Tiny ML

Tassadaq Hussain

Professor Namal University

Director Centre for AI and Big Data

Collaborations:

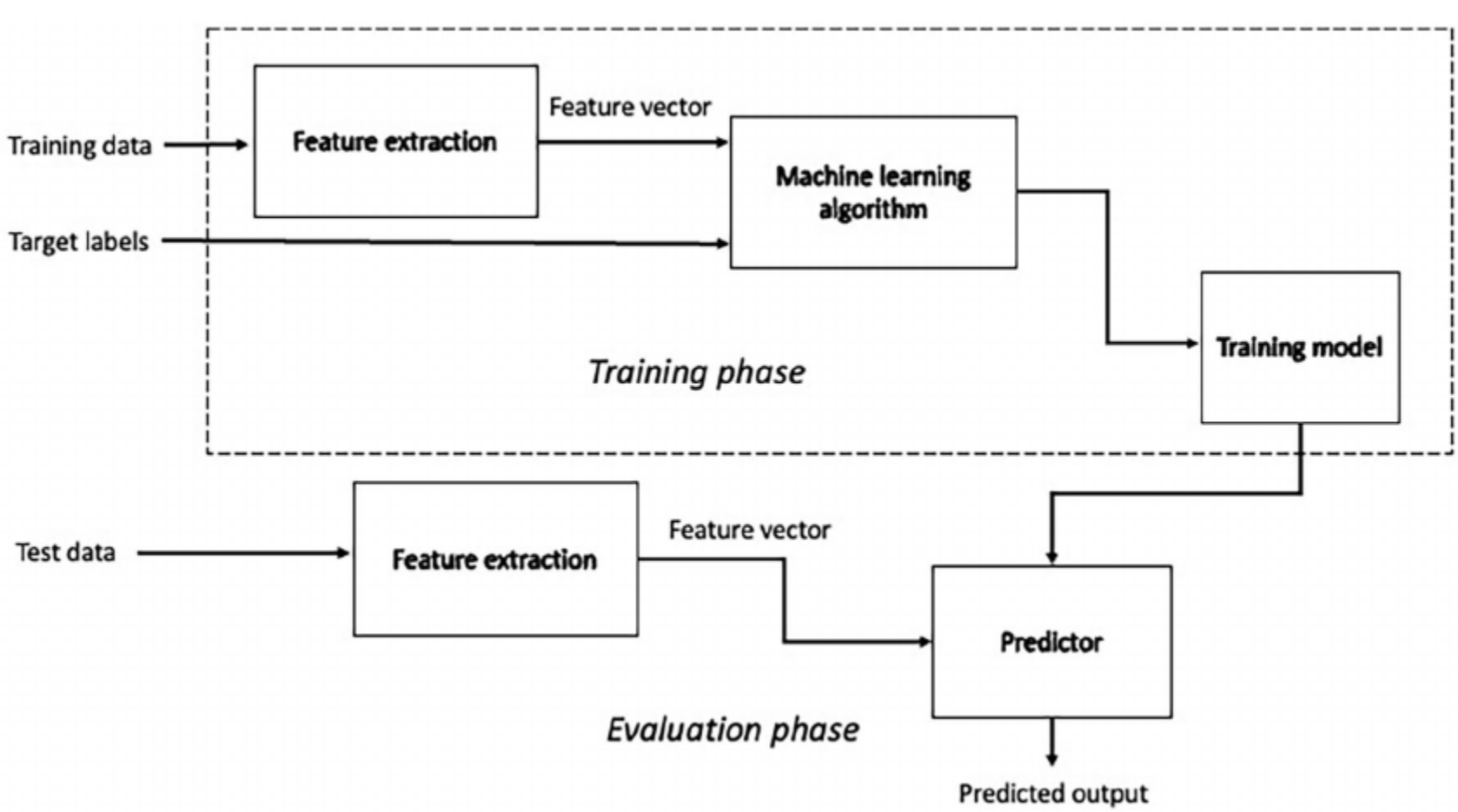
Barcelona Supercomputing Center Barcelona, Spain

