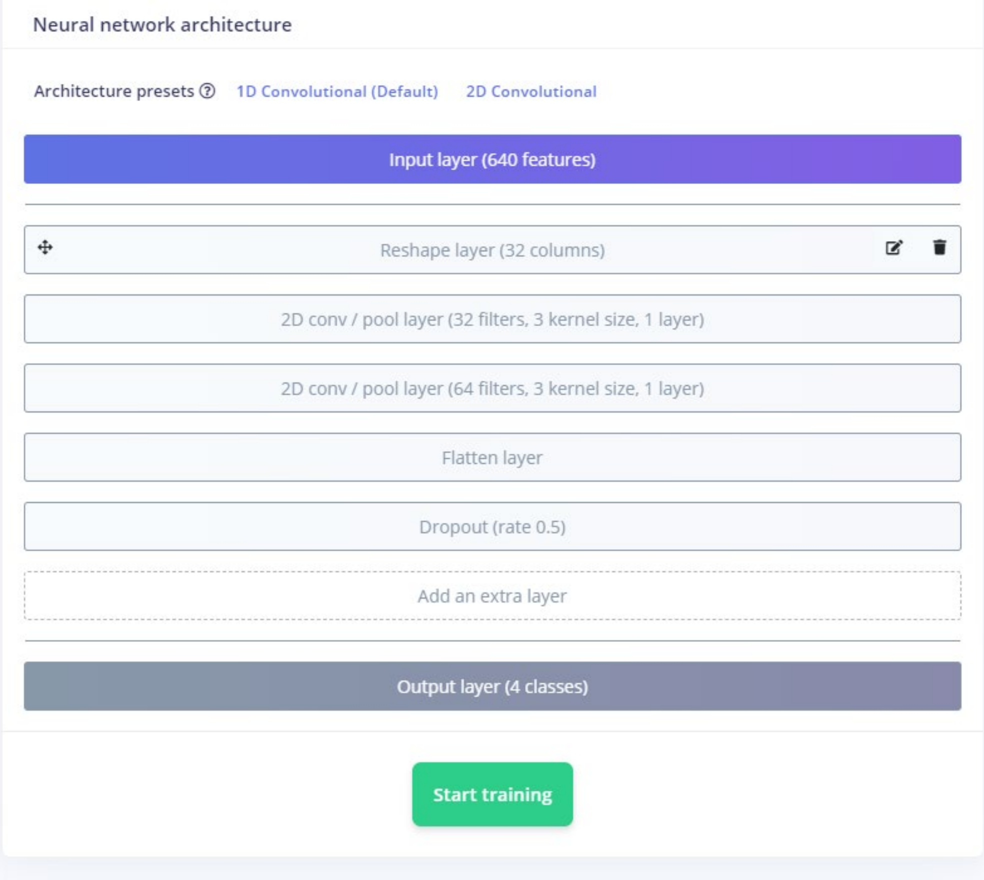
```
# model architecture
model = Sequential()
channels = 1
columns = 32
rows = int(input_length / (columns * channels))
model.add(Reshape((rows, columns, channels), input_shape=(input_length, )))
model.add(Conv2D(32, kernel_size=3, kernel_constraint=tf.keras.constraints.MaxNorm
model.add(MaxPooling2D(pool_size=2, strides=2, padding='same'))
model.add(Conv2D(64, kernel_size=3, kernel_constraint=tf.keras.constraints.MaxNorm
model.add(MaxPooling2D(pool_size=2, strides=2, padding='same'))
model.add(Flatten())
model.add(Dropout(0.5))
model.add(Dense(classes, name='y_pred', activation='softmax'))
```

Learning Rate

```
# this controls the learning rate
opt = Adam(learning_rate=LEARNING_RATE, beta_1=0.9, beta_2=0.999)
callbacks.append(BatchLoggerCallback(BATCH_SIZE, train_sample_count, epochs=EPOCHS)
```

Training Neural Network

```
# train the neural network
model.compile(loss='categorical_crossentropy', optimizer=opt, metrics=['accuracy']
model.fit(train_dataset, epochs=EPOCHS, validation_data=validation_dataset, verbose
```



Live Demo, Predicted Performance:

