The Journal of Open Source Software

emlearn-micropython: Machine Learning and Digital Signal Processing for MicroPython

Jon Nordby [1]

1 Soundsensing, Norway

DOI: 10.21105/joss.09093

Software

- Review
- Repository
- Archive

Editor: Neea Rusch

Reviewers:

- @mikey247
- @swouf
- @Puneetha-Ramachandra

Submitted: 02 August 2025
Published: 20 December 2025

Summary

emlearn-micropython enables sensor data analysis on low-cost microcontrollers. The library provides implementations of a set of algorithms commonly used for sensor data processing. This includes Digital Signal Processing techniques such as Infinite Impulse Response and Fast Fourier Transform, as well as Machine Learning inference such as Random Forest, Nearest Neighbors and Convolutional Neural Networks. Any kind of sensor data can be processed, including audio, images, radar, and accelerometer data.

The library builds on MicroPython, a tiny Python implementation designed for microcontrollers. The modules expose a high-level Python API and are implemented in C code for computational efficiency. This enables engineers, researchers, and makers that are familiar with Python to build efficient embedded systems that perform automatic analysis of sensor data. A range of hardware architectures are supported, including ARM Cortex M, Xtensa/ESP32 and x86-64.

Statement of need

Over the last decade, it has become possible to create low-cost embedded devices that can automatically collect and analyze sensor data. These devices often combine one or more MEMS sensors with a microcontroller, and then using a combination of digital signal processing and machine learning algorithms to extract relevant information from the sensors. The development and utilization of such systems is an active area of research with a wide range of applications in science, industry, and consumer products (Ray, 2022).

Python is among the most commonly used application languages for machine learning and data science. Thanks to the MicroPython (George & contributors, 2014) project, it has become feasible to use Python also on microcontrollers, and this is an attractive proposition for practitioners that are familiar with Python. However, research has identified that running computationally intensive algorithms in Python with MicroPython can be inefficient (Dokic et al., 2020; Ionescu & Enescu, 2020; Plauska et al., 2023). This also limits the effectiveness of tools that generate Python code, such as m2cgen (Zeigerman et al., 2019).

The library ulab (Vörös & contributors, 2019) implements efficient numeric computing facilities for MicroPython, including the core parts of NumPy (Harris et al., 2020), plus some parts of SciPy (Virtanen et al., 2020). However, as of 2025, there are no implementations of machine learning algorithms available in ulab.

OpenMV (Abdelkader et al., 2017) is a project for machine-vision/computer-vision applications using high-end microcontrollers. They have their own distribution of MicroPython, which includes some additional DSP and ML functionality, including image and audio classifiers based on TensorFlow Lite for Microcontrollers (Google LLC and contributors, 2019). However, their solution only officially supports the OpenMV hardware, which limits applicability.

For these reasons, we saw a need to develop a software library for MicroPython with the following properties: 1) supports inference for common machine learning algorithms, 2) is computationally efficient (in terms of execution speed, program space, and RAM usage), 3) run on any hardware supported by MicroPython, 4) can be installed easily.

Our goal is to make research and development in applied machine learning for embedded systems easier for those that prefer developing in Python over conventional C or C++. We believe that this is attractive for many researchers and engineers. It may also be relevant in an educational context.

Within one year of the first release, emlearn-micropython was referenced in a work for on-device learning of decision trees (Karavaev et al., 2024).

Package contents

The emlearn-micropython software package provides a selection of machine learning inference algorithms, along with some functions for digital signal processing. The algorithms have been selected based on what is useful and commonly used in embedded systems for processing sensor data. The implementations are designed to be compatible with established packages, notably scikit-learn (Pedregosa et al., 2011), Keras (Chollet & others, 2015) and SciPy (Virtanen et al., 2020). Table 1 provides a listing of the provided functionality.

