Introduction to SensiML

• AI toolkit for building intelligent IoT sensor endpoints
  • Little to no data science requirement
  • Automation of algorithm firmware coding
  • Working code in days/weeks not months
  • Built to support OEM IoT product teams
  • Full workflow from data collection through testing

• Started in 2012 as an Intel® SW development tool

• Acquired by QuickLogic® in January
  • Full HW/SW solution offering with QuickAI HDK
  • Standalone SW tool for prevalent 3rd party HW

AI Endpoint IoT Sensor Algorithms For...

Initially built to support Intel® Curie™ & Quark™ MCUs and Atom™ MPUs

ARM Cortex-M core architecture supported

Raspberry Pi 3 support added

QuickLogic QuickAI multi-core edge AI HDK optimized

Ongoing Integrations for Additional HW
Validating a TinyML Use Case

1. Is data privacy a concern?

2. Is this project solvable when unlimited resources are available?

3. Is this problem simple enough to solve with less sophisticated techniques, i.e. is it solvable with a thresholding?

4. Does it make sense to run at the edge, or is the better solution to stream data back to a central server?

5. Are there resource limits that prevent using a larger local processor capable where current tools could be run to meet the need?

6. Does machine learning make this solution scalable?
Applications for TinyML

- Industrial Wearables
- Voice Command & Control
- Fleet Maintenance
- Predictive Maintenance
- Factory Process Automation
- Pet Wearables
- Sports/Fitness Wearables
- Smart Appliances
- Livestock Monitoring
- Smart City Infrastructure
- Agricultural Smart Sensing
- Retail Traffic Analysis
What makes predictive valuable

• Maximize uptime
• Better schedule maintenance
• Identify production problems
• Catch/Prevent serious or dangerous issues

Brief History of Predictive Maintenance and Vibration Analysis

1950s
First Portable Vibration Analysis Machine

1980s
Handheld Vibration Analysis Machines

1990s
Personal Computer for off-line analysis and tape for data acquisition

2000s
Broader Commercialization of predictive maintenance for critical machines

2010s
Cloud analytics and connected sensors

http://environmentaltestanddesign.com/past-present-future-vibration-analysis/
What will it take to get to the next stage of the smart factory

1. Low cost, low power, high fidelity Sensors easy to integrate

2. Low cost, Low power, networks that easy to deploy in industrial environments.

3. Low cost, low-power, microcontroller capable of running necessary algorithms

Are we there yet?
Low Cost, High Fidelity Vibration Sensors

Traditional Industrial Analog sensor
- 20khz
- Can cost $1,000 including power source and signal conditioning
- More difficult to deploy (noise associated with analog signal)

Latest MEMS sensors
- 24Khz sampling for 30$.
- 5k-10k sampling for 10$
- Can be Integrated directly and powered with the same power supply as the SoC.
- Easier to calibrate, easier to isolate from electronic noise or other interference.

MEMS Sensors are a game changer for predictive maintenance.

Currently good enough, but still getting better in terms of fidelity, power draw and price.
Low Cost, Low Power Easy to deploy in Industrial Setting

Long range networks that are easy to deploy
• 1 km for urban areas, 30km in wide open areas.
• Data upload and download
• Battery life for years depending on usage

Downside is low Data Rate - 100bps.

https://radiobridge.com/blog/choosing-the-right-wireless-standard-for-your-iot-sensors/

Makes a great use case for TinyML
Low Cost, Low Power High Performance Microcontroller

PC Based Condition Monitoring - Industry standard minimum 20kHz real-time speed

Pentium P54C 166Mhz 1994
- FFT 1024 166 per second (1)
- Cost $1,500
- 100’s Watts

M4 processor
- FFT 1024 800 per second on (2)
- Cost less that <$3
- mW power draw (using DSP)

Current M class Processors are just as capable as the computers that revolutionized the Industry in the 80’s and 90’s

2. https://community.arm.com/cfs-file/__key/communityserver-blogs-components-weblogfiles/00-00-00-21-42/7563.ARM-white-paper-_2D00_DSP-capabilities-of-Cortex_2D00_M4-and-Cortex_2D00_M7.pdf
What will it take to get to the next stage of the smart factory

1. Low cost, low power, high fidelity Sensors easy to integrate

2. Low cost, Low power, networks that easy to deploy in industrial environments.

3. Low cost, low-power, microcontroller capable of running necessary algorithms

What about actually developing the algorithms?
Hardware is there, what about Software?