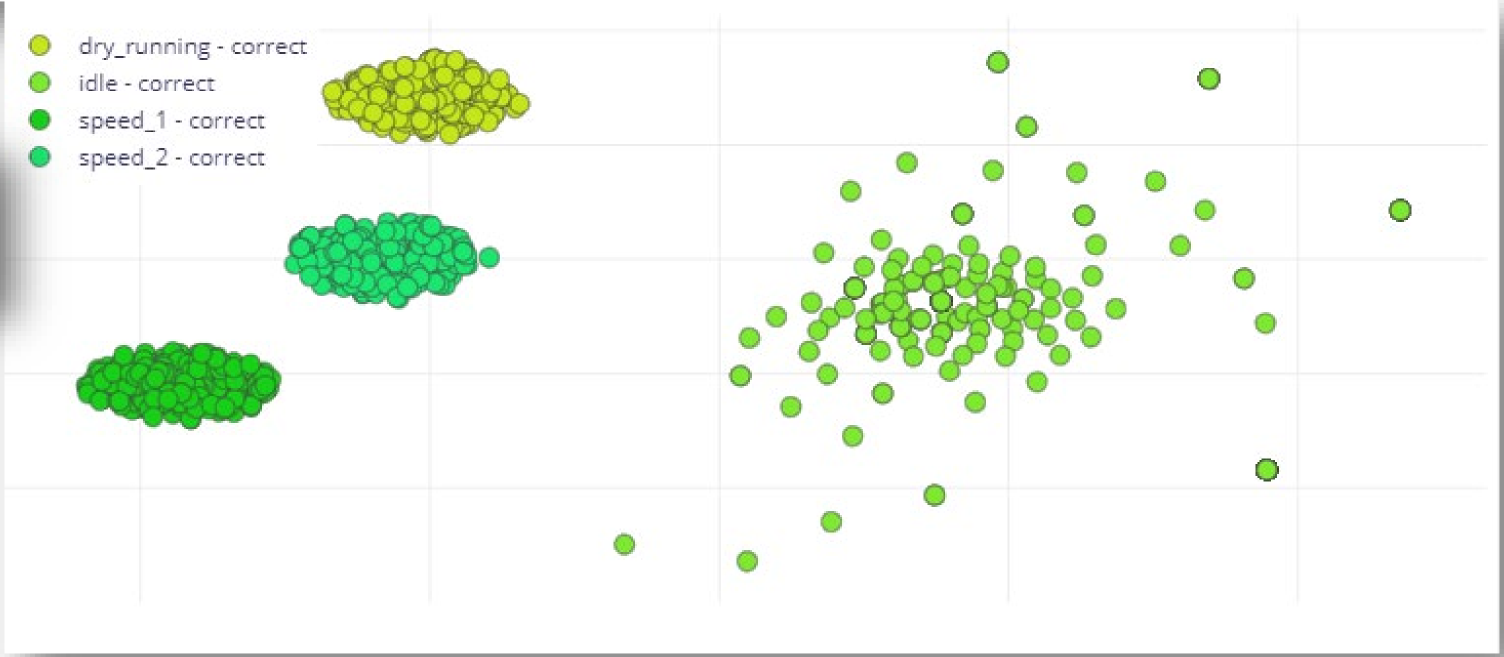
ELECTRICAL & COMPUTER ENGINEERING

	DRY_RUNNING	IDLE	SPEED_1	SPEED_2
DRY_RUNNING	100%	0%	0%	0%
IDLE	0%	100%	0%	0%
SPEED_1	0%	0%	100%	0%
SPEED_2	0%	0%	0%	100%
F1 SCORE	1.00	1.00	1.00	1.00

Cloud Prediction (TensorFlow)

