“Tiny machine learning with Arduino”

Massimo Banzi - Arduino

UK Area Group – April 20, 2021
tinyML Talks Sponsors

arm

Deeplite

EDGE IMPULSE

maxim integrated

Qeexo

Reality AI

SynSense

Additional Sponsorships available – contact Olga@tinyML.org for info
Optimized models for embedded
Runtime (e.g. TensorFlow Lite Micro)
Optimized low-level NN libraries (i.e. CMSIS-NN)
RTOS such as Mbed OS
Arm Cortex-M CPUs and microNPUs

1. Connect to high-level frameworks
   - Profiling and debugging tooling such as Arm Keil MDK

2. Supported by end-to-end tooling
   - Application

3. Connect to Runtime

Stay Connected
- @ArmSoftwareDevelopers
- @ArmSoftwareDev

Resources: developer.arm.com/solutions/machine-learning-on-arm
WE USE AI TO MAKE OTHER AI FASTER, SMALLER AND MORE POWER EFFICIENT

Automatically compress SOTA models like MobileNet to <200KB with little to no drop in accuracy for inference on resource-limited MCUs.

Reduce model optimization trial & error from weeks to days using Deeplite's design space exploration.

Deploy more models to your device without sacrificing performance or battery life with our easy-to-use software.

BECOME BETA USER bit.ly/testdeeplite
TinyML for all developers

Dataset
Acquire valuable training data securely

Enrich data and train ML algorithms

Edge Device
Real sensors in real time
Open source SDK

Embedded and edge compute deployment options

Test

Test impulse with real-time device data flows

Copyright © EdgeImpulse Inc.
www.edgeimpulse.com
<table>
<thead>
<tr>
<th>Advanced AI Acceleration IC</th>
<th>Low Power Cortex M4 Micros</th>
<th>Sensors and Signal Conditioning</th>
</tr>
</thead>
<tbody>
<tr>
<td>The new MAX78000 implements AI inferences at low energy levels, enabling complex audio and video inferencing to run on small batteries. Now the edge can see and hear like never before. <a href="www.maximintegrated.com/MAX78000">www.maximintegrated.com/MAX78000</a></td>
<td>Large (3MB flash + 1MB SRAM) and small (256KB flash + 96KB SRAM, 1.6mm x 1.6mm) Cortex M4 microcontrollers enable algorithms and neural networks to run at wearable power levels. <a href="www.maximintegrated.com/microcontroller">www.maximintegrated.com/microcontroller</a></td>
<td>Health sensors measure PPG and ECG signals critical to understanding vital signs. Signal chain products enable measuring even the most sensitive signals. <a href="www.maximintegrated.com/sensor">www.maximintegrated.com/sensor</a></td>
</tr>
</tbody>
</table>
Qeexo AutoML

Automated Machine Learning Platform that builds tinyML solutions for the Edge using sensor data

**Key Features**

- Supports 17 ML methods:
  - Multi-class algorithms: GBM, XGBoost, Random Forest, Logistic Regression, Gaussian Naive Bayes, Decision Tree, Polynomial SVM, RBF SVM, SVM, CNN, RNN, CRNN, ANN
  - Single-class algorithms: Local Outlier Factor, One Class SVM, One Class Random Forest, Isolation Forest
- Labels, records, validates, and visualizes time-series sensor data
- On-device inference optimized for low latency, low power consumption, and small memory footprint applications
- Supports Arm® Cortex™- M0 to M4 class MCUs

**End-to-End Machine Learning Platform**

**Target Markets/Applications**

- Industrial Predictive Maintenance
- Smart Home
- Wearables
- Automotive
- Mobile
- IoT

For more information, visit: www.qeexo.com
Add Advanced Sensing to your Product with Edge AI / TinyML

Pre-built Edge AI sensing modules, plus tools to build your own

**Reality AI solutions**
- Prebuilt sound recognition models for indoor and outdoor use cases
- Solution for industrial anomaly detection
- Pre-built automotive solution that lets cars “see with sound”

**Reality AI Tools® software**
- Build prototypes, then turn them into real products
- Explain ML models and relate the function to the physics
- Optimize the hardware, including sensor selection and placement
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
Successful tinyML Summit 2021:

