Spring 2026

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

HW2 Assigned: Feb 3 DUE: Feb 12 (CST 11:59:59pm)

Work in teams of two students. At the end of the PDF file, insert a paragraph where you describe each member's contribution and two valuable things you learned from this homework. Only one submission per group is required.

Introduction

This assignment is meant to deepen your understanding of Cyber-Physical Systems (CPS) and their use to run ML workloads efficiently. Specifically, you will be working with an Odroid MC1 edge device. You will also use the Scikit-learn machine learning framework to train and evaluate machine learning models. By working on this assignment, you will learn:

- How to modify the frequency of the cores to lower their energy consumption, how to measure and predict thermal and power consumption;
- How to visualize and interpret various metrics and parameters of interest to understand the impact of core frequency, power and thermal dissipation on the overall performance of edge devices;
- How to run simple benchmarks while considering both the "cyber" (i.e., performance) and the "physical" (i.e., power/thermal) components of the Odroid MC1 edge device.
- How to use Scikit-learn to train classification/regression models on real data from edge devices.

Problem 1 [30p]: Cyber-Physical Systems and Benchmarks

Question 1: [12p] Connect to your designated Odroid MC1 using the steps in **Appendix A2.1**. Change the frequency of the LITTLE cluster to 0.2GHz and the big cluster to 2GHz and run ***TPBench*** only on the first big core, i.e., core 4 (see **Appendix A2.2**). Draw as a function of time [s] three plots: a plot for the system (total) power consumption [W], a plot for big cores temperature [°C] with one curve drawn for each big core, and a plot for the big cores usage [% utilization] with one curve for each big core.

Question 2: [3p] How many phases of benchmark execution can you identify based on the temperature variation in the plot? A phase is a significant increase in the temperature over an extended period of time.

Question 3: [15p] Run the ***blacksholes*** and ***bodytrack*** benchmarks only on all the big cores (see **Appendix A2.2** and **A2.3**) with a frequency value of 2GHz, while keeping the LITTLE cluster at 0.2GHz. For the ***blackscholes*** benchmark set the number of threads to 4 to use all 4 big cores (see **Appendix A2.3)**. Likewise, for the ***bodytrack*** benchmark set the number of threads to 4 to use all 4 big cores. For *each benchmark,* draw two plots as a function of time [s] (i.e., 4 plots total, two for each benchmark): one plot for the *system power* [W] and one plot for the *max big temp* [°C]. Complete ***Table 1***:

Table 1

Benchmark	Run time [s]	Avg. power [W]	Avg. max temp [°C]	Max temp [°C]	Energy [J]
blacksholes					
bodytrack					

Problem 2 [40p]: System Power Prediction

Question 1: [20p] Use an SVM model from Scikit-learn to classify the states of the big cluster, namely *"cluster active"* and *"cluster idle"*. An *active state* of the big cluster corresponds to a power consumption larger than 1W, while an *idle state* corresponds to a power consumption less than 1W. Use all the input features for classification, except the big cluster power consumption (i.e., **w_big**). Train the model on your computer[1] on the ***training_dataset.csv*** dataset and then test the models on ***testing_blacksholes.csv*** and ***testing_bodytrack.csv*** datasets. Use the thermal, power, core usage, and frequency data provided in the ***training_dataset.csv*** to train the models.

Visualize the *confusion_matrix* (feel free to use this Scikit-learn function) for the two testing datasets. Compute the following performance metrics: accuracy, average precision, average recall and average F1-score. Based on all these performance metrics and the confusion matrix, explain the performance of your classifier. Complete ***Table 2***:

Table 2

Benchmarks	Accuracy [%]	Avg. Precision	Avg. Recall	Avg. F1-Score
blackscholes				
bodytrack				

Question 2: [5p] Use a Linear regression model to predict the actual power values of the big cluster (i.e., *w_big*) based on the current state of the system (i.e., the provided features). Do *not* use any of the power features (i.e., ***total_watts, w_little, w_gpu*** and ***w_mem***) as input features for the model. Full points will be given if you design a regressor that can obtain a test MSE value less than 0.15.

