“Create a Voice-Controlled Pac-Man Video Game with Silicon Labs”

Dan Riedler – Silicon Labs

October 25, 2022
Thank you, tinyML Strategic Partners, for committing to take tinyML to the next Level, together.
Executive Strategic Partners
Arm AI Virtual Tech Talks

The latest in AI trends, technologies & best practices from Arm and our Ecosystem Partners.

Demos, code examples, workshops, panel sessions and much more!

Fortnightly Tuesday @ 4pm GMT/8am PT

Find out more: www.arm.com/techtalks
EDGE IMPULSE

The Leading Development Platform for Edge ML

dedgeimpulse.com
Advancing AI research to make efficient AI ubiquitous

Power efficiency
- Model design, compression, quantization, algorithms, efficient hardware, software tool

Personalization
- Continuous learning, contextual, always-on, privacy-preserved, distributed learning

Efficient learning
- Robust learning through minimal data, unsupervised learning, on-device learning

Perception
- Object detection, speech recognition, contextual fusion

Reasoning
- Scene understanding, language understanding, behavior prediction

Action
- Reinforcement learning for decision making

A platform to scale AI across the industry
Accelerate Your Edge Compute

SYNTIANT
Making Edge AI A Reality

www.syntiant.com
Fastest Video Analytics Solutions on Arm CPUs
High-Value or Safety-Critical Use Cases?

For your most important projects, use Reality AI Tools®

TinyML software that covers the full engineering lifecycle:

- AutoML for non-visual sensing based on advanced signal processing math
- Hardware design analytics
- Explanation of TinyML models in terms of underlying physics
- Automated Data Readiness assessment

https://reality.ai/  @SensorAI  info@reality.ai  Reality AI
Renesas is enabling the next generation of AI-powered solutions that will revolutionize every industry sector.
Gold Strategic Partners
Witness potential made possible at analog.com.
FOTA HUB

Making Over-the-Air Firmware and ML models Updates Simple and Accessible!

- Securely update your IoT devices regardless of their Hardware Platform (Silicon) Provider and physical location.
- Unlock TinyML business value through OTA Firmware and ML models update.
- Pay-as-you-go

www.fotahub.com
contact@fotahub.com
TOGETHER, WE ACCELERATE THE BREAKTHROUGHS THAT ADVANCE OUR WORLD

www.nxp.com/ai
Deploy TinyML into the Real World - Plug and Play ML

**Sensors:**
- modulated and ready-to-use sensors to simplify the setup process
- support 500+ Grove modules

**Wio Terminal:**
- completed AI platform integrated with a 2.4" LCD Screen, on-board IMU (LIS3DHTR), microphone, buzzer, microSD card slot, light sensor, infrared emitter (IR 940nm)

**Codecraft:**
- no code Programming platform to Get Started With TinML
- supports Arduino, Python, C or JavaScript etc.

**TensorFlow:**
- to easily train low memory usage machine learning models

**Edge Impulse:**
- to optimize data utilization and enable deploy a machine learning model faster than ever

**Inference:**
- Motion / Gesture / Speech / Smell / Sports
- Barcode / Face / Image

**Applications:**
- Artificial Nose
- AI Thermal Camera for Safe Camping
- Azure IoT Squirrel Feeder
The Right Edge AI Tools Can Make or Break Your Next Smart IoT Product

Analytics Toolkit Suite

AutoML

Data Collection

Model Training

Code Generation

Test & Validation

Team Collaboration

Version Control and Model Management

sensiml.com/tinyML
STMicroelectronics provides extensive solutions to make tiny Machine Learning easy
ENGINEERING
EXCEPTIONAL
EXPERIENCES

We engineer exceptional experiences for consumers in the home, at work, in the car, or on the go.

www.synaptics.com
**SynSense** builds sensing and inference hardware for ultra-low-power (sub-mW) embedded, mobile and edge devices. We design systems for real-time always-on smart sensing, for audio, vision, IMUs, bio-signals and more.