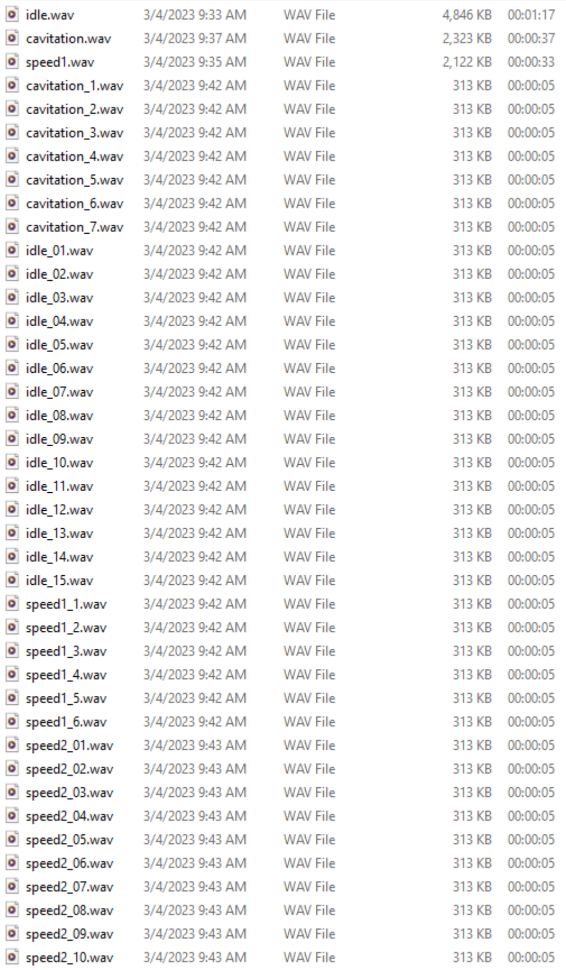
On-device performance

INFERENCING TIME	PEAK RAM USAGE	FLASH USAGE
721 ms.	54.6K	65.6K

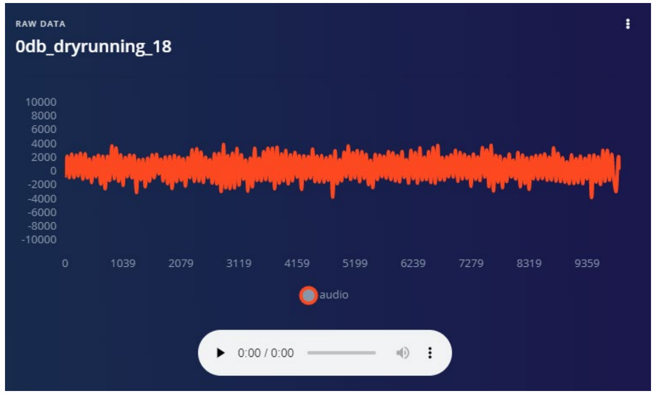


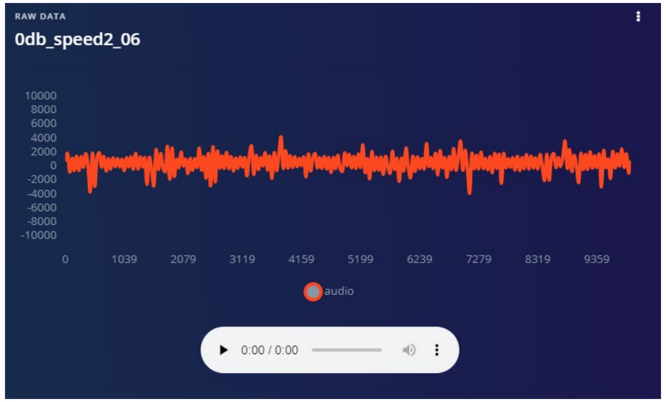
Live Demo:

(Used Different Recording Devices For Training Data Set Vs. Live Deployment)



idle.wav	3/4/2023 9:33 AM	WAV File	4,846 KB	00:01:17
cavitation.wav	3/4/2023 9:37 AM	WAV File	2,323 KB	00:00:37
speed1.wav	3/4/2023 9:35 AM	WAV File	2,122 KB	00:00:33
cavitation_1.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
cavitation_2.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
cavitation_3.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
cavitation_4.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
cavitation_5.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
cavitation_6.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
cavitation_7.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_01.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_02.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_03.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_04.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_05.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_06.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_07.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_08.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_09.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_10.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_11.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_12.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_13.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_14.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
idle_15.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
speed1_1.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
speed1_2.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
speed1_3.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
speed1_4.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
speed1_5.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
speed1_6.wav	3/4/2023 9:42 AM	WAV File	313 KB	00:00:05
speed2_01.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_02.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_03.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_04.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_05.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_06.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_07.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_08.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_09.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05
speed2_10.wav	3/4/2023 9:43 AM	WAV File	313 KB	00:00:05



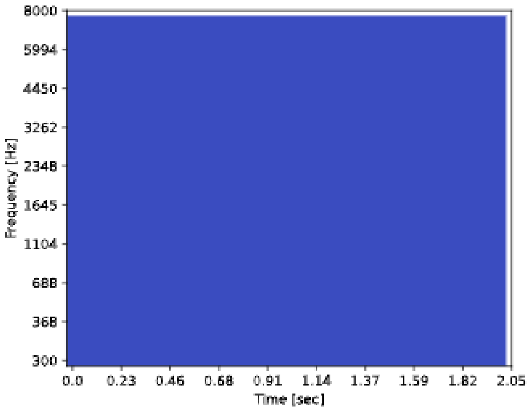


ELECTRICAL & COMPUTER ENGINEERING

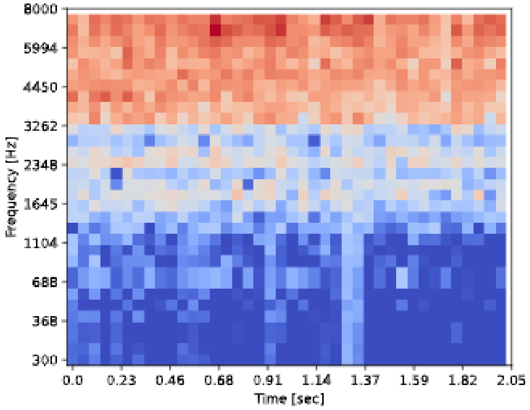
Live Demo:

