Tiny ML for Tiny Sensors:
Waking smarter for less

Abbas Ataya
March 29, 2022
Abstract:

Machine Learning at the Edge – where does the buzz stop and the value begin? If the edge is where the internet stops and the real world begins, sensors define the edge. By application, sensors convert physical phenomena into digital signals: data. What happens next is where ML is driving innovation.

Edge implies the connectivity of a node to a bigger system. The node has a sensor chip, created in a process optimized for cost, size, and power. Transmitting raw sensor data is resource-intensive. Converting the data into information or knowledge before transmission can greatly reduce the overall burden on the system. Machine learning is revolutionizing conversion. Nodes also have a radio chip for connectivity, similarly optimized, though radio chips are made in more advanced process nodes. As such, the radio chip will always have more resources than the sensor chip. The HW/SW node design must balance the conversion giving each chip a clearly defined role to improve accuracy and power conservation in the application.

I will explore how machine learning creates new opportunities to partition far reaching sensor systems and greatly reduce required resources such as dollars, volume, and watts. Using examples of systemic optimization studied in our lab, I will show how phasing the wake up of the nodes in the hierarchy of the system reduces resources while improving response. The analogies we discuss are similar to a person waking up in the night in response to a noise, a smell, a light, or a vibration.
Sensor & TinyML
We’re into sensors

• Sensors are the edge – physical converted into digital
  ➡️ Actuators convert digital into physical

• What is the job of the sensor?
  ➡️ The extremities of this central nervous system that is the internet
  ➡️ The sentry, at the edge of the system, trigger the appropriate response
  ➡️ Control the wake up, control the power usage

• Tiny is the resources, power, but not importance
  ➡️ Sensor has to transform its data into a first level of response,
  ➡️ It often serves as a first door to digital for users,
  ➡️ Must be accurate for the users and for the power
Sensor Node: An Always-On, Low Power, Sentinel

Mission

At Very Low Power
Always On
Identifies a pattern
Triggers the first layer of the main system

Sensor Node

Communication
Edge Processor
Sensors
Power

Smart Sensor

Low
Cost
Power
Size

Always on

50 to 500μA

Main System

Cloud
Thousands

Fog
Millions

Edge

5mA to 500mA

10 watts and above

Often off

Cost
Power
Size
<table>
<thead>
<tr>
<th>Power</th>
<th>Motion</th>
<th>Mic</th>
<th>Computation</th>
<th>Human wakes up in middle of the night</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td>Plugged in</td>
</tr>
<tr>
<td>&gt;1 A</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>Context, accurate Location</td>
<td>Speech to text</td>
<td>AP on, connecting to cloud (system and radio power)</td>
<td>Making breakfast, going to work</td>
</tr>
<tr>
<td>100 mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>Image stabilization, rough location</td>
<td>ANC, CW</td>
<td>Sensor hub on (fusion with other sensors)</td>
<td>Out of bed</td>
</tr>
<tr>
<td>2 mA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gestures Activity class, GAF</td>
<td>WW</td>
<td>On chip (edge) ML (windowed)</td>
<td>Sit up, lights on</td>
</tr>
<tr>
<td>250 µA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Bring2See Pedometer</td>
<td>VAD, AAD</td>
<td>DMP, Target for chip ML</td>
<td>Open eyes</td>
</tr>
<tr>
<td>75 µA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Being held/worn</td>
<td>No Audio</td>
<td>RTL, WoM</td>
<td>Listening, did I hear something</td>
</tr>
<tr>
<td>10 µA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td>Roll over</td>
</tr>
<tr>
<td>1 µA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>No motion</td>
<td></td>
<td>N-Zero program</td>
<td>Grunt</td>
</tr>
<tr>
<td>100 nA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td>Snoring</td>
</tr>
<tr>
<td>10 nA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Level</td>
<td>Current (mA)</td>
<td>Sensor Domain</td>
<td>Specific Actions</td>
<td>Duration</td>
</tr>
<tr>
<td>-------</td>
<td>-------------</td>
<td>---------------</td>
<td>-----------------</td>
<td>----------</td>
</tr>
<tr>
<td>8</td>
<td>&gt;1 A</td>
<td>Motion</td>
<td>Speech to text</td>
<td>Plugged in</td>
</tr>
<tr>
<td>7</td>
<td>100 mA</td>
<td>Power</td>
<td>Cloud (big iron), learning</td>
<td>Human wakes up in middle of the night</td>
</tr>
<tr>
<td>6</td>
<td>2 mA</td>
<td>Motion</td>
<td>Context, accurate Location</td>
<td>Making breakfast, going to work</td>
</tr>
<tr>
<td>5</td>
<td>250 µA</td>
<td>Motion</td>
<td>Gesture, Activity class, GAF</td>
<td>Image stabilization, rough location</td>
</tr>
<tr>
<td>4</td>
<td>75 µA</td>
<td>Motion</td>
<td>VAD, AAD</td>
<td>ANC, CW</td>
</tr>
<tr>
<td>3</td>
<td>10 µA</td>
<td>Motion</td>
<td>Being held/worn</td>
<td>Image stabilization, rough location</td>
</tr>
<tr>
<td>2</td>
<td>1 µA</td>
<td>Motion</td>
<td>No Audio</td>
<td>Gesture, Activity class, GAF</td>
</tr>
<tr>
<td>1</td>
<td>100 nA</td>
<td>Motion</td>
<td>No motion</td>
<td>VAD, AAD</td>
</tr>
<tr>
<td>0</td>
<td>10 nA</td>
<td>Motion</td>
<td>No motion</td>
<td>VAD, AAD</td>
</tr>
</tbody>
</table>