Draw two plots, one for ***blacksholes*** and one for ***bodytrack***. For each plot, draw two curves: one for the true power values and one for the predicted power of the big cluster over time [s]. Complete ***Table 3:***

Table 3

Dataset	***training***	***blacksholes***	***bodytrack***
R^2			
MSE			

Question 3: [15p] Considering the dynamic power formula given in **Lecture 8**, use the term $V_{dd}^2 f$ as a feature in the training set, where f is the frequency of the big cluster and the corresponding V_{dd} is obtained from ***Table 4***. Do not use any of the power features (i.e., **total_watts**, **w_little**, **w_gpu** and **w_mem**) as input features for the model.

Train the linear regression model on ***training_dataset.csv*** again and use the feature importance function to plot all feature importances and mention which are the top 3 positive features that contribute to the performance of the regressor. What do you observe? Explain.

Table 4

V_{dd} **[V]**	0.975	1	1.1375	1.362
f **[GHz]**	0.9	1	1.5	2

[1] We will not use GPUs for this homework.

Problem 3 [30p+10Bp]: System Temperature Prediction

Question 1: [25p] Train four *MLPRegressor* models, one for each of the big cores, to predict the *temperature* values for the next time step based on the features of the current time step. Do *not* use any of the power features (i.e., *total_watts, w_big, w_little, w_gpu* and *w_mem*) as input features for the model. You may use the *temperature* features from the *current* time step to predict the *temperature* for the *next* time step.

Evaluate the performance of your model using the ***testing_blacksholes.csv*** and ***testing_bodytrack.csv*** test datasets. Draw two plots, one for ***blacksholes*** and one for ***bodytrack***. For each plot, draw two curves: one for the true temperature and one for the predicted temperature values of the big core 4 over time [s]. Complete ***Table 5***:

Table 5

Dataset	Test MSE (Core 4)	Test MSE (Core 5)	Test MSE (Core 6)	Test MSE (Core 7)
blacksholes				
bodytrack				

Question 2: [5p] What other techniques can be used to further improve the performance of your regressor? List at least two such techniques.

BONUS Question 3: [10Bp] Using an Odroid MC1, we already implemented an on-demand governor algorithm (pseudo-code given below in ***Algorithm 1***). ***Table 6*** shows the results obtained for both ***bodytrack*** and ***blackscholes*** benchmarks when executed with the on-demand governor active and a temperature threshold of 60°C.

What are the possible "cyber-physical" trade-offs when having such a governor running? Discuss such trade-offs by comparing the runtime, average power consumption, thermal limits, and energy consumption of each benchmark with what you already obtained in ***Table 1***.

Algorithm 1 On-Demand Governor for Dynamic CPU Frequency Scaling Based on Temperature

$F = \{f_0, f_1, ..., f_{N-1}\}$ ▷ List of N available frequencies f_i [GHz], with $f_0 < f_1 < ... < f_{N-1}$
Initialize $i = 0$, with the CPU frequency to f_i, the lowest available frequency
Define the temperature threshold $\tau = 60°C$
while True **do**
  Measure $T = max\{t_k\}$, the maximum CPU temperature t [$°C$] of all cores k
  if $T > \tau$ **then**
    Set $i = \max(0, i-1)$ ▷ Decrease CPU frequency
  else if $T < \tau$ **then**
    Set $i = \min(i+1, N-1)$ ▷ Increase CPU frequency
  end if
  Set CPU frequency to f_i
  Wait for the next sampling interval
end while

Table 6

Benchmark	Runtime [s]	Avg. power [W]	Avg max temp [°C]	Max temp [°C]	Energy [J]
blacksholes	148.99	6.23	51.71	62	928.23
bodytrack	138.20	7.24	51.32	63	1,000.12

Submission Instructions

Include your solutions to all the problems into a single zip file named <**Team#**>**.zip**. The zip file should contain:

1. A single PDF file containing all your results and discussions.
2. For **Problem 1** submit your **.py** files named suggestively. For **Problem 2** and **Problem 3**, submit **two distinct Jupyter Notebooks** (named ***p2.ipynb*** and ***p3.ipynb,*** respectively) containing the outcomes of executing your code (e.g., training and test phases, tensor shapes for training and test, features used to train your models, and training and test accuracies).
3. A *readme.txt* file describing all your items in the zip file.

Good luck!