https://SynSense.ai
Silver Strategic Partners
Next tinyML Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, November 1</td>
<td>Pete Warden CEO, Useful Sensors</td>
<td>No-Code Edge ML</td>
</tr>
</tbody>
</table>

Webcast start time is 8:00 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting
Join Growing tinyML Communities:

meetup

tinyML - Enabling ultra-low Power ML at the Edge

LinkedIn

The tinyML Community
https://www.linkedin.com/groups/13694488/

11.7k members in 46 Groups in 37 Countries

3.1k members & 9.4k followers
Subscribe to tinyML YouTube Channel for updates and notifications (including this video)
www.youtube.com/tinyML
Our next tinyML Trailblazers Series
Success Stories with Massimo Banzi
(CTO and Co-Founder, Arduino)

LIVE ONLINE November 9th, 2022 at 8 am PDT

Register now!
<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, November 1</td>
<td>Pete Warden, CEO, Useful Sensors</td>
<td>No-Code Edge ML</td>
</tr>
</tbody>
</table>

Webcast start time is 8:00 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting
Reminders

Slides & Videos will be posted tomorrow

tinyml.org/forums  youtube.com/tinyml

Please use the Q&A window for your questions
Dan Riedler is a Principal Machine Learning Software Architect at Silicon Labs. He has been with the company for 5 years and has been in the industry for 13 years. For the past several years he has been helping to design and enable Silicon Lab's machine learning offerings. As part of the machine learning enablement, he was the primary contributor to the Silicon Labs Machine Learning Toolkit.
Welcome

Creating a Voice-Controlled Pac-Man Video Game

Dan Riedler
dan.riedler@silabs.com
Presentation Outline

- Overview of Silicon Labs’ machine learning (ML) enablement
- Voice-controlled Pac-Man demonstration
- Silicon Labs’ Machine Learning Toolkit (MLTK) overview
- Creating a keyword-spotting ML model
- Deploying an ML model to an embedded device
- Summary
- Q & A

siliconlabs.github.io/mltk
Silicon Labs Machine Learning Solution Benefits

- **Industry’s widest portfolio of wireless solutions combined with ML for Tiny Edge devices**
  - Bluetooth, 802.15.4/ZigBee/Thread, Matter, Z-Wave, Prop, Wi-Sun, Sidewalk

- **Integrated ML hardware accelerator (xG24) provides up to 8X faster ML inference with 1/6th of energy**
  - Reduces BOM, footprint and design complexity while minimizing latency

- **ML development tools and solutions for explorers to experts for faster application development**
  - Partnerships with Edge Impulse, SensiML, Sensory, and MicroAI accelerate embedded ML development
  - Tensorflow-Lite Micro supported in Silicon Labs’ Gecko SDK
  - Silicon Labs’ Machine Learning Toolkit (MLTK) on GitHub provides complete control & flexibility for the expert developers
  - We want customers to choose the tool that's best for them
Pac-Man Demo

Hardware and software overview with a live demonstration

siliconlabs.github.io/mltk
Hardware & Software Overview

Hardware
- DK2601B development kit
- xG24 wireless SoC with multi-protocol radio including Bluetooth Low-Energy (BLE)
- ARM® Cortex-M33 with TrustZone, 256 KB RAM and 1536 KB Flash, 80 MHz
- Machine learning hardware accelerator
- I2S digital microphone

Software
- Silicon Labs’ Machine Learning Toolkit (MLTK) with BLE audio classifier example application
  - Gecko SDK with BLE stack
  - Tensorflow-Lite Micro with custom kernels optimized for the ML hardware accelerator
  - Model training scripts, model profiler, model evaluation utilities
- Pac-Man video game webapp modified for usage with BLE
System Overview

1) User says keyword
2) xG24 classifies keyword using machine learning
3) Detected keyword sent to webpage via Bluetooth Low-Energy
4) Webpage video game updated
Live Demo

- **Live demo webpage**
  - mltk-pacman.web.app

- **Development board product page**
  - silabs.com/development-tools/wireless/efr32xg24-dev-kit

- **Demo documentation**
  - siliconlabs.github.io/mltk/mltk/tutorials/keyword_spotting_pacman
MLTK Overview

Overview of the Silicon Labs' Machine Learning Toolkit (MLTK)

siliconlabs.github.io/mltk
What is the MLTK?