- **Sensor Domain**
  - Motion
  - Power
  - Computation
  - Speech to text
  - Cloud (big iron), learning
  - Making breakfast, going to work
  - Human wakes up in middle of the night

- **Specific Actions**
  - Context, accurate Location
  - Gesture, Activity class, GAF
  - VAD, AAD
  - Being held/worn
  - No motion
  - N-Zero program
  - Roll over
  - Listening, did I hear something
  - Grunt
  - Snoring
  - Listening, did I hear something
  - Grunt
  - Snoring

- **Duration**
  - Plugged in
  - Human wakes up in middle of the night
  - Making breakfast, going to work
  - Image stabilization, rough location
  - Gesture, Activity class, GAF
  - VAD, AAD
  - Being held/worn
  - No motion
  - N-Zero program
  - Roll over
  - Listening, did I hear something
  - Grunt
  - Snoring
  - Listening, did I hear something
  - Grunt
  - Snoring

- **Levels**
  - 8: >1 A
  - 7: 100 mA
  - 6: 2 mA
  - 5: 250 µA
  - 4: 75 µA
  - 3: 10 µA
  - 2: 1 µA
  - 1: 100 nA
  - 0: 10 nA

- **Units**
  - hours
  - days
  - months
  - years
System phased Wake Up View

<table>
<thead>
<tr>
<th>Time Unit</th>
<th>Power</th>
<th>Motion</th>
<th>Mic</th>
<th>Computation</th>
<th>Human Activity</th>
</tr>
</thead>
<tbody>
<tr>
<td>years</td>
<td>100 mA</td>
<td>Context, accurate Location</td>
<td>Speech to text</td>
<td>Cloud (big iron), learning</td>
<td>Making breakfast, going to work</td>
</tr>
<tr>
<td>months</td>
<td>250 µA</td>
<td>Gestures, Activity class, GAF</td>
<td>ANC, CW</td>
<td>Sensor hub on (fusion with other sensors)</td>
<td>Out of bed</td>
</tr>
<tr>
<td>days</td>
<td>75 µA</td>
<td>Bring2See, Pedometer</td>
<td>VAD, AAD</td>
<td>Sit up, lights on</td>
<td>On chip (edge) ML (windowed)</td>
</tr>
<tr>
<td>weeks</td>
<td>10 µA</td>
<td>Being held/worn</td>
<td>No Audio</td>
<td>Listening, did I hear something</td>
<td>Open eyes</td>
</tr>
<tr>
<td>days</td>
<td>1 µA</td>
<td>No motion</td>
<td></td>
<td>Roll over</td>
<td></td>
</tr>
<tr>
<td>days</td>
<td>100 nA</td>
<td></td>
<td>N-Zero program</td>
<td>Grunt</td>
<td></td>
</tr>
<tr>
<td>days</td>
<td>10 nA</td>
<td></td>
<td></td>
<td>Snoring</td>
<td></td>
</tr>
</tbody>
</table>