The software is distributed as MicroPython native modules (George & contributors, 2020), which can be installed at runtime using the `mip` package manager. The modules provided by emlearn-micropython are independent of each other, and typically a few kilobytes large. This makes it easy to install just what is needed for a particular application, and to fit in the limited program memory provided by the target microcontroller.

Table 1: Overview of modules provided by emlearn-micropython

Module	Description	Corresponds to
emlearn_trees	Decision tree ensembles	sklearn RandomForestClassifier
emlearn_neighbors	Nearest Neighbors	sklearn KNeighborsClassifier
emlearn_cnn	Convolutional Neural Network	keras Model+Conv2D
emlearn_linreg	Linear Regression	sklearn ElasticNet
emlearn_fft	Fast Fourier Transform	scipy.fft.fft
emlearn_iir	Infinite Impulse Response filters	scipy.signal.sosfilt
emlearn_arrayutils	Fast utilities for array.array	N/A

Most of the modules are implemented as wrappers of functionality provided in the emlearn C library (Nordby et al., 2019). However, the emlearn_cnn module is implemented using the TinyMaix library (Wu & contributors, 2022).

Usage example

As an illustrative example of sensor data analysis with emlearn-micropython, we show how data from an accelerometer can be used to recognize human activities. This can, for example, be deployed in a fitness bracelet or smartphone.

Example data is taken from the PAMAP2 Physical Activity Monitoring dataset (Reiss & Stricker, 2012). The tri-axial data stream from the wrist-mounted accelerometer is split into consecutive fixed-length windows. Each window is then processed using Fast Fourier Transform (with `emlearn_fft`), to extract the energy at frequencies characteristic of human activities (typically below 10 Hz). These features are then classified using a Random Forest Classifier

(with `emlearn_trees`). A running median filter is applied to the predictions to smooth out noise. The data at these processing stages is shown in Figure 1.

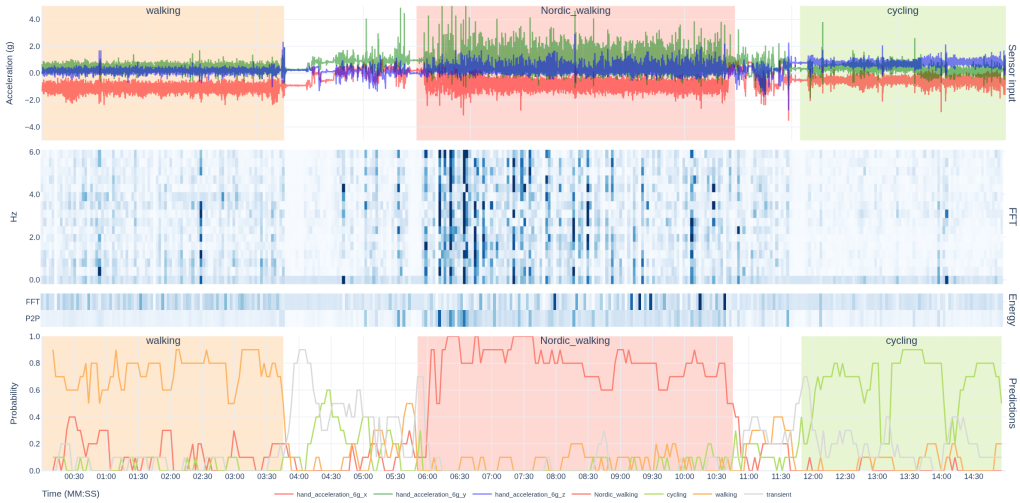

Figure 1: Data pipeline for recognizing physical activities from accelerometer data using emlearn-micropython. Top plot shows input data from the 3-axis accelerometer. Middle plots show extracted features. The bottom plot shows the output probabilities from the classification model. The colored sections indicate the labeled activity (ground-truth).

