

ECE 361E: Machine Learning and Data Analytics for Edge AI

HW2 - In depth explanations

IMPORTANT! Do not use VSCode/PyCharm or any other IDE to connect to the edge devices in this class. They consume a lot of memory and impact everything that runs on the devices. Only use a plain terminal to connect to the devices.

The Odroid MC1 (Fig. R1) uses Exynos 5422, a heterogeneous multi-processor system-on-a-chip (MPSoC) that consists of two clusters of ARM cores (organized into a big.LITTLE architecture) and a small GPU core¹. The “big” cluster consists of four A15 cores designed for high performance, while the “LITTLE” cluster implements four A7 cores intended for maximum power efficiency.

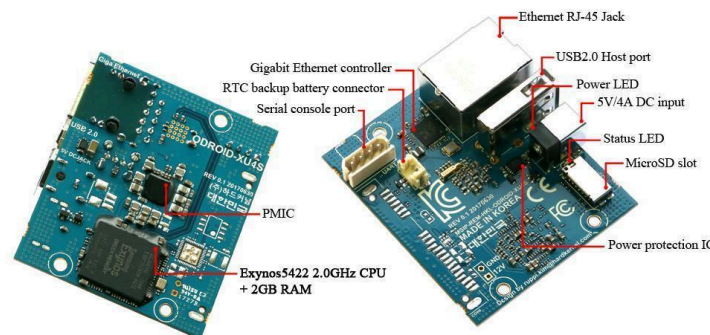


Figure R1. View of the Odroid MC1 edge device

IMPORTANT! All plots in all homeworks must have: a title, labels on both axes, a grid and a legend with corresponding labels for each line in the plot.

Use *matplotlib.pyplot* to draw figures and save your figures as images (e.g., PNG/JPEG). Save all data for the plots in a CSV file so there is no need to re-train in case of an error or system malfunctioning. For the entire HW2 use a sampling period of **200 milliseconds [ms]**.

IMPORTANT! As a general rule for measuring power and temperature on edge devices, *leave at least 2 minutes* between running the benchmarks to allow the MC1 thermals and loads to reach the idle steady-state. Of note, avoid using print statements in the measurement file since they may cause a small delay in the measurements/inference code that compounds over time.

For better readability, for all numbers that are over 999 in value, put comma signs, e.g., 1,800,903 instead of 1800903. In the same context, use 0.XXXX for decimal values (i.e., do *not* use more than 4 decimal digits) for any results written in tables, e.g., use 0.9384 instead of 0.938411291111. For time measurement, use the **time.time()** function.

IMPORTANT! Everything you need is already provided to you on the devices. You are not allowed to install anything on the Odroid devices. By extension, you are not allowed to make any modification through software to the edge devices unless we mention it explicitly in the homework. When solving the homework, create your own directory and do *not* modify the *HW2_files*, *HW3_files* or *HW4_files* folders and their contents.

¹ More details on Exynos 5422 available [here](#).

Problem 1: Cyber-Physical Systems and Benchmarks

In this problem, you will explore the cyber-physical aspects of MC1 using two types of workloads: a single-threaded throughput benchmark (*TPBench*) and two multi-threaded benchmarks (*blackscholes* and *bodytrack*). *TPBench* is an application that loads each individual core to 100% utilization with four types of instructions: floating-point multiply-accumulate, integer multiply-accumulate, floating point addition, and integer addition. The *blackscholes* and *bodytrack* (from the *PARSEC*² suite) are two multithreaded computational benchmarks. The *blackscholes* benchmark is a computational finance options pricing benchmark; *bodytrack* is an algorithm for tracking an unmarked person across several images.

Question 1

To answer this question, you need to check **Appendix A2.3** to learn how to run the benchmarks. To send files to and from the device see **Appendix A2.4**.

Hint: Start power and temperature logging before running the benchmark and stop it right after. To make it easier, either open multiple terminal windows or use **tmux** as described in **Appendix A2.7**.

Use the *sysfs* paths provided in *sysfs_paths.py* to access resource voltage and frequency settings and thermal paths. For power measurements, we use the Smart Power 2 device (details are given in **Appendices A2.5 and A2.6**). For core usage, we recommend the *cpu_percent()* function from the *psutil* library. In the *cpu_percent()* function, use the sampling rate as the *interval* parameter, and use *percpu=True* to obtain the core usage for each core of the CPU. Check in **Table R1** the thermal zones corresponding to the big cores. Note that the LITTLE cores do not have thermal sensors.