**Traditional Coding Approach**
- Devise Application Hypothesis
- Construct Prototype
- Collect Training Data
- Analyze Patterns
- Construct Idealized Algorithm
- Simplify for Embedded
- Code and Optimize
- Test and Iterate as Required

**AutoML Based Approach**
- Devise Application Hypothesis
- Construct Prototype
- Collect Well-Labeled Training Data
- Auto-Generate Algorithm Embedded Code
- Test and Iterate as Required

Labeled Dataset Embodies Application Knowledge, Paramount to the Process
Building a Smart Sensing Device with SensiML

1. Import or Collect Sensor Data Preferred
   Smart Device Processor

2. Auto-Generate
   Smart Sensor Algorithm

   SensiML Knowledge Pack
   (Optimized Algorithm Code)

3. Flash Result to
   Your Device and Test

   Your Intelligent IoT
   Sensing Endpoint
Building a Smart Sensing Device with SensiML

1. Import or Collect Sensor Data Preferred
   Smart Device Processor

2. Auto-Generate
   Smart Sensor Algorithm

3. Flash Result to
   Your Device and Test

SensiML Knowledge Pack
(Optimized Algorithm Code)

Your Intelligent IoT
Sensing Endpoint

All product and company names, logos, and images are trademarks of their respective holders.
Data Collection and Labeling

• Prediction algorithms attempt to classify useful insight from varying input data
  • Machine condition and fault causes from vibration and audio data
  • Gait analysis for performance and injury prevention from motion sensor data
  • Security alarm states from audio, vibration and/or visual image sensor data

• Variance sources include:
  1) **Signal**: Useful sensor measured differences correlated to intended insight
  2) **Metadata**: Knowable contextual differences that can be useful filters
  3) **Noise**: Unknown differences not correlated to intended insight

• Data collection and labeling objective
  1) **Signal**: Capture/label examples for each desired insight state across the distribution of (2) and (3)
  2) **Metadata**: Seek to convert as much noise variance into measurable metadata as possible
  3) **Noise**: Seek to suppress; more unexplained variance = more data required for a good algorithm
SensiML Data Capture Lab: Data Collection Made Simpler

**Integrated Data Acquisition**
Collect raw data directly from device without custom scripts or conversion

**Time-Series Data Strip Charts**
Mark segments of interest; auto-segment others based on learned criteria

**Project Explorer View**
Lists all available data collected and its status

**Event and Metadata Labeling**
Label segment classifications and relevant contextual metadata

**Video Annotation Window**
Time-synched visual cues about collected sensor data waveforms
Conditional Monitoring Event Types

Periodic events: continuous events requiring constant monitoring to look for specific faults.

Gesture events: Specific gestures that require segmentation for start and end then classification.

Adaptive Gestures: Similar, but not necessarily repetitive gestures. Can often involve active searching or other control mechanisms.
Signal, Metadata, and Noise: An Example

Motor Fault Detection and Classification

• **Signal** (*measurable variance driving intended insight classification*)
  - Bearing faults: From vibration and audio sensors
  - Lubrication issues: From audio, vibration, and thermal sensors
  - Loose mount: From vibration sensors

• **Metadata** (*knowable contextual differences*)
  - Motor frame size: Changes vibration and audio signatures
  - Mount type/foundation: Seismic mount, rubber isolators, direct to equipment, etc.
  - Load and shaft coupling: Different loads as used, coupler type, measured runout / alignment

• **Noise** (*unknown variance not correlated to intended insight*)
  - Motor driven load variance and hysteresis
  - Unmeasured on-site installation variance
  - Unit-to-unit motor manufacturing tolerances
Building a Smart Sensing Device with SensiML

1. Import or Collect Sensor Data Preferred
   Smart Device Processor

2. Auto-Generate
   Smart Sensor Algorithm
   SensiML Knowledge Pack
   (Optimized Algorithm Code)

3. Flash Result to
   Your Device and Test
   Your Intelligent IoT
   Sensing Endpoint

All product and company names, logos, and images are trademarks of their respective holders.
Model Building: Generating an Endpoint Algorithm

Generalized AI Processing Pipeline

1. **Raw Sensor Data Input**
   - Signal Pre-Processing:
     - Filtering
     - Downsampling
     - Windowing
     - Normalization
   - Feature Transformation:
     - Amplitude
     - Zero-Crossing Rate
     - Histogram
     - FFT / MFCC
   - Classification:
     - DTW
     - KNN
     - SVM
     - ANN
     - Ensembles
   - Insight

Common Temptation Not Well-Suited for Endpoint AI

- Skip pre-processing and feature engineering
- Throw raw data at deep learning framework
- Requires large datasets to converge
- Requires lots of expertise
- Complex NNs with large neuron counts
- Not optimized for resource constrained endpoints
Model Building with SensiML

Algorithm Pipeline Optimized in Cloud Analytics Engine for Ideal Endpoint Performance

- **Raw Sensor Data Input**
- **Signal Pre-Processing**
  - Increase signal to noise ratio (SNR)
  - Segment event data of interest
  - Filter and remove aliasing effects
- **Feature Transformation**
  - Maximize desired class separability
  - Library of 80+ feature transforms
  - Automatic feature selection
- **Classification**
  - Many supported algorithms
  - Appropriate for endpoint MCUs
  - Optimization selects best choice for given dataset

Endpoint Optimized Insight

Algorithm Development
Case Study: Machine Trainer with SensorTile

1. Consult with domain experts on what faults are expected.

2. Determine what sensors to use and what location the sensors should be placed.

3. Collect initial data and validate sensors are working correctly, results are repeatable.
Sensor Analysis – Verify Sensor Data is Accurate

