University of Bologna

Industry 4.0, 2024
Kimberly Garcia

Assignment 1: Onboard vs. Offboard Data Analytics

Deadline: May 3rd, 2024; 22:00 CET

Overview

In this assignment[1], you will implement and evaluate two variants of deploying a given Artificial Neural Network (ANN) with an embedded device so that it can recognize simple movement gestures (or, in the future, other activities by industrial devices and people). The embedded device has an accelerometer which we will use to supply data to our ANN. The ANN **has already been trained** and the weights and biases of the model are made available to you.[2]

The two possible variants of deploying this ANN model which we will consider in this exercise are:

- *On-board* deployment on the embedded device, such that the sensor data is fed directly to the ANN, and only the *results* need to be transmitted.
- *Off-board* deployment, such that the ANN is hosted on an external computation host (e.g., your laptop), and the embedded device uses its communication interface (Bluetooth Low Energy; BLE) to transmit the *raw data* to the off-board device.

In this exercise, we are concerned with how much energy the embedded device requires in each variant, and how the energy profiles look like. This is an important question when maximizing the lifetime of battery-powered devices since the energy budget of such devices is limited. In the first variant, the embedded device has to locally carry out the computation task of the ANN model, whereas in the second variant it has to send raw data over a wireless link – depending on the device and application, either of these might be energy-hungry tasks. You will implement both versions on a Puck.js[3] device using the Espruino[4] software, and contrast the two approaches.

Across all tasks in this and the other assignments in this course, you are **required to declare** any support that you received from others and, within reasonable bounds,[5] any support tools that you were using while solving the assignment.

[1] This assignment is based on an assignment on *Resource-aware Sensor Data Analytics* part of the Ubiquitous Computing course at the University of St.Gallen, held in Autumn 2023 and 2022, and supported by Ganesh Ramanathan, Andres Gomez, and Simon Mayer.

[2] https://github.com/Interactions-HSG/2023-HS-MCS-UbiComp-Public/tree/main/Assignment1

[3] https://www.puck-js.com/

[4] https://www.espruino.com/

[5] It is not required that you declare that you were using a text-editing software with orthographic correction; it is however required to declare if you were using any non-standard tools such as generative machine learning models such as GPT

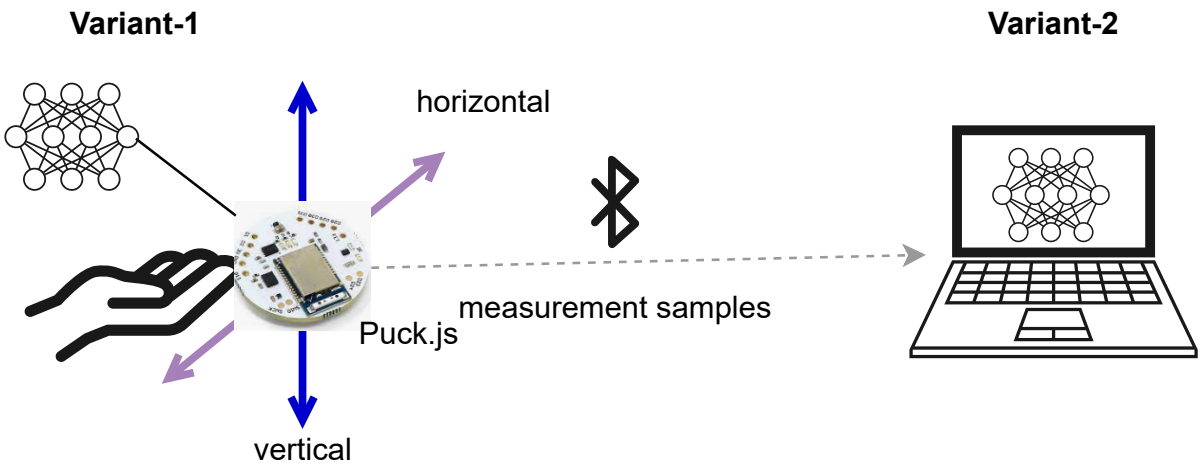


Figure 1: Which one would be more energy efficient – running the ANN on-board (Variant 1) or off-board (Variant 2)?

The Puck.js Device and Espruino Software

You are going to be using an embedded device called Puck.js which can be programmed using the Espruino framework. Espruino allows you to program in JavaScript; an interpreter on board the device executes your program.