The emlearn-micropython documentation contains complete example code for Human Activity Recognition, image classification, and more. The documentation can be found at https://emlearn-micropython.readthedocs.io

Acknowledgements

We would like to thank Volodymyr Shymanskyy for his work on improving native module support in MicroPython, Damien P. George and Alessandro Gatti for fixes to native modules, and Jeremy Meyer for user testing of the emlearn_cnn module.

Soundsensing has received financial support in the period from the Research Council of Norway for the EARONEDGE project (project code 337085).

References

Abdelkader, I., El-Sonbaty, Y., & El-Habrouk, M. (2017). Openmv: A Python powered, extensible machine vision camera. *CoRR*, *abs/1711.10464*. http://arxiv.org/abs/1711.10464

Chollet, F., & others. (2015). *Keras*. https://keras.io.

Dokic, K., Radisic, B., & Cobovic, M. (2020). MicroPython or Arduino C for ESP32 - Efficiency for Neural Network Edge Devices. In C. Brito-Loeza, A. Espinosa-Romero, A. Martin-Gonzalez, & A. Safi (Eds.), *Intelligent computing systems* (pp. 33–43). Springer International Publishing. https://doi.org/10.1007/978-3-030-43364-2_4

George, D. P., & contributors. (2014). MicroPython - Python for microcontrollers. In *GitHub repository*. https://github.com/micropython/micropython; GitHub.

George, D. P., & contributors. (2020). MicroPython - native machine code in .mpy files. In *MicroPython documentation*. https://docs.micropython.org/en/latest/develop/natmod.

html; MicroPython.

Google LLC and contributors. (2019). tflite-micro - TensorFlow Lite for Microcontrollers. In *GitHub repository*. https://github.com/tensorflow/tflite-micro; GitHub.

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

Ionescu, V. M., & Enescu, F. M. (2020). Investigating the performance of MicroPython and C on ESP32 and STM32 microcontrollers. *2020 IEEE 26th International Symposium for Design and Technology in Electronic Packaging (SIITME)*, 234–237. https://doi.org/10.1109/SIITME50350.2020.9292199

Karavaev, A., Hejda, J., Kutilek, P., Volf, P., Sokol, M., & Leova, L. (2024). TinyDecisionTreeClassifier: Embedded C++ library for training and applying decision trees on the edge. *SoftwareX*, *27*, 101778. https://doi.org/10.1016/j.softx.2024.101778

Nordby, J., Cooke, M., & Horvath, A. (2019). *emlearn: Machine Learning inference engine for Microcontrollers and Embedded Devices*. https://doi.org/10.5281/zenodo.2589394

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Plauska, I., Liutkevičius, A., & Janavičiūtė, A. (2023). Performance Evaluation of C/C++, MicroPython, Rust and TinyGo Programming Languages on ESP32 Microcontroller. *Electronics*, *12*(1), 143. https://doi.org/10.3390/electronics12010143

Ray, P. P. (2022). A review on TinyML: State-of-the-art and prospects. *Journal of King Saud University-Computer and Information Sciences*, *34*(4), 1595–1623. https://doi.org/10.1016/j.jksuci.2021.11.019

Reiss, A., & Stricker, D. (2012). Introducing a new benchmarked dataset for activity monitoring. *2012 16th International Symposium on Wearable Computers*, 108–109. https://doi.org/10.1109/iswc.2012.13

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental algorithms for scientific computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

Vörös, Z., & contributors. (2019). Ulab - numpy-like fast vector module for micropython, circuitpython, and their derivatives. In *GitHub repository*. https://github.com/v923z/micropython-ulab; GitHub.

Wu, C., & contributors. (2022). TinyMaix - a tiny inference library for microcontrollers. In *GitHub repository*. https://github.com/sipeed/TinyMaix; GitHub.

Zeigerman, I., Titov, N., Yershov, V., & contributors. (2019). m2cgen - transform ML models into a native code with zero dependencies. In *GitHub repository*. https://github.com/BayesWitnesses/m2cgen; GitHub.