Spring 2023

ECE 361E: Machine Learning and Data Analytics for Edge AI
HW2 - In depth explanations

The Odroid MC1 (Fig. 1) 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 core[1]. 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.

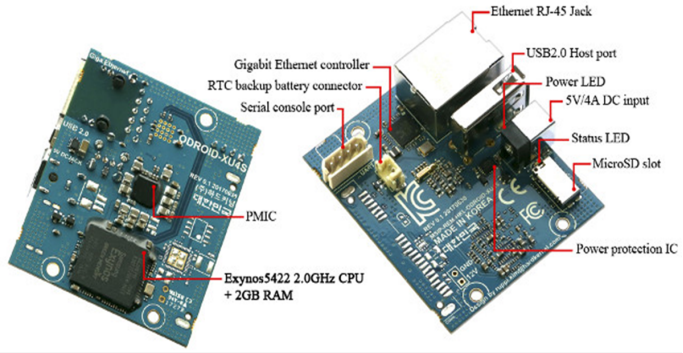

Figure 1. View of the Odroid MC1 edge device

Each plot in this homework 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 PNG files. 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).

For better readability, for all numbers that are over 999 in value, put comma signs, e.g.,1,800,903 instead of 1800903.

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 A2.1** for instructions on how to connect to the device. Check **Appendix A2.3** for how to run the benchmarks. To send files to and from the device see **Appendix A2.4**.

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

[1] More details on **Exynos 5422** available here.
[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.

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.

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. Leave enough time (e.g., at least one or two minutes) between the completion of one benchmark before you start the next benchmark so the MC1 thermals and loads can reach the idle steady-state.

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).

Question 1

Full points will be awarded only if the classifier provides test accuracy greater than 98% for both the benchmarks. You can modify any hyperparameters of the models to achieve the required 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***. Make sure you use a scaler for all the features such that the feature importances will be relevant.

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 apply any regularization term. You need to train four models in total, one for each big core temperature.