- **5** days of tutorials, talks, panels, breakouts, symposium
  - **4** tutorials
  - **6** keynotes & **6** plenary tinyTalks (more in breakouts)
  - **2** panel discussions
  - **5** disruptive news presentations
  - **17** breakout/partner sessions
  - **6** Best Product and Innovation Award Finalists & Presentations
  - **89** Speakers

- **5006** registered attendees representing:
  - **104** countries, **1000+** companies and **400+** academic institutions

- **26** Sponsoring companies

www.youtube.com/tinyML with 150+ videos

tinyML Summit-2022, January 24-26, Silicon Valley, CA
June 7-10, 2021 (virtual, but LIVE)
Deadline for abstracts: May 1

Sponsorships are being accepted: sponsorships@tinyML.org
Next tinyML Talks

<table>
<thead>
<tr>
<th>Date</th>
<th>Presenter</th>
<th>Topic / Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tuesday, April 27</td>
<td>Michael Jo and Xingheng Lin</td>
<td>Train-by-weight (TBW): Accelerated Deep Learning by Data Dimensionality Reduction</td>
</tr>
<tr>
<td></td>
<td>Rose-Hulman Institute of Technology</td>
<td></td>
</tr>
</tbody>
</table>

Webcast start time is 8 am Pacific time

Please contact talks@tinyml.org if you are interested in presenting
Local UK Committee

Alessandro Grande  
Developer advocate & ecosystem manager, Arm

Dominic Binks  
VP Technology, Audio Analytic

Gian Marco Iodice  
ML Techlead, Arm

Neil Cooper  
VP Marketing, Audio Analytic
Reminders

- Slides & Videos will be posted tomorrow
- Please use the Q&A window for your questions
- tinyml.org/forums
- youtube.com/tinyml
Massimo Banzi

Massimo Banzi is the co-founder of the Arduino project. He is an Interaction Designer, Educator and Open Source Hardware advocate. He has worked as a consultant for clients such as: Prada, Artemide, Persol, Whirlpool, V&A Museum and Adidas. Massimo started the first FabLab in Italy which led to the creation of Officine Arduino, a FabLab/Makerspace based in Torino. He spent 4 years at the Interaction Design Institute Ivrea as Associate Professor. Massimo has taught workshops and has been a guest speaker at institutions all over the world. Before joining IDII he was CTO for the Seat Ventures incubator. He spent many years working as a software architect, both in Milan and London, on projects for clients like Italia Online, Sapient, Labour Party, BT, MCI WorldCom, SmithKlineBeecham, Storagetek, BSkyB and boo.com. Massimo is also the author of “Getting Started with Arduino” published by O’Reilly. He is a regular contributor to the Italian edition of Wired Magazine and Che Futuro, an online magazine about innovation. He currently teaches Interaction Design at SUPSI Lugano in the south of Switzerland and is a visiting professor at CIID in Copenhagen.
Introduction

In this talk we walk through some examples of TinyML on Arduino devices to show how simple it is to build working examples of ML with minimal coding using three different platforms compatible with Arduino.
Enabling anyone to innovate by making complex technologies simple to use.

Arduino
29M+ Users/Year on arduino.cc
900K+ Arduino Forum Users
17M+ IDE downloads/year
1.2M+ Users of Arduino Create Cloud
4000+ Libraries Available for Arduino

923K
302K
408K
Arduino Today

Easy for
Beginners,
Fast for
professionals
Arduino Hardware
Arduino Nano BLE Sense

- Nano form factor
- nRF52840 Arm Cortex M4f @ 64MHz
- 1MByte Flash
- 256Kb Ram
- Bluetooth 5
- Sensors: IMU, Microphone, Gesture, Light, Proximity, Pressure, Temperature, Humidity
Tiny Machine Learning Kit

- VGA Camera and Nano BLE Sense
- Used for the EdX course on TinyML
The Future of ML is Tiny and Bright

Professional Certificate in
Tiny Machine Learning (TinyML)

https://www.edx.org/professional-certificate/harvardx-tiny-machine-learning

- Fundamentals of machine learning and embedded devices.
Welcome

Welcome message for CS249r @ Harvard.