Keep Smallest Size, Lowest Power, All While Bringing More and More Features and Functionalities

Human wakes up in middle of the night

Motion, Power, Mic, Computation

Senso Domain

Best (FRR/FAR) are optimizing power

- Low FRR is key for the performance
  - False Rejection is when we miss a Wake Up,
  - In Consumer product, means a user fails his interaction with his device
  - Poor user experience

- Low FAR is key for the power,
  - Example of a Motion to wake up a SmartWatch (Bring to See/B2S)
    o The more B2S False Accepts (detect B2S where it shouldn’t),
    o the more power we burn uselessly by waking up the screen (turn on)
  - Example of a Voice Activity Detection (VAD) on Sensor to trigger a KeyWord Spotting (KWS on MCU) to wake up Automatic Speech Recognition (ASR) on Cloud
    o the more VAD False Accepts (detect Voice Activity when there’s none),
    o the more power we burn in useless KWS

Machine Learning with DNN is (best) candidate to provide optimized FAR/FRR Solution
What do we do?
Our Approach

Neural Networks Trained Model is any combination of:

- Dense Layers (DNN)
- Convolutional Layers (CNN)
- Recursive Layers (GRU, time series)
- Activations Layers (Tanh...)
- Decision Layers (Softmax...)

~10kB RAM, ROM, 4 MHz
50µA matters
Core SW Tech Automated Embedded code Generation: From TensorFlow to optimized integer C

- All Floats computations converted to integers (Matrix, Tanh.., less die size)
- Windows computations converted to singles frames (less memory, less CPU)
- Layers parameters reduced (Weights quantized to 8 bits, less memory)
- Accuracy tested and kept, reported.
- Code optimized and ready, run smoothly on M0+, M4..

### AAR example

<table>
<thead>
<tr>
<th>TF.Lite</th>
<th>TDK Quantizer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>303Ko</td>
</tr>
<tr>
<td>Model</td>
<td>6.72Ko</td>
</tr>
<tr>
<td>Stack</td>
<td>12Ko</td>
</tr>
<tr>
<td>Accuracy</td>
<td>85.5%</td>
</tr>
</tbody>
</table>

```python
def create_model_AudioNN(input_shape, nb_hidden, nb_class): #HM1-TDK 1.0.8 model
    layersizes=[48, 48, 48, 48]
dropout=[0.5, 0.5, 0.5, 0.3]
stride_size=1
x_input = Input(shape=input_shape)

# Step 1: Conv layer
x = Conv2D(layersizes[0], kernel_size=10, strides=stride_size, padding="same"')(x_input)
X = BatchNormalization()(x) # Batch normalization
x = Activation('relu')(x) # ReLU activation
x = Dropout(dropout[0])(x) # dropout (p=0.5)

# Step 2: First RNN layer
X = GRU(units=layersizes[1], return_sequence=True)(x)
X = BatchNormalization()(x) # Batch normalization

# Step 3: Second RNN layer
X = GRU(units=layersizes[2], return_sequence=True)(x)
X = BatchNormalization()(x) # Batch normalization

# Step 4: final dense layer
X = Dense(layersizes[3], activation="relu"')(x)
X = Dropout(dropout[3])(x)
X = TimeDistributed(Dense(nb_class, activation="sigmoid"))(x) # Time distributed (softmax)
model = Model(inputs=x_input, outputs=x)
return model
```

```c
int32_t ML2MCU_Predict_DNN(int32_t* pData, int32_t* pred_prob, int32_t* prediction)
{
    Conv1DForward(pData, pWork1, &conv1d_layer_0);
    pData = pWork1;
    BatchNormalization(pData, iBatchScaleQ8_2, iBatchOffsetQ8_2, 1.48);
    QuantizedRelu(pData, 0.5);
    GRUForward(pData, pWork2, &gru_layer_1);
    pData = pWork2;
    BatchNormalization(pData, iBatchScaleQ8_5, iBatchOffsetQ8_5, 1.40);
    GRUForward(pData, pWork1, &gru_layer_6);
    pData = pWork1;
    BatchNormalization(pData, iBatchScaleQ8_7, iBatchOffsetQ8_7, 1.40);
    GlobalGain(pData, iGainQ8_8, 0.4);
    VectorMatrixMul(pData, pWork2, &weights_8_40, 60);
    AddBias(pWork2, pBias_8_60);
    pData = pWork2;
    QuantizedRelu(pData, 0.5);
    GlobalGain(pData, iGainQ0_10, 60);
    TimeMultDistributed(pData, pWork1, pWeights_10, 0.5);
    TimBiasDistributed(pWork1, pBias_10, 1.0);
    pData = pWork1;
    SmoothSigmoidPiecewiseDesq(pData, 0);
    CopyTo(pData, pred_prob, 0);
    Threshold(pData, 0.5);
    CopyTo(pData, prediction, 0);
    return 0;
}
```
Machine Learning to Tiny MCU Flow

