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
HW3 - In depth explanations

All the plots in this homework need to have: a title, labels on both X,Y 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.

In this homework you should first *train your models exclusively* using the GPUs of Maverick2, and then perform inference on the edge devices. **Use only the GTX nodes on Maverick2.** You do ***not*** need to change the loss, the optimizer, the batch size, the number of training epochs or the learning rate when training. For time measurement, use the **time.time()** function.

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: PyTorch Evaluation

In this problem, you will compare and contrast the training accuracy, memory consumption and training time of a medium model (VGG11) and a large model (VGG16) using the CIFAR10 dataset.

Question 1

Compared to VGG for ImageNet dataset presented in ***Lecture 5***, there are two differences in its implementation for CIFAR10 dataset: instead of 4096 neurons per fully connected layer, you now have 512, and instead of 1000 classes you have now only 10 classes.

Question 2

Use the **Appendix A1.1** to set up the Maverick2 environment and the **Appendix A3.1** to train both models in parallel, on a single Maverick2 GTX computation node.

Problem 2: Deployment on the Edge Using ONNX

In this part of the homework, you will deploy the VGG11 and VGG16 models on real edge devices (i.e., Odroid MC1 and Raspberry Pi 3B+) using the ONNX framework. You need to compare how these two models perform when doing inference on edge devices. Make sure you follow the instructions in **Appendix A3.2** to install ONNX locally on your machine before you start working on this problem. Use **Appendices A2.1** and **A3.3** to connect to each device properly.

Question 1

The first three parameters for the convert function are: the model itself, the random input tensor with the same input size as the model, and the file name to export to. For Odroid MC1, use:

```
torch.onnx.export(your_model, random_input_tensor,
                  net_name_pt.onnx, export_params=True, opset_version=13)
```

For RaspberryPi 3B+, use:

```
torch.onnx.export(your_model, random_input_tensor,
                 net_name_pt.onnx, export_params=True, opset_version=17)
```

Question 2

You need to send all ONNX models (obtained in **Problem 2, Question 1**) and the ***deploy_onnx.py*** file to each device over SSH using the ***scp*** command (similar to HW2). **The testing accuracy you obtain when deploying the models** on the edge devices must be the same as the one obtained in **Problem 1, Question 2** for the VGG11 and VGG16 models, respectively.

The RAM memory consumption for RaspberryPi 3B+ and Odroid MC1 is the amount of RAM memory consumed when doing inference. Using the info in the **Appendix A3.5,** check the amount of RAM memory before running inference, and write it down (this is the idle memory consumption). After you start doing inference, check the memory usage value again (this is the inference memory consumption + idle memory consumption) and write the difference between the current value and the previous one in ***Table 2*** (i.e., how much memory the inference process consumes compared to the idle state).

Of note, the inference is when ***sess.run()*** is being executed, *not* when considering all the image loading and preprocessing times.

Question 3

For both RaspberryPi 3B+ and Odroid MC1, the power measurement process is identical (since both RaspberryPi and Odroid are connected to the Smart Power 2 device) and needs to be done as in **HW2**. Check **Appendix A3.4** to see how to collect temperature measurements for the RaspberryPi device. Because you have multiple temperature sensors for the Odroid MC1, the temperature you need to use throughout this homework will be the *average* of all four big core temperatures.

Problem 3: MobileNet-v1 on Edge Devices

In this problem you will deploy the MobileNet-v1 model on Raspberry Pi 3B+ and Odroid MC1 devices. As discussed in **Lecture 5**, the MobileNet-v1 model is specifically designed and optimized for edge devices so this architecture is very relevant to edge applications. You need to explore (in terms of training and inference latency, energy consumption and accuracy performance) what it means to have a model that is optimized for edge devices, as opposed to the VGG11 and VGG16 models which are general purpose models.