Course Overview

Tiny Machine Learning (TinyML) is an introductory course at the intersection of Machine Learning and Embedded IoT Devices. The pervasiveness of ultra-low-power embedded devices, coupled with the introduction of embedded machine learning frameworks like TensorFlow Lite for Microcontrollers, will enable the mass proliferation of AI-powered IoT devices. The explosive growth in machine learning and the ease of use of platforms like TensorFlow (TF) make it an indispensable topic of study for modern computer science and electrical engineering students.

https://sites.google.com/g.harvard.edu/tinyml/home

The course will cover all things related to machine learning on embedded devices. The topics range from TinyML applications and algorithms to the design of the frameworks built for operating ML on embedded systems to the microcontroller hardware designed, optimized, and built for ultra-low-power, ultra-efficient operation.
Portenta H7

- Portenta form factor
- Dual core
  - Arm Cortex M7@480MHz
  - Arm Cortex M4@240MHz
- up to 64 MByte SDRAM
- up to 128 MByte QSPI Flash
- Common Criteria Crypto
- WiFi b/g/n 65Mbps + BT 5.1
- 100Mbit Ethernet
- CAN
- USB-C with DisplayPort output
Portenta Vision Shield

- Portenta form factor
- Low power cameras
  - Himax HM-01B0
  - Himax HM-0360
- Dual microphone (beamforming)
- Ethernet
- Lora
- SD Card
/* Copyright 2019 The TensorFlow Authors. All Rights Reserved.

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*/

#include <TensorFlowLite.h>
#include "main_functions.h"
#include "accelerometer_handler.h"
#include "constants.h"
#include "gesture_predictor.h"
#include "magic_wand_model_data.h"
#include "output_handler.h"
#include "tensorflow/lite/micro/micro_error_reporter.h"
#include "tensorflow/lite/micro/micro_interpreter.h"
#include "tensorflow/lite/micro/micro_op_resolver.h"
#include "tensorflow/lite/schema/schema_generated.h"
#include "tensorflow/lite/version.h"

// Globals, used for compatibility with Arduino-style sketches.
namespace {
  tflite::ErrorReporter* error_reporter = nullptr;
  const tflite::Model* model = nullptr;
  tflite::MicroInterpreter* interpreter = nullptr;
  TfliteTensor* model_input = nullptr;
  int input_length;
}

// Create an area of memory to use for input, output, and intermediate access.
```python
import pyb
import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE)
sensor.set_framesize(sensor.VGA)
sensor.skip_frames(time = 2000)

thresholds = (100, 255)
ledRed = pyb.LED(1)
ledGreen = pyb.LED(2)
ledBlue = pyb.LED(3)

clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()

    # Find blobs
    blobs = img.find_blobs([thresholds], area_threshold=200, merge=False)

    # Draw blobs
    for blob in blobs:
        img.draw_rectangle(blob.rect(), color=255)
        img.draw_cross(blob.cx(), blob.cy(), color=255)

    # Toggle LEDs
    if len(blobs) > 0:
        ledGreen.on()
        ledRed.off()
    else:
        ledGreen.off()
        ledRed.on()

time.sleep(50)

print(clock.fps())
```
Examples
Arduino Tiny ML
TensorFlow Lite for Microcontrollers is an experimental port of TensorFlow Lite designed to run machine learning models on microcontrollers and other devices with only kilobytes of memory.

It doesn't require operating system support, any standard C or C++ libraries, or dynamic memory allocation. The core runtime fits in 16 KB on an Arm Cortex M3, and with enough operators to run a speech keyword detection model, takes up a total of 22 KB.

There are example applications demonstrating the use of microcontrollers for tasks including wake word detection, gesture classification from accelerometer data, and image classification using camera data.

Get started

To try the example applications and learn how to use the API, read Get started with microcontrollers.
We’re going to use the code provided here:
https://github.com/arduino/ArduinoTensorFlowLiteTutorials/tree/master/GestureToEmoji
Visualize the IMU Data