- Enable, ease and Automate the ML path to constrained MCUs (M Class, M4 SensorHub type to M0 for lowest power)
- Automate ML compact code size and MIPS reduction, through MCU specific design rules, Feature Selection,
- Automate MetaOptimization (e.g., NN Architecture, Activation Functions),
- Provide Standard Features library
Smart Watch Example

Bring to See
Bring2See for a SmartWatch: FAR/FRR

- Bring 2 See is to wakeup the Smart Watch Host MCU

**Performances**

- Uses raw data
- Improved latency of 95ms versus handcrafted

```
\begin{align*}
\text{False Reject Rate (in \%)} & : 10 \rightarrow 0 \\
\text{False Accepts per hour} & : 10 \rightarrow 100
\end{align*}
```

```
\begin{align*}
\text{ML Solution} & \quad + \\
\text{Hand Crafted Solution} & \quad +
\end{align*}
```

```
\text{“Perfect” zone} \\
\text{Low False Rejection Rate} \\
\text{Low False Accept Rate}
```

---

**Motion Sensor**

32KB/4MHz Clock

<table>
<thead>
<tr>
<th>Modules</th>
<th>ROM Size (bytes) ro code + ro data</th>
<th>RAM Size (bytes) rw data</th>
<th>Execution time (eq on M0+ 4MHz Pebble for 50 ms frames)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ml2mcu general functions</td>
<td>1202</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>b2s model and feature extraction</td>
<td>918</td>
<td>416</td>
<td>3.2ms</td>
</tr>
<tr>
<td>Total:</td>
<td>2210</td>
<td>416</td>
<td></td>
</tr>
</tbody>
</table>
Bring2See for a SmartWatch: Power

- Assuming 1066mWh battery

<table>
<thead>
<tr>
<th>Smartwatch</th>
<th>mW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sleeping</td>
<td>15</td>
</tr>
<tr>
<td>Dozing</td>
<td>32</td>
</tr>
<tr>
<td>Awake (2%)</td>
<td>310</td>
</tr>
<tr>
<td>Average</td>
<td>25.9</td>
</tr>
</tbody>
</table>


- HandCrafted (State Machine) VS Machine Learning False Accept Rate power Impact
  - HandCrafted optimized for User experience
  - Machine Learning point positioned with same FRR (same user experience)

<table>
<thead>
<tr>
<th></th>
<th>FA Rate</th>
<th>mW drained due to False Accepts</th>
<th>Battery Life (hours)</th>
</tr>
</thead>
<tbody>
<tr>
<td>ML</td>
<td>10 per hour</td>
<td>42.1</td>
<td>35.3</td>
</tr>
<tr>
<td>HandCrafted</td>
<td>100 per hour</td>
<td>30.2</td>
<td>25.3</td>
</tr>
</tbody>
</table>
Robotics Example

Audio Wake up and Commands
Low power embedded keyword spotting for robots

Use case

You’re served, can I go back?