"The Machine Learning Toolkit (MLTK) is a collection of scripts and utilities to aid the development of machine learning models for Silicon Labs' embedded MCUs"

- **Command-line interface** for executing various tasks, e.g. profile, train, evaluate, etc.
- **Python interface** for programmatic access to utilities and scripts
- **C++ Python wrappers** for sharing code between model training and embedded inference
- **Reference ML models** providing complete examples of how to generate models for embedded execution
- **Example applications** demonstrating how to load and execute the ML models on embedded devices
Why Use the MLTK?

- Training logic may spread across numerous scripts, the MLTK allows for combining into single script
  - Dataset generation, training, evaluation, quantization
  - Can help to reduce complexity and make model generation reproducible and distributable

- Any data preprocessing used during training must be used on the MCU, the MLTK features C++ Python wrappers
  - Allows for sharing code between model training scripts and embedded inference

- Critical that preprocessing settings used during training are distributed with model, MLTK allows for embedding settings into model file
  - Ensures that preprocessing settings and ML model are an atomic unit

- Many different software libraries needed to create model and execute on MCU, the MLTK provides seamless integration
  - Tensorflow, Tensorflow-Lite, Tensorflow-Lite Micro, Gecko SDK
  - Consistent development experience during model training and embedded runtime
What comes with the MLTK?

- **Model profiler**
  - Provides model memory usage, latency, CPU cycles, energy, etc.
  - Simulator and physical device supported

- **Audio Feature Generator C++ Python wrapper**
  - Converts streaming audio to spectrograms
  - C/C++ algorithms shared between model training and embedded MCU

- **Model generation scripts**
  - Data preprocessing, model training, evaluation, quantization

- **C++ example applications**
  - End-to-end examples of common ML use cases

- **Other utilities and documentation**
MLTK Resources

- Online documentation
  - siliconlabs.github.io/mltk

- GitHub source code
  - github.com/siliconlabs/mltk

- Python package
  - pypi.org/project/silabs-mltk

- Pac-Man tutorial
  - siliconlabs.github.io/mltk/mltk/tutorials/keyword_spotting_pacman
Creating a Keyword Spotting Model

Overview of how to create a keyword spotting machine learning model using the MLTK

siliconlabs.github.io/mltk
1) Audio signal from microphone captured

2) Audio signal converted to spectrogram (2D, grayscale image)

3) Convolutional Neural Network (CNN) classifies spectrogram

4) Post-process model output and send detected keyword to application

<table>
<thead>
<tr>
<th>Class ID</th>
<th>Keyword</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Left</td>
</tr>
<tr>
<td>1</td>
<td>Right</td>
</tr>
<tr>
<td>2</td>
<td>Up</td>
</tr>
<tr>
<td>3</td>
<td>Down</td>
</tr>
<tr>
<td>4</td>
<td>Stop</td>
</tr>
<tr>
<td>5</td>
<td>Go</td>
</tr>
</tbody>
</table>

More details in the MLTK documentation
Data Preprocessing with the Audio Feature Generator

- Data preprocessing must be consistent between training and embedded run-time
  - Any divergence may cause the embedded ML model to "see" different data and reduce accuracy

- Audio Feature Generator utility allows for sharing code between environments
  - Converts audio signal into spectrogram
  - Provided as Gecko SDK C/C++ library
  - C/C++ library accessible to Python training scripts via Python wrapper
  - Library settings configured in Python are automatically inserted into ML model
  - Ensures spectrograms generated during training "look" the same as what the MCU generates at run-time

More details in the MLTK documentation
Visualizing the Audio Feature Generator Parameters

- Many settings needed to generate spectrogram
  - Sample rate, window size, window step, upper-band frequency, etc.