On an embedded device like the Puck.js, energy is consumed by different components – for example, the CPU, the radio electronics, the IO peripherals, sensors, memory chips, etc. You can see a breakdown of the power consumption on the official website [6]. When embedded devices are powered using a limited source of energy, say a battery or a photovoltaic (PV) cell, then your program needs to be constructed keeping the energy constraints in mind. For example, you might have to optimize your code such that its CPU time is minimal, or you need to make judicious use of sensors, LEDs and network communication. Battery-powered devices are often seen in many Wireless Sensor Network (WSN) applications, with both static and mobile deployments. Two well-known examples WSN applications include animal tracking[7] and Alpine permafrost monitoring[8], among many others.

The ANN Model

We supply you with an ANN model (see Figure 2) that consists of 60 input nodes which represent 20 consecutive samples of (3-axis) acceleration measurements. The input value must be an integer to the range *-127..127* which corresponds to *-2g..+2g* acceleration measurement. The 20 samples are assumed to be sampled at a frequency of 12.5 Hz. The ANN processes these samples and it output is an array of three signed integers representing the classification likelihood (between -128 and +128). The elements of the array correspond to the probability belonging to each of the following classes: `random`, `up-down`, or `sideways`. For simplicity, you can start by processing one ANN every 20 samples. In most applications, it is common to have multiple overlapping classifications by processing the accelerometer data stream in a sliding window manner, for example, every 2 new samples.

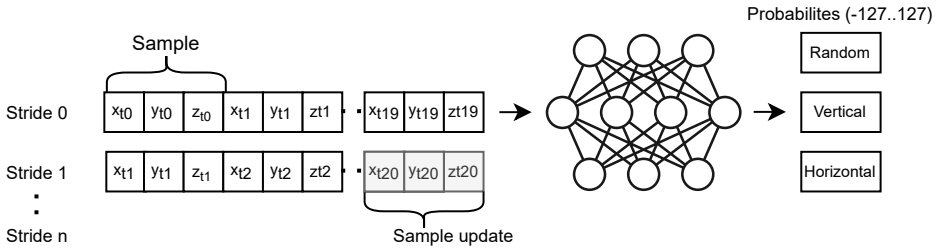
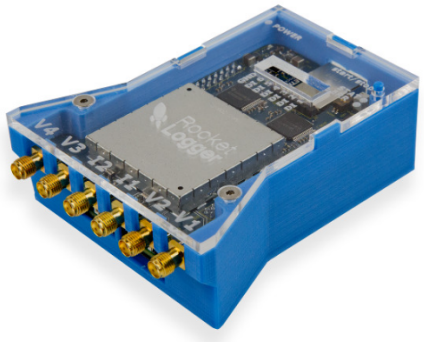


Figure 2: The ANN model needs to be fed with 20 samples of measurement and it outputs probabilities of three gestures. You can improve the prediction by feeding consecutive samples.

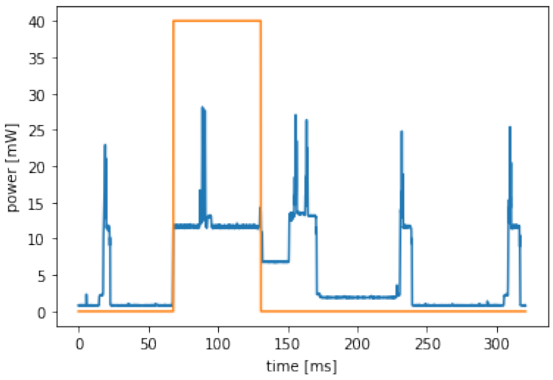
Power Measurement

The power consumption of an embedded device (as it runs a program) is measured using a specialized device called a power logger (see Fig. 3). A power logger monitors the DC current and voltage at the power input port of the embedded device at a high frequency (up to 64 KHz) and logs the data to a storage medium. From this time-series data (i.e., timestamped power measurements; see Fig. 3), one can determine the energy consumed by different parts of the program.

[6] https://www.espruino.com/Puck.js#power-consumption
[7] https://www.princeton.edu/~mrm/sensys04.pdf
[8] https://cn.dmi.unibas.ch/people/iat/Papers/EmNets_IEEE_LaTeX_A4_v9.pdf



(a) The RocketLogger[9] Device.



(b) Sample RocketLogger Trace with power (in blue) and task indication (in orange).