Sensor 1
Offset bias
Signal P2P – 7 bits
Noise – 4-5 bits

Sensor 2
Sensor 3
Ground
Signal P2P – 4-5 bits
Noise ~ 4-5 bits
Signal P2P – 7 bits
Noise – 4-5 bits
Case Study: Machine Trainer with SensorTile

1. Consult with domain experts on what faults are expected.

2. Determine what sensors to use and what location the sensors should be placed.

3. Collect initial data to validate sensors are working correctly, results are repeatable.

4. Develop a data collection strategy and routine. Understand how much data is needed (depends on variance, complexity of tasks etc.)
Label Data and Choose Segmentation

Build a Query
AutoML Search with Cross Session Validation

```python
# Main Settings
Pipeline Name: Demo
Settings:
- Query: All Classes
- Segmenter: Windowing
- Window Size: 1024

# Pipeline Settings

# Model Building

dsk.pipeline.set_input_query("All Classes")

dsk.pipeline.add_transform(
    "Windowing", params={"window_size": 1024, "delta": 1024, "train_delta": 0}
)

dsk.pipeline.add_feature_generator(
    ["name": "25th Percentile", "params": {"columns": ["GyroscopeZ"]}],
    ["name": "Minimum", "params": {"columns": ["GyroscopeZ"]}],
    ["name": "75th Percentile", "params": {"columns": ["AccelerometerZ"]}],
    ["name": "Global Peak to Peak of Low Frequency", "params": {
        "smoothing_factor": 4, "columns": ["AccelerometryY"]}
    ],
)

dsk.pipeline.add_feature_selector(
    ["name": "Custom Feature Selection By Index", "params": {
        "custom_feature_selection": [0, 1, 3, 4]
    }]
)
SensiML Analytics Studio: AutoGenerated Firmware

Classifier Output Mapping
Simple ordinal class map reduces volumes of raw sensor data to single application specific event byte

Hardware Target Selection
• Choose platform from simple dropdown
• Growing list of supported HW platforms
• Bare metal or RTOS variant

Embedded Algorithm Format*
Either binary or source C file library for extension and customization

Codeless Firmware Options
• Options for streaming classifier output over standard comms protocols
• Great for rapid device testing without additional coding required

Flash to Device Within SensiML UI
• No need to invoke separate vendor flash utility*
SensiML TestApp: Endpoint Device Testing

Current Classification

Class Map Legend

Feature Space Visualization
Select 2D view across any of model’s overall n-dimensional feature space

Realtime and Logged Output
Model (for concurrent classifiers), class label, and feature vector

Windows and Android Mobile Application
Working Demo

https://www.youtube.com/watch?v=7z6emHRQSu8&t=7s
For More Information...

Visit https://sensiml.com

• Video examples and tutorials
• Whitepapers on endpoint AI, IoT smart sensing
• Example applications
• SensiML tool suite documentation (see Support... Knowledge Base)
• Data Depot: repository of curated Time-Series Datasets
  https://datadepot.sensiml.com/
Thank You!

Questions?

contact: chris.knorowski@sensiml.com
Structure of a Knowledge Pack Pipeline

1. **Sensor Transforms**
   - Modifies data or combines data as it is streamed into the knowledge pack.

2. **Segmentation**
   - Determines the start and end of a segment of data for processing.

3. **Segment Transforms**
   - Transforms which operate on a single segment of data.

4. **Feature Extraction**
   - Extract Features from the segment of data.

5. **Feature Transforms**
   - Transforms that can be applied to the feature vector prior to classification.

6. **Classifier**
   - A trained edge optimized model.
Hierarchical Models

Decision Tree of Strong Classifiers:

1. Divide and conquer approach allows optimized resource usage.
2. Lazy evaluation of features extractors
3. Smaller Individual model sizes
4. Allows scaling the resolution depending between different classes
5. Provides higher accuracy than simpler flat models.
Signal to Noise: Optimal Sample Rate Analysis

- Perform a search over resolution and signal frequency by checking the model accuracy at different resolutions and frequency for all input channels.

- Identified the value of the optimal resolution

- Identified the value of the optimal frequency beyond which a further increase in frequency didn’t improve the model accuracy
Cloud-Based AI for IIoT Sensor Applications

Issues With Cloud-Based IoT Analytics:
- Can swamp networks with raw data traffic
- Latency is too great for many time critical applications
- No insight if network fails or throughput sufficiently impaired
- Security and data privacy require end-to-end coverage

Can Be Acceptable When:
- Analysis needs are not real-time
- Sensors are simple state devices or slow varying signals
- Network bandwidth is plentiful
Endpoint AI for IIoT Sensor Applications

Benefits With Cloud-Based IoT Analytics:
✓ Endpoints send only useful insights, not voluminous raw data
✓ Vastly reduced traffic allows use of cellular and IoT networks
✓ Zero network latency with local AI insight at endpoint
✓ No network dependence with local AI insight at endpoint
✓ Partitioned security/data privacy with local pre-processing

Vital When:
• Insight and/or decisions are needed in real-time
• Physical source sensors are high bandwidth dynamic signals
• Network bandwidth is limited