European Network on High Performance and Embedded Architecture and Compilation

Pakistan Supercomputing Center

UCERD
Gathering
Intellectuals
www.ucerd.com

Supervised ML



UCERD

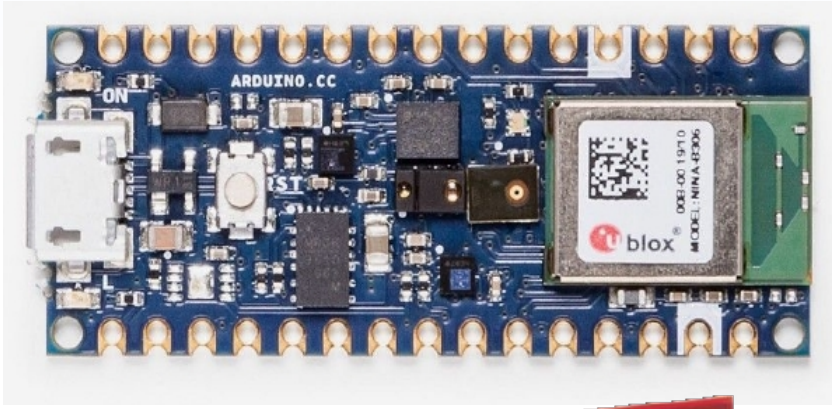
What is TinyML

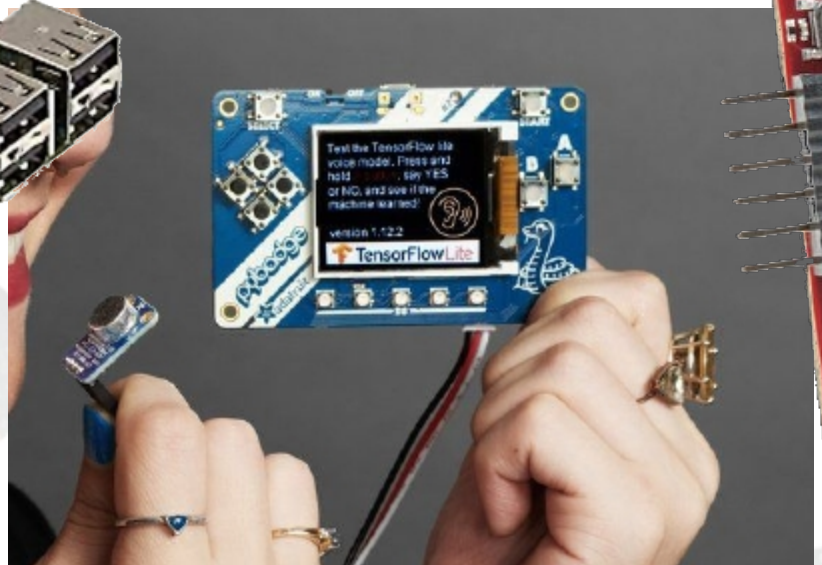
- The usual definition is running machine learning on embedded devices at an average of less than one milliwatt in power.
- This power requirement is important because it allows unattended devices on batteries or energy harvesting.
- Here we'll stretch the definition temporarily to include MCUs that use 10's of mWs, since they're easier to work with and widely available