- Open `ArduinoSketches/IMU_Capture/IMU_Capture.ino` in the Arduino IDE.
- Compile the sketch and upload it to the board: Sketch -> Upload
- Open the Serial Monitor: Tools -> Serial Monitor
- Press the button, IMU data will be captured and outputted for 1 second
- Close the Serial Monitor window
- Open the Serial Plotter: Tools -> Serial Plotter
- Press the button, and perform a gesture
- You'll see a graph of the data capture
- Repeat capturing various gestures to get a sense of what the training data will look like
- Close the Serial Plotter
<table>
<thead>
<tr>
<th>aX, aY, aZ, gX, gY, gZ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.704, -0.221, 0.726, 35.156, -70.557, 31.921</td>
</tr>
<tr>
<td>0.815, -0.241, 0.612, 50.049, -68.298, 36.255</td>
</tr>
<tr>
<td>0.767, -0.212, 0.686, 35.461, -81.909, 31.494</td>
</tr>
<tr>
<td>0.782, -0.244, 0.716, 36.316, -88.318, 22.522</td>
</tr>
<tr>
<td>0.822, -0.276, 0.724, 44.067, -88.928, 17.639</td>
</tr>
<tr>
<td>0.795, -0.320, 0.791, 44.556, -88.257, 14.221</td>
</tr>
<tr>
<td>0.787, -0.401, 0.918, 40.466, -90.942, 13.916</td>
</tr>
<tr>
<td>0.712, -0.511, 1.042, 31.982, -95.581, 13.062</td>
</tr>
<tr>
<td>0.676, -0.651, 1.153, 26.184, -80.688, 18.127</td>
</tr>
<tr>
<td>0.641, -0.719, 1.207, 20.142, -59.998, 29.053</td>
</tr>
<tr>
<td>0.616, -0.763, 1.294, 14.160, -27.832, 44.006</td>
</tr>
<tr>
<td>0.627, -0.889, 1.324, 7.690, 0.427, 60.425</td>
</tr>
<tr>
<td>0.492, -0.974, 1.323, -3.174, 24.414, 84.412</td>
</tr>
<tr>
<td>0.522, -0.931, 1.305, -7.019, 39.001, 120.178</td>
</tr>
<tr>
<td>0.517, -0.836, 1.226, -11.230, 59.875, 159.729</td>
</tr>
<tr>
<td>0.604, -0.694, 1.173, -18.555, 85.083, 183.044</td>
</tr>
<tr>
<td>0.684, -0.309, 1.319, -38.940, 111.633, 184.143</td>
</tr>
</tbody>
</table>

**TensorFlow Lite for Microcontrollers / Gesture Recognition**
TensorFlow Lite for Microcontrollers / Gesture Recognition
TensorFlow Lite for Microcontrollers / Gesture Recognition
Gather Training Data

- Press the reset button on the board
- Open the Serial Monitor: Tools -> Serial Monitor
- Make a punch gesture with the board in your hand - you should see the sensor data log in the Serial Monitor
- Repeat 10 times to gather more data
- Copy and paste the data from the serial output to new text file called punch.csv using your favorite text editor
- Close the Serial Monitor
- Press the reset button on the board
- Open the Serial Monitor: Tools -> Serial Monitor
- Make a flex gesture with the board in your hand
- Repeat 10 times
- Copy and paste the serial output to new text file flex.csv using your favorite text editor
Tiny ML on Arduino

Setup Python Environment

Upload Data

Graph Data (optional)

Train Neural Network

- Parse and prepare the data
  - Randomize and split the input and output pairs for training
  - Build & Train the Model
  - Verify
    - Graph the loss
    - Graph the loss again, skipping a bit of the start
    - Graph the mean absolute error
    - Run with Test Data
- Convert the Trained Model to TensorFlow Lite
- Encode the Model in an Arduino Header File
- Classifying IMU Data

Gesture recognition tutorial
- Sandeep Mistry - Arduino
- Don Coleman - Chariot Solutions


Setup Python Environment

The next cell sets up the dependencies in required for the notebook, run it.

```
# Setup environment
!apt-get -qq install xxd
!pip install pandas numpy matplotlib
!pip install tensorflow==2.6.0-rc1
```

Upload Data

1. Open the panel on the left side of Colab by clicking on the >
2. Select the files tab
Machine Learning

- Go to this colab:
- Drag punch.csv and flex.csv files from your computer to the tab to upload them into colab.
- Follow the instructions on the Notebook and you’ll end up with “model.h” file you can import into arduino
Classifying IMU Data