- Audio visualization utility allows for adjusting settings and viewing generated spectrogram in real-time
  - Exact source code that would run on MCU used to generate spectrogram

- Simple command-line:
  - `mltk view_audio`

More details in the MLTK documentation
Finding an Optimal Model Architecture Using the Model Profiler

- **Latency is critical for in-game experience**
  - An inference should take less than 100ms

- **Profile model *before* training to ensure it meets requirement**
  1. Create model specification (i.e. layers, number of filters, etc.)
  2. Build model (e.g. *tflite*) using a few epochs and subset of dataset
  3. Program model to device (or simulator) and profile execution
  4. If inference time too high then return to step 1) and reduce model size
  5. Train optimal model

- **Simple command-line:**
  - `mltk profile my_model --build --device --accelerator mvp`

More details in the MLTK documentation
Training in the Cloud

- Model training can be very computationally expensive
  - i.e. It can a long time to train a model on a laptop

- Many times, bottleneck is CPU and not GPU
  - Data preprocessing commonly done on CPU
  - Takes longer to generated batch of augmented training samples than to do back-propagation on GPU

- Many third-party vendors allow for renting cloud machines
  - Selection of machines with 1+ GPUs and 64+ CPU cores $< 2/hr.
  - Provide SSH access to Linux environment
Training in the Cloud (cont.)

- **MLTK SSH command allows for seamlessly training model on remote machine**
  - All training output files automatically downloaded to local machine
- **Simple command-line:**
  - `mltk ssh train my_model`

More details in the MLTK documentation
Evaluating Using the Audio Classifier Utility

▪ After training, the MLTK will automatically evaluate the model
  • Class accuracies, confusion matrix, etc.

▪ Also helpful to run model on physical device
  • Accounts for sensor capture and preprocessing discrepancies
  • Accounts for MCU latency

▪ Audio classification utility allows for executing model in real-time
  • Supports PC or development board
  • Supports dumping audio and generated spectrograms

▪ Simple command-line:
  • `mltk classify_audio my_model --device --accelerator mvp --verbose`

More details in the MLTK documentation
Deploying the ML Model

Overview of how to deploy the ML model to the development board

siliconlabs.github.io/mltk
TF-Lite Model Generation

- After training completes the model is automatically quantized and converted to the .tflite model format
  - Uses TFLiteConverter to quantize weights/filters from float32 to int8
  - .tflite model executes in Tensorflow-Lite Micro Python wrapper to determine tensor arena (i.e. RAM) size
    ▶ Also verifies if the .tflite can fully execute on MCU

- Audio Feature Generator settings embedded into .tflite model as "metadata"
  - Settings used to generate spectrogram are inserted into .tflite model and read by MCU at run-time
  - Ensures settings used during model training (on PC) and model inference (on MCU) are in lock-step

More details in the MLTK documentation
Running the BLE Audio Classification App on MCU

- **MLTK features general purpose BLE audio classification example app**
  - Continuously captures microphone audio, generates spectrograms, and classifies
  - Detected keywords are written as a BLE GATT server notification

- **Audio Feature Generator to generate spectrograms from audio**
  - Settings dynamically loaded from .tflite model

- **Tensorflow-Lite Micro for inference**
  - .tflite model loaded from MCU flash memory
  - Custom kernels to accelerate model execution on MCU hardware accelerator

More details in the MLTK documentation
Summary

- The MLTK helps to bridge the gap between model training and embedded execution
  - Python wrappers to share code between training scripts and MCU run-time
  - Embed training hyperparameters into .tflite model file for later retrieval by MCU

- The MLTK features a rich set of tools to aid the development of embedded ML models
  - The tools are optional and may be used independently

- The MLTK offers end-to-end examples of common ML use cases
  - Demonstrates how to develop ML models and how to run them on an MCU
Thank You

siliconlabs.github.io/mltk
Copyright Notice

This multimedia file is copyright © 2022 by tinyML Foundation. All rights reserved. It may not be duplicated or distributed in any form without prior written approval.

tinyML® is a registered trademark of the tinyML Foundation.

www.tinyml.org
Copyright Notice

This presentation in this publication was presented as a tinyML® Talks webcast. The content reflects the opinion of the author(s) and their respective companies. 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