Yes* / no*
Ex: “robot-chan iiyo!”

MCU-based versus Cloud
✓ Improved latency (250x)
✓ Reduced Power (80x)

Results & performance
- Fully embedded solution
- 16KHz Mic ODR
- 1 decision every 160 ms

Inputs & built dataset
- 300+ hours of data

Weeks to customize Robokit’s KeyWord Spotting
Use Case definition - Audio samples data collection
Audio samples check and post-processing - Data augmentation
Neural Network training - System integration
Performance Tests

Inputs & built dataset
- 300+ hours of data

Results & performance
- Fully embedded solution
- 16KHz Mic ODR
- 1 decision every 160 ms

McU-based versus Cloud
- Improved latency (250x)
- Reduced Power (80x)

Memory
<table>
<thead>
<tr>
<th></th>
<th>WW+Cloud</th>
<th>WW+CW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash</td>
<td>~46KB</td>
<td>~37.5KB</td>
</tr>
<tr>
<td>RAM</td>
<td>~8.5KB</td>
<td>~9.8KB</td>
</tr>
<tr>
<td>Stack</td>
<td>~1KB</td>
<td>~1.3KB</td>
</tr>
</tbody>
</table>

Exe time/frame
<table>
<thead>
<tr>
<th></th>
<th>WW+Cloud</th>
<th>WW+CW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.92ms</td>
<td>0.48ms</td>
</tr>
</tbody>
</table>

Extra cloud
<table>
<thead>
<tr>
<th></th>
<th>WW+Cloud</th>
<th>WW+CW</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2.5s</td>
<td>0.5ms</td>
</tr>
</tbody>
</table>

Detection accuracy on a difficult DB
- Wake word detection 98.68%
- Command detection 90.38%
- Full system WW + CM 89.02% (99.43% on a std DB)
- NN Misclassification 1.15%

Kepler test performance - Near Field No Background Noise - 12h negative

Eval 1
Eval 2
TWS Example

Low power Audio staged detection
Audio and Voice Activity Detector

- Audio activity detector is Microphone-based responsible to detect any sound event/activity
  - Runs in HW in extremely low power; First stage for voice detection

- Large band Accelerometer inside TWS or headphone to detect periods of active speech/voice
  - Runs in motion sensor chip MCUSS in very low power; a stage enabling keyword spotting (wake-up word)

- Keyword spotting based on microphone to detect wake-up word
  - Runs on sensor hub, or host MCU;

**TWS application example (battery 100mAh)**
Conclusions

• Sensors are the edge to the real world
  ➤ Smallest sentinels linked to bigger components
    o Are ~10kB, Not much MIPs,
    o Our definition of “Tiny”
  ➤ Have a mission to wake up from patterns
    o False Accept cost power,
    o False Rejects cost user experience.
  ➤ Machine Learning can make best use of Data to generate the Wake Up,

• ML brings the promise to save power across all devices without regression of experience
  ➤ We’re pushing ML to the “Tiny” level that will improve user experience / and save power

• The Critical Step in this process is the Quantizer / Compress the Network
  ➤ Our Challenge is “How to bring a unified Quantizer”,
    o Easy to use (automated)
    o Generates Integer and HW accelerated logic
    o Covers a multitude of NN Architectures
Copyright Notice

The presentation(s) in this publication comprise the proceedings of tinyML® Summit 2021. The content reflects the opinion of the authors and their respective companies. This version of the presentation may differ from the version that was presented at the tinyML Summit. The inclusion of presentations in this publication does not constitute an endorsement by tinyML Foundation or the sponsors.

There is no copyright protection claimed by this publication. However, each presentation is the work of the authors and their respective companies and may contain copyrighted material. As such, it is strongly encouraged that any use reflect proper acknowledgement to the appropriate source. Any questions regarding the use of any materials presented should be directed to the author(s) or their companies.

tinyML is a registered trademark of the tinyML Foundation.

www.tinyML.org