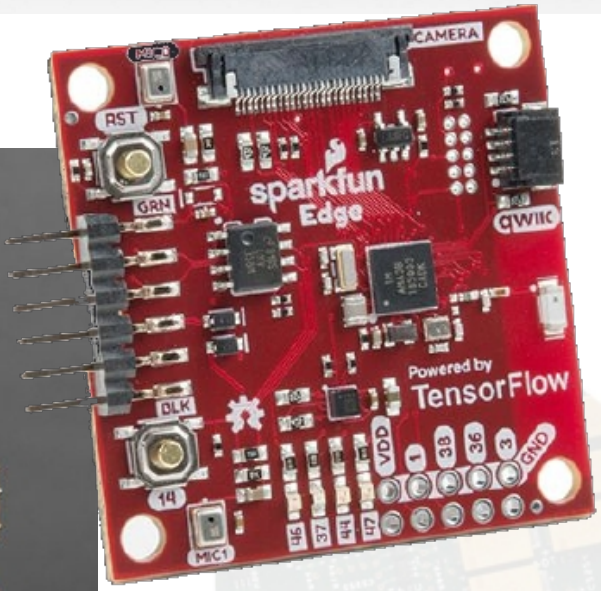
UCERD

Targeted Hardware

Single Board Computer (SBC)
Low Performance
Low Power
Low Cost



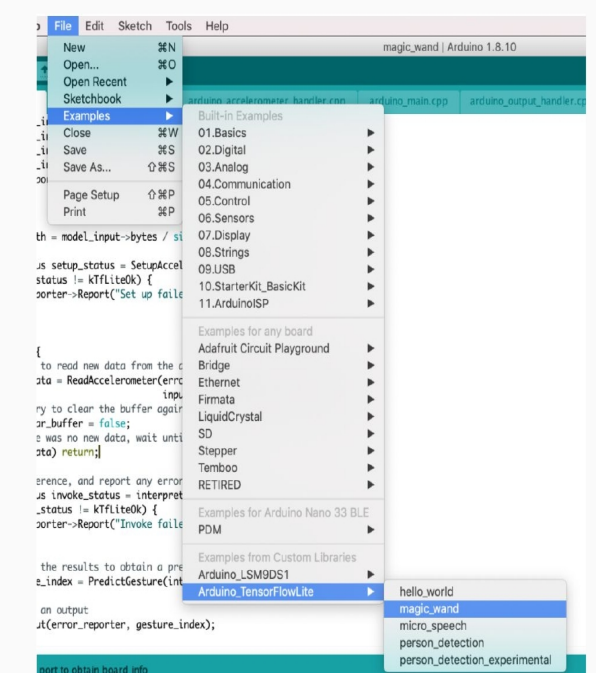




UCERD

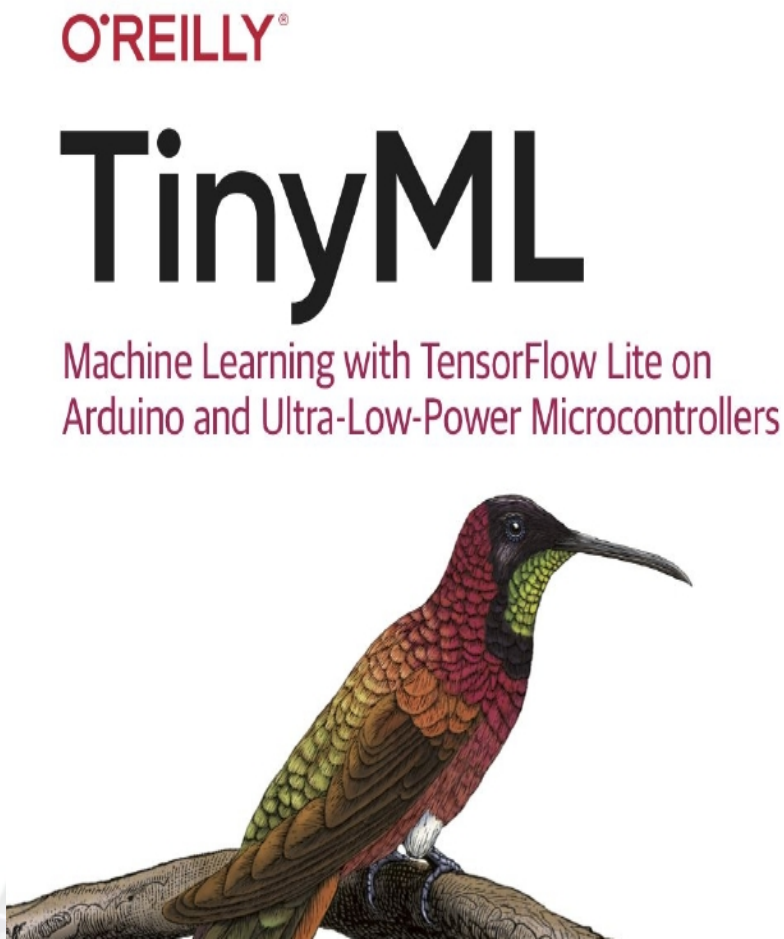
Software Development Framework

- tensorflow.org/lite/microcontrollers!
  - It fits in less than 20KB of binary footprint, and has no operating system, malloc/free or C library dependencies, so it can run on bare metal.
- github.com/uTensor/uTensor
- edgeimpulse.com
- cartesiam.ai
-
-