• Open ArduinoSketches/IMU_Classifier/IMU_Classifier.ino in the Arduino IDE.
• Switch to the model.h tab
• Replace the contents of model.h with the version you downloaded from Colab
• Upload the sketch: Sketch -> Upload
• Open the Serial Monitor: Tools -> Serial Monitor
• Press the button, and perform a gesture
• The confidence of each gesture will be printed to the Serial Monitor (0 -> low confidence, 1 -> high confidence)
<table>
<thead>
<tr>
<th>punch</th>
<th>flex</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.529982</td>
<td>0.470548</td>
</tr>
<tr>
<td>0.917286</td>
<td>0.082714</td>
</tr>
<tr>
<td>0.877585</td>
<td>0.122415</td>
</tr>
<tr>
<td>0.000002</td>
<td>0.999997</td>
</tr>
<tr>
<td>0.176803</td>
<td>0.823197</td>
</tr>
<tr>
<td>0.999973</td>
<td>0.000027</td>
</tr>
<tr>
<td>0.077787</td>
<td>0.922213</td>
</tr>
<tr>
<td>0.036865</td>
<td>0.963135</td>
</tr>
</tbody>
</table>

**TensorFlow Lite for Microcontrollers / Gesture Recognition**
Gesture Controlled USB Emoji Keyboard

• The Emoji_Button example, ArduinoSketchs/Emoji_Button/Emoji_Button.ino, shows how to create a USB keyboard that prints and emoji character. (This only works on Linux and macOS, so if you're running Windows, find a friend to work on this exercise.)

• Try combining the Emoji_Button example with the IMU_Classifier sketch to create an gesture controlled emoji keyboard. 👊
TensorFlow Lite for Microcontrollers / Gesture Recognition
OpenMV / Tensorflow / Portenta Vision Shield / Image Recognition
REQUIRED HARDWARE AND SOFTWARE

• Portenta H7 board
• Portenta Vision Shield - LoRa or Portenta Vision Shield - Ethernet
• USB-C cable (either USB-A to USB-C or USB-C to USB-C)
• An Edge Impulse account for training the ML model
• Fruits (or other objects) to create the classification model 🍏🍌🍐
Open a document

- File > Open File (%O)
- File > Examples
- File > Recent Files

Drag and drop files here.
# Dataset Capture Script - By: sebastianhunkeler - Fri Mar 5 2021

# Use this script to control how your OpenMV Cam captures images for your data
# You should apply the same image pre-processing steps you expect to run on im
# that you will feed to your model during run-time.

import sensor, image, time

sensor.reset()
sensor.set_pixformat(sensor.GRAYSCALE) # Modify as you like.
sensor.set_framesize(sensor.QVGA) # Modify as you like.
sensor.skip_frames(time = 2000)

clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    # Apply lens correction if you need it.
    # img.lens_corr()
    # Apply rotation correction if you need it.
    # img.rotation_corr()
    # Apply other filters...
    # E.g. mean/median/mode/midpoint/etc.
    print(clock.fps())
Script - By: sebastianhunkeler - Fri Mar 5 2021

To control how your OpenMV Cam captures images for you, you can run the same image pre-processing steps you expect to run on the same images, then store the results in your model during run-time.

clock = time.clock()

while(True):
    clock.tick()
    img = sensor.snapshot()
    # Apply lens correction if you need it.
    img.lens_corr()
    # Apply rotation correction if you need it.
    img.rotation_corr()
    # Apply other filters...
    # E.g. mean/median/mode/mode/midpoint/etc
DATA ACQUISITION (FRUIT-DETECTOR)

Training data  Test data

DATA COLLECTED 1,142 items

LABELS 4

Record new data

No devices connected to the remote management API.

Collected data

<table>
<thead>
<tr>
<th>SAMPLE NAME</th>
<th>LABEL</th>
<th>ADDED</th>
<th>LENGTH</th>
</tr>
</thead>
<tbody>
<tr>
<td>banana.banana-0</td>
<td>banana</td>
<td>Today, 17:10:55</td>
<td>-</td>
</tr>
<tr>
<td>banana.banana-0</td>
<td>banana</td>
<td>Today, 17:10:55</td>
<td>-</td>
</tr>
<tr>
<td>banana.banana-0</td>
<td>banana</td>
<td>Today, 17:10:55</td>
<td>-</td>
</tr>
<tr>
<td>banana.banana-0</td>
<td>banana</td>
<td>Today, 17:10:55</td>
<td>-</td>
</tr>
<tr>
<td>banana.banana-0</td>
<td>banana</td>
<td>Today, 17:10:55</td>
<td>-</td>
</tr>
</tbody>
</table>

RAW DATA

banana.banana-0003.jpg.21rt7sgo
An impulse takes raw data, uses signal processing to extract features, and then uses a learning block to classify new data.
Training set

Data in training set: 1,142 items

Classes: 4 (apple, banana, pear, unknown)

Feature explorer (1,142 samples)

Feature generation output

Mon Mar 29 16:07:20 2021 Construct embedding
completed 0 / 500 epochs
completed 50 / 500 epochs
completed 100 / 500 epochs
completed 150 / 500 epochs
completed 200 / 500 epochs
completed 250 / 500 epochs
completed 300 / 500 epochs
completed 350 / 500 epochs
Still running...
completed 400 / 500 epochs
completed 450 / 500 epochs
Mon Mar 29 16:07:25 2021 Finished embedding
Reducing dimensions for visualizations OK
Job completed
Turn your impulse into optimized source code that you can run on any device.