MFE Visualization Results: 1000 ms Window, 16 KHz , 500 ms Slide, 32 Frequency Bands

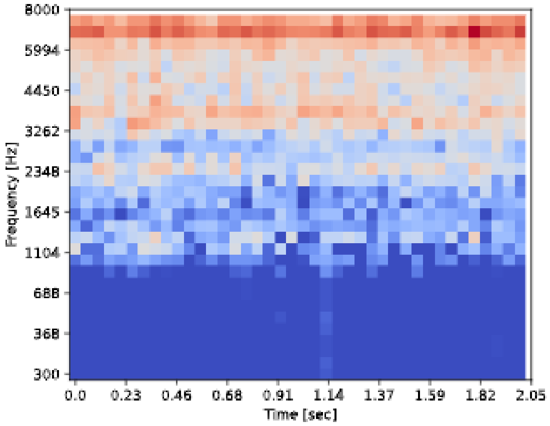
Idle



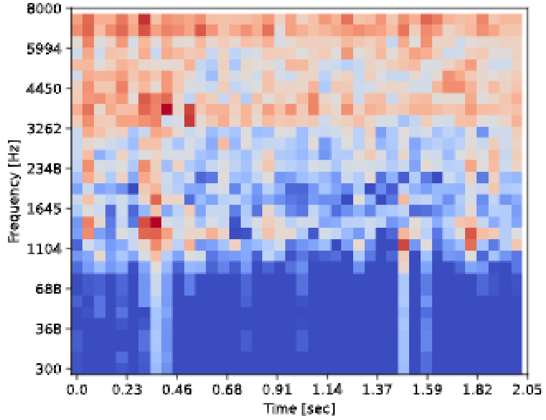
Speed 1



Speed 2



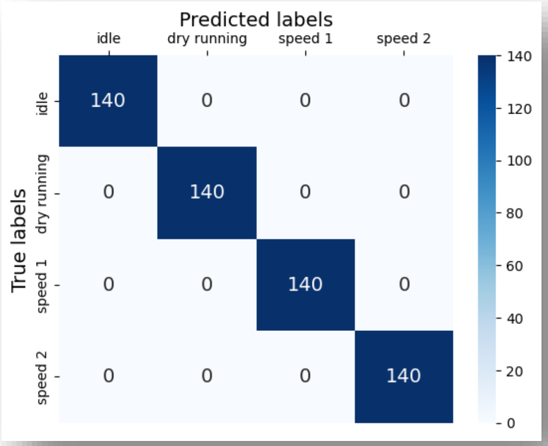
Dry Running



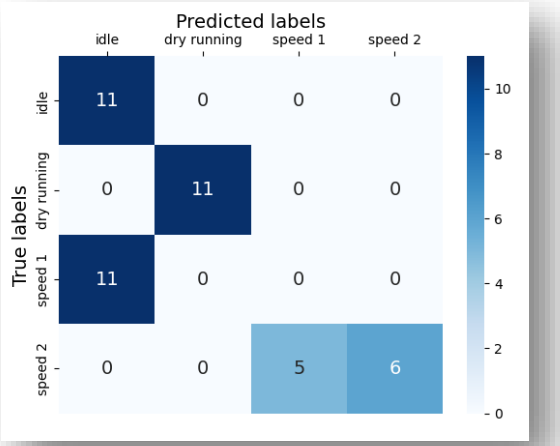
Live Demo:

ELECTRICAL & COMPUTER ENGINEERING

Simulation Performance



Production Performance



Raw Audio Features	Tuning Hyper-Parameters	Processed Image Features
16,000	5	870

<u>Final MFE Tuning Parameters & Values:</u>

1. MFE Frame Length = 0.1
2. MFE Frame Stride = 0.066
3. Number of Filterbanks = 30
4. FFT Length = 256
5. Frequency Cutoff = 1250 Hz low cutoff

<u>Deployed Results (Sample)</u>

- Classification Only, No Anomaly Detection
- Idle And Nominal Running (Speed1 + Speed2) Had Good Recognition
- Poor Distinction Between Speed_1 & Speed_2
- Dry Running Gave Was The Most Easy To Properly Classify.
- Additional 'Cavitation' Class Did Not Produce Expected Uncertainty In Other Classes, Strongly Misclassed As Dry Running.

Discussion:

- Development workflow overall was successful…
- Classification worked. Best online, less accurate as deployed but worked.
- Still working out Anomaly Detection. MFE Spectrograms to NN create too many features. Possibly try Spectral Analysis instead of Spectrogram.
- Use of different model microphones for training data vs live deploy did not have noticeable impact.
- Overall, project was majority success but with much room to continue development.

Conclusion / Remarks / References