The University of Texas at Austin
Chandra Department of Electrical and Computer Engineering
Cockrell School of Engineering

Spring 2023

ECE 361E: Machine Learning and Data Analytics for Edge AI
HW4 Assigned: Feb 28 DUE: Mar 9 (11:59:59pm CST)

Work in groups 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

In this homework, you will design and deploy popular deep learning models on different edge devices (i.e., Odroid MC1 and RaspberryPi 3B+) using TensorFlow and TensorFlow Lite (TFLite). You will train different networks on the CIFAR10 dataset and compare their accuracy, latency, and energy consumption for inference. By working on this assignment, you will be able to:

- Understand the process of designing TensorFlow models (e.g., training a model, converting it to TensorFlow Lite);
- Understand the process of deploying a deep neural network with TensorFlow Lite.

Of note, neither Tensorflow, nor PyTorch are bound to any deployment framework. For example, we can also deploy a PyTorch model with TFLite or deploy a TensorFlow model with ONNX.

Problem 1 [40p]: TensorFlow Evaluation

Question 1: [15p] Starting from the given TensorFlow code of VGG11, write the code for VGG16 and place it in the ***models/*** folder. For the TensorFlow models, we use the naming convention of adding the suffix ***_tf*** to the model name (e.g., ***vgg11_tf.py*** and ***vgg16_tf.py***).

Question 2: [15p] Train VGG11 and VGG16 using ***main_tf.py*** on Maverick2, in parallel. Follow the ***TODO*** parts from the ***main_tf.py*** code to make it work properly and then add the necessary code needed to complete ***Table 1***.

Table 1

Model	Training accuracy [%]	Testing accuracy [%]	Total time for training [s]	Number of trainable parameters
VGG11				
VGG16				

Question 3: [10p] Draw (on the same plot) the testing accuracy values of VGG11 and VGG16 as a function of the number of training epochs.

Problem 2 [40p]: Deployment on the Edge using TensorFlow Lite

Question 1: [10p] Create a new file named ***convert_tflite.py*** where you define a function to convert a pre-trained TensorFlow model to TFLite.

Question 2: [10p] Use the ***deploy_tflite.py*** file from ***HW4_files*** to deploy the TFLite version for each VGG11 and VGG16 models onto the RaspberryPi 3B+ and Odroid MC1 devices. Complete the ***TODO***

parts from the ***deploy_tflite.py*** file. Perform inference on the *entire* test dataset available in the ***HW4_files/test_deployment*** folder, record the total time required for inference [s] and complete ***Table 2***.

Table 2

	Total inference time [s]		RAM memory [MB]		Accuracy [%]	
	MC1	RaspberryPi	MC1	RaspberryPi	MC1	RaspberryPi
VGG11						
VGG16						

Question 3: [20p] For VGG11, draw on the same plot, the variation of power consumption for each device (i.e., one curve for Raspberry Pi and one MC1). In another plot, draw the temperature variation of the CPU for each device (i.e., one curve for Raspberry Pi and another curve for MC1). Complete ***Table 3***:

Table 3

Model	MC1 total energy consumption [J]	RaspberryPi total energy consumption [J]
VGG11		
VGG16		

Problem 3 [20p+10Bp]: ONNX vs TensorFlow Lite

Question 1: [5p] Use the given MobileNet-v1 TensorFlow implementation to train it on the CIFAR10 dataset using the ***main_tf.py*** file on Maverick2. Extend ***Table 1*** in Problem 1 with a new row containing the new results obtained for MobileNet-v1. Save the trained model. Use the ADAM optimizer with a **learning rate = 0.001** (different from the VGG models) to train the MobileNet-v1.

Question 2: [15p] Use ***convert_tflite.py*** (from **Problem 2, Question 1**) to convert the MobileNet-v1 model in TFLite and deploy it using the ***deploy_tflite.py*** file on both Odroid MC1 and Raspberry Pi 3B+ devices. Extend ***Table 2*** and ***Table 3*** from ***Problem 2*** with new rows containing the new results.

BONUS Question 3: [10p] While using the ONNX (from ***HW3***) and TensorFlow Lite on both RaspberryPi and Odroid devices, which deployment framework tends to be more efficient? Explain the possible tradeoffs of these two deployment frameworks in terms of inference time, RAM consumption, and energy consumption. Which deployment framework do you prefer? Explain your choice.

Submission Instructions

Include your solutions into a single zip file named <**Group#**>**.zip**. The zip file should contain:

1. A single PDF file containing all your results and discussions.
2. A *readme.txt* file explaining all your items in the zip file.
3. Your code files, named suggestively (e.g., **p1_q1.py** for **Problem 1 Question 1** code).

Good luck!