- C++ library
- Arduino library
- Cube.MX CMSIS-PACK
- WebAssembly

Build firmware

Or get a ready-to-go binary for your development board that includes OpenMV cameras.

- Eta Compute ECM3532 AI Vision
- Himax WE-1 Plus
- OpenMV Library

Build
confidence = predictions_list[i][1]
label = predictions_list[i][0]
print("%s = %f" % (label[2:], confidence))

if confidence > 0.9 and label != "unknown":
    print("It's a ", label, "!")
clock = time.clock()
while(True):
    clock.tick()

    img = sensor.snapshot()

    # default settings just do one detection... change them to search
    for obj in tf.classify(net, img, min_scale=1.0, scale_mul=0.8, x):
        print("**********\nPredictions at [x=%d,y=%d,w=%d,h=%d]\n%" %
        img.draw_rectangle(obj.rect()))
    # This combines the labels and confidence values into a list
    predictions_list = list(zip(labels, obj.output))

    for i in range(len(predictions_list)):
        confidence = predictions_list[i][1]
        label = predictions_list[i][0]
        print("%s %f %f %f (label, confidence)" %
        if confidence > 0.9 and label != "unknown":
            print("It's a %s, label, %f")

print(clock.fps(), "fps")
https://www.arduino.cc/pro/tutorials/portenta-h7/vs-openmv-ml
Responding to your voice

In this tutorial, you'll use machine learning to build a system that can recognize audible events, particularly your voice through *audio classification*. The system you create will work similarly to "Hey Siri" or "OK, Google" and is able to recognize keywords or other audible events, even in the presence of other background noise or background chatter.

You'll learn how to collect audio data from microphones, use signal processing to extract the most important information, and train a deep neural network that can tell you whether your keyword was heard in a given clip of audio. Finally, you'll deploy the system to an embedded device and evaluate how well it works.

At the end of this tutorial, you'll have a firm understanding of how to classify audio using Edge Impulse.

There is also a video version of this tutorial:

Build Your own ML-powered keyword spotting model in 30 KB RAM
Models

Model History

<table>
<thead>
<tr>
<th>DATE</th>
<th>DATASETS</th>
<th>BUILD ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>10:21:09</td>
<td>Lamp - SWIFE Lamp - BACKGROUND</td>
<td>000001</td>
</tr>
</tbody>
</table>

FEATURE SELECTION: Off

HYPERPARAMETER OPTIMIZATION INTERVAL (sec): 1024

PREDICTION SAMPLE INTERVAL (sec): 250

CLASSIFICATION INTERVAL (sec): 0

BUILD STATUS:
- [ ] Ready
- [ ] Error
- [ ] Fail

PERFORMANCE SUMMARY:
- [ ] Run

ML MODEL | CROSS VALIDATION | LATENCY | SIZE | TRAINING DETAILS | SAVE | PUSH TO HARDWARE | LIVE CLASSIFICATION ANALYSIS |
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Artificial Neural Network</td>
<td>1.0 +/- 0.00</td>
<td>4.5 ms</td>
<td>83.4 KB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Convolutional Neural Network</td>
<td>1.0 +/- 0.00</td>
<td>4.5 ms</td>
<td>333 KB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Gradient Boosting Machine</td>
<td>1.0 +/- 0.00</td>
<td>4.5 ms</td>
<td>807 KB</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Logistic Regression</td>
<td>1.0 +/- 0.00</td>
<td>4.3 ms</td>
<td>360 B</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

ROC Curve

Learning Curve

Qeexo / https://github.com/qeexo/Qeexo_AutoML_Arduino_Nano33BLE_pub
That’s a wrap, Thank you!

Massimo Banzi
askmassimo@arduino.cc
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