Figure 3: The RocketLogger is a specialized power measurement device for monitoring energy consumption in embedded electronics. The example trace shows a Puck.JS device's power consumption, as well as GPIO used to indicate when a specific task is being executed.

Hand-in and Grading

The evaluation of this assignment is **binary**. The requirements for passing the assignment are (a) submission of running code for Tasks 2 and 3 and written answers to the questions from Tasks 1, 4 and 5. Please include the student names in the submission's filename.

Your Tasks

(1) Preliminaries

To get you started with some basics of energy-aware programming, try to answer the following questions:

a) Given the power values in the Espruino documentation, estimate the average power consumption when the Puck.js device has an active BLE connection, the accelerometer enabled at 12.5 Hz, and takes 5 milliseconds to execute the ANN (at 100% CPU utilization) every 1.6 seconds.

b) Consider the battery characteristics of a CR2032 cell. Estimate the battery life in terms of how many times the gesture classification can be executed. You can assume each gesture classification consumes an average of 2.5 mW and takes 1.6 seconds to complete.

c) Assuming you have a discrete-time power trace and digital signal that indicates when a task is running, how can you estimate the energy consumption of that task?

(2) On-Board Data Analysis

a) Clone the Espruino repository[10], which already contains the ANN model implemented in C. Compile the Espruino code and upload the new binary to the Puck.js device using Nordic's NRF Connect or DFU Update apps. You can use either the iOS version[11] or the Android version[12].

b) Write a program for the Puck.js that supplies accelerometer data to the onboard ANN model. The program needs to initialize the accelerometer with the correct settings and then wait

[10] https://github.com/Interactions-HSG/UbiComp-Espruino
[11] https://apps.apple.com/us/app/nrf-connect-for-mobile/id1054362403
[12] https://play.google.com/store/apps/details?id=no.nordicsemi.android.mcp

for updates. After 20 samples are obtained, the ANN can be invoked and the classification obtained. In the simplest version, you can wait for 20 new samples before calling the ANN model again.

c) (Optional) Implement a sliding window mechanism that performs the classification after every sample (i.e., stride length = 1).

(3) Off-Board Data Analysis

Write another program for the Espruino that transmits accelerometer data via WebBluetooth[13], where we recommend using the BLE UART connection for the data transmission. Create a simple browser-based application that connects to the WebBluetooth port and receives the updates. Collect these updates and feed them to the ANN model. The core functions are found in the files with "infxl" in their names. These files are found in the following Espruino folder[14].

(4) Analyze Power Traces

For this task, use the **pre-recorded measurement traces** of our sample solutions and the sample post-processing script[15]. These traces were recorded with a RocketLogger device using one current, one voltage and one digital channel. The first two can be used to determine the device's power consumption. The digital signal is used to determine when the device is performing a specific task - it's low when inactive and high when executing the task. In the Onboard version, the task is the ANN model execution. In the Offboard version, the task is the console printing the comma-separated values via UART service.

a) Analyze the power trace of the *on-board version* using the provided RocketLogger file [16] Calculate the average power consumption over the entire measurement, as well as the energy consumption of the Puck.js while it is performing the gesture classification. The Digital Input 1 (D1) signal indicates when this task is being executed and can be used for the energy calculation.

b) Analyze the power trace of the *off-board version* using the provided RocketLogger file. Calculate average power consumption over the entire measurement, as well as the energy consumption of the Puck.js while it is printing data to the UART service. The Digital Input 1 (D1) signal indicates when this task is being executed and can be used for the energy calculation.

c) (Optional). Introduce a sleep state to your program. In this way, the Puck.js device consumes very little energy until a button is pressed. When this happens, the device wakes up and reads accelerometer data for 10 seconds, then goes back to the sleep state.

(5) Compare and Contrast

Compare the on-board and off-board versions in terms of energy and explain the similarities and differences you see. Explain what has the biggest overhead in terms of energy, and describe any additional power optimization you would make to maximize the battery lifetime. Based on this, draw conclusions about the trade-offs between the two approaches while focusing on energy consumption but also going beyond (e.g., latency).

[13] https://www.espruino.com/ide/

[14] https://github.com/Interactions-HSG/UbiComp-Espruino/tree/master/libs/misc

[15] https://github.com/Interactions-HSG/2023-HS-MCS-UbiComp-Public/tree/main/Assignment1/postprocessing_measurements.ipynb

[16] https://github.com/Interactions-HSG/2023-HS-MCS-UbiComp-Public/tree/main/Assignment1/data