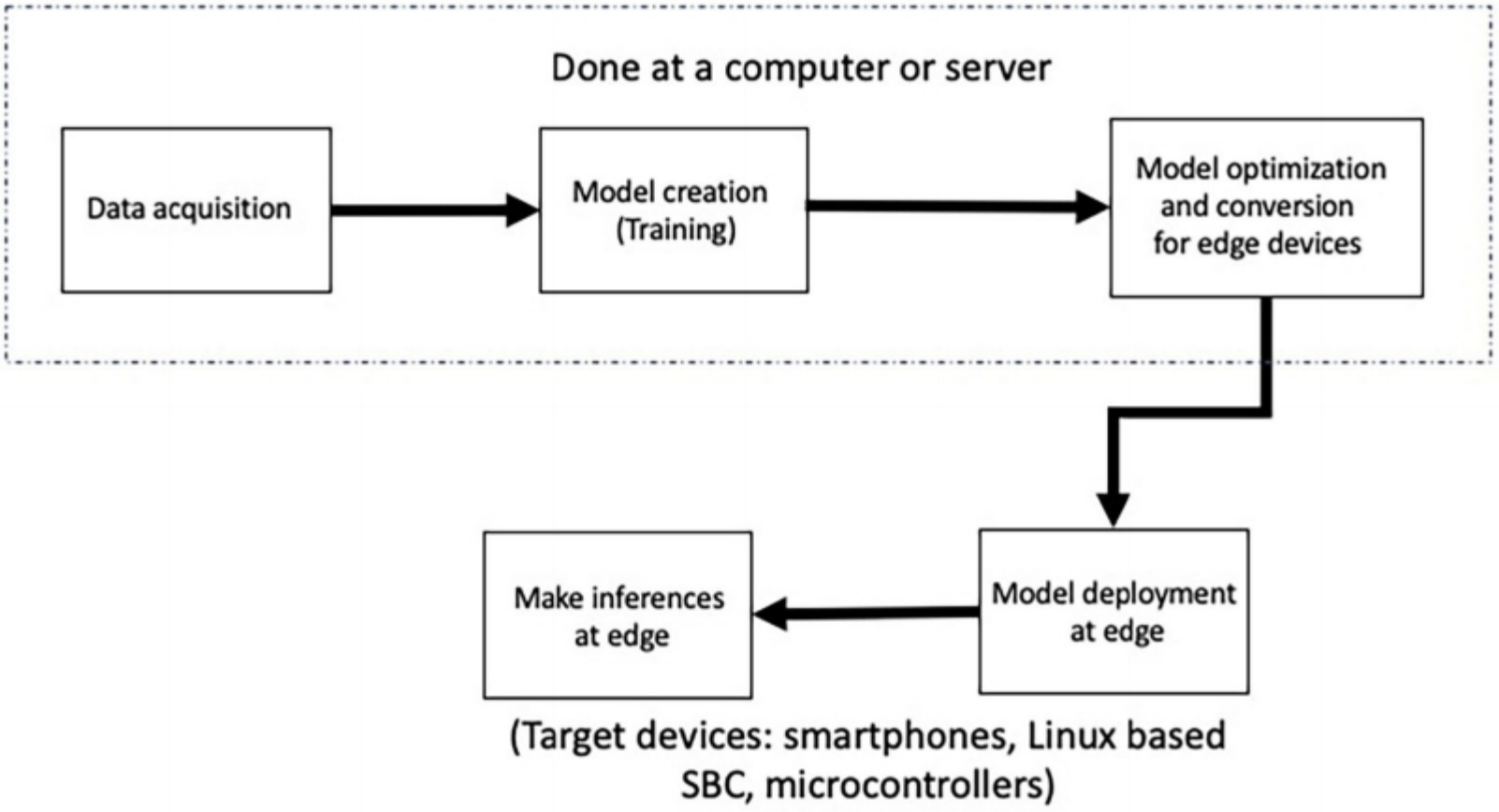
UCERD

Data and Literature



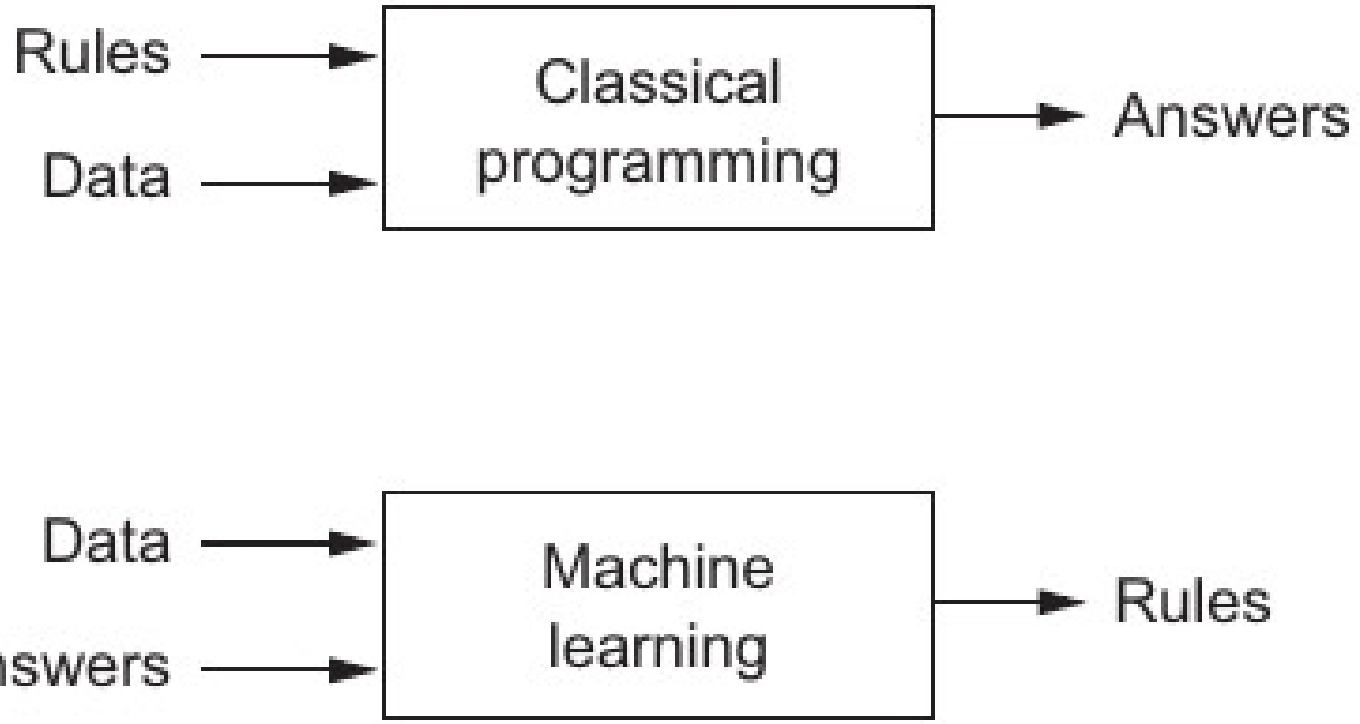
UCERD

TinyML Model Deployment



UCERD

Conventional VS ML Programming



UCERD

Applications

- Keyword Spotting:
- Gesture Recognition:
- Anomaly Detection in Sensor Data:
- Health Monitoring:
- Object Detection in Edge Cameras:
- Industrial IoT (IIoT) Applications:
- Environmental Monitoring:

UCERD

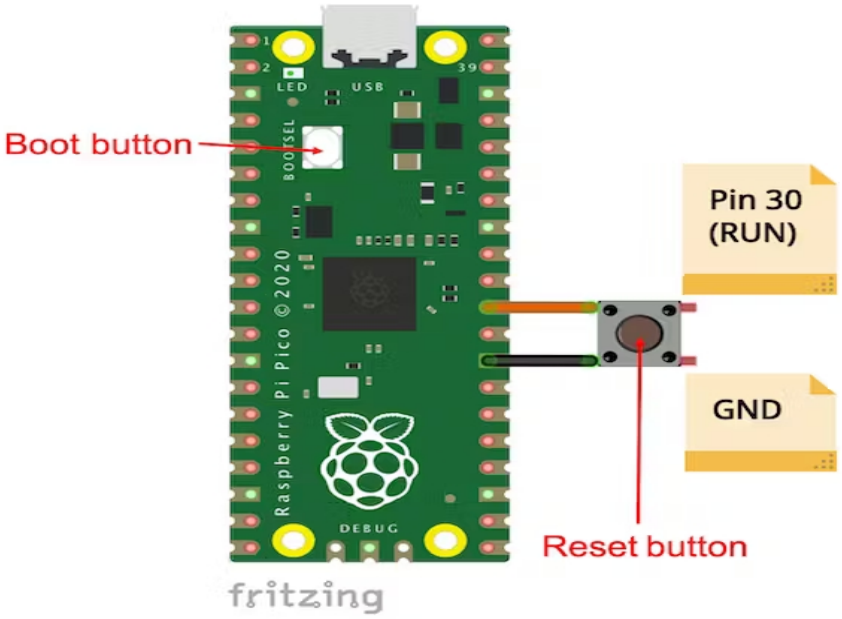
Solvers and Models

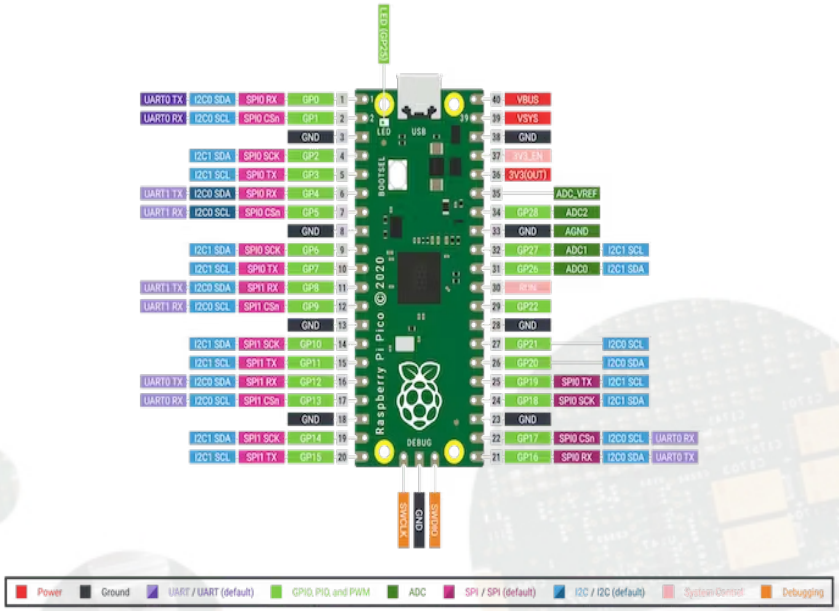
- Utensor
- CMSIS-NN (Cortex Microcontroller Software Interface Standard - Neural Network):
- TensorFlow Lite for Microcontrollers:
- EdgeML:
- MicroML

UCERD

Example

- Micro RaspperyPi





- https://github.com/ShawnHymel/tinyml-example-anomaly-detection

UCERD

First Neural Network

- import tensorflow as tf
- import numpy as np
- from tensorflow import keras
-
- # define a neural network with one neuron
- model = tf.keras.Sequential([keras.layers.Dense(units=1, input_shape=[1])])
-
- # use stochastic gradient descent for optimization and
- # the mean squared error loss function
- model.compile(optimizer='sgd', loss='mean_squared_error')
-
- # define some training data (xs as inputs and ys as outputs)
- xs = np.array([-4.0, -2.0, 2.0, 4.0, 6.0, 8.0], dtype=float)
- ys = np.array([-3.0, -1.0, 1.0, 3.0, 5.0, 7.0], dtype=float)
-
-
- # fit the model to the data (aka train the model)
- model.fit(xs, ys, epochs=300)
-
- print(model.predict([20.0]))

UCERD