Table R1. Core types, numbers and thermal zones

Core type	LITTLE 1	LITTLE 2	LITTLE 3	LITTLE 4	big 1	big 2	big 3	big 4
Core number	0	1	2	3	4	5	6	7
Thermal zone number	-	-	-	-	0	3	2	1

Question 3

The *run time* is the total time needed for a benchmark to be fully executed (i.e., from start to finish). The *average power* is the mean of system power consumption needed to execute a particular benchmark. The *average maximum temperature* is the mean of the *max big temp* for a benchmark. The *max temperature* is the peak value of *max big temp* for a benchmark, e.g., *max big temp* = $\max \{\text{big core 4 temp, big core 5 temp, big core 6 temp, big core 7 temp}\}$. For the *max big temp* plot, you only need one curve, i.e., the maximum out of all big core temperatures (no need to plot the temperature for each big core).

The *energy* used to run a benchmark is computed as the sum of power consumed by the system **while running a benchmark** multiplied by the power sampling period. This means you need to capture the start and end time of when the benchmark ran and only compute the energy used during that period.

Problem 2: System Power Prediction

In this problem, you will build a system power predictor model using *scikitlearn*. The provided datasets contain on each row the state of the system (i.e., power consumption, temperature). Each row represents one time step [s] (seconds) and the time difference between two consecutive rows is 1 second.

² Bienia, C. Sanjeev K., Jaswinder P. S., and Kai L. "The PARSEC benchmark suite: Characterization and architectural implications." In Proc. International Conference on Parallel Architectures and Compilation Techniques, pp. 72-81. 2008.

Question 1

Full points will be awarded only if the classifier provides a **test accuracy greater than 98% for both *blacksholes* and *bodytrack* benchmarks**. You can modify any hyperparameters of the models to achieve the required accuracy.

The confusion matrix used should be from [sklearn.metrics.confusion_matrix](#) and you should *plot* it, not display a mathematical matrix with square brackets. Feel free to use [this Scikit-learn function](#).

In the *csv* file: *total_watts* is the total power consumption, *w_big* is the power consumption of the big cluster, *w_little* is the power consumption of the little cluster, *w_gpu* is the power consumption of the GPU and all are measured in [W].

Use label '0' for the 'cluster idle' and label '1' for 'cluster active'. When computing the average precision, average recall and average F1-scores, use the corresponding sklearn functions [precision_score](#), [recall_score](#) and [f1_score](#) with the parameter *average='macro'*.

Hint: Try various combinations of model fine-tuning and dataset normalization to achieve higher accuracy. For example, you can divide the temperature by 100 and also divide the frequency of the big cluster by 1,000,000,000. If needed, you can also try to use scalers ([StandardScaler](#), [MinMaxScaler](#)) from *scikit-learn* to rescale the data and see what methods work best to provide the best model accuracy.

Question 2

R^2 is called the coefficient of determination. It determines the proportion of variance in the output that is explained by the input features. This gives us a good idea of whether we need better input features or not. The scikit-learn implementation of R^2 is available [here](#).

Question 3

For each data sample in *training_dataset.csv*, add a new column with the computed value of the $V_{dd}^2 f$ feature. For each data sample, compute the new feature using the frequency from the dataset and the appropriate V_{dd} from **Table 3**. After any manual scaling done as in **Problem 2, Question 1**, make sure you use a [StandardScaler](#) for all the features such that the feature importances will be relevant.

Hint: The most important features are denoted by *the largest positive values* of their corresponding parameters.

Problem 3: System Temperature Prediction

The last part of the homework focuses on temperature prediction on our edge device. The training and test datasets are the same as in **Problem 2**.

Question 1

To solve this question, for each big core, use an MLPRegressor model with the following parameters: *hidden_layer_sizes* = (128, 64, 32), *activation* = 'relu', *random_state* = 42. Do **not** change the regularization term, i.e., leave the alpha parameter on its default value. You need to train four models in total, one for each big core temperature. Do **not** normalize the temperature features (i.e., *temp4*, *temp5*, *temp6*, *temp7* and *temp_gpu*); leave them unchanged.

Note: The test MSE should be for all core temperatures **lower than 1.5**, for both benchmarks.

Hint: You can divide the frequency of the big cluster by 1,000,000,000 to have a better scale for the features used in regression.