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

Spring 2026

ECE 361E: Machine Learning and Data Analytics for Edge AI
HW1 Assigned: Jan 20 DUE: Jan 29 (11:59:59pm CST)

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 is the main document of the homework assignment with all problems and afferent questions. If you need clarifications, read the ***HW1_README*** file which can help you. Finally, the ***Appendix_A1*** contains more practical things such as code examples to run and debug.

This assignment is meant to be an introduction to training and testing Machine Learning (ML) models on the MNIST dataset[1] in the Cloud. Specifically, you will be working with PyTorch, one of the most popular frameworks for developing ML models. By working on this assignment, you will learn:

- The basics of ML techniques (e.g., training and testing a model, handling overfitting, comparing and contrasting various optimizers, evaluating the complexity of a model and fine-tuning the hyperparameters of a model);
- How to visualize various metrics and parameters and their impact on different design decisions;
- How to interpret and discuss the significance of your experimental results.

Problem 1 [20p]: Logistic Regression (using TACC Machines)

Question 1: [5p] Modify ***starter.py*** by specifying the target device for your model (by adding ***model = model.to(torch.device('cuda'))*** at line 66) to train the model on a GPU. Run ***starter.py*** on Lonestar6 (see **Appendix A1.1** for environment setup, **Appendix A1.2** to run your code, and **Appendix A1.3** for additional details on how to run jobs on TACC).

Question 2: [6p] Draw a *loss plot* consisting of two curves: the *training loss* and the *test loss* of your model for each *epoch*. Also, draw a second plot consisting of two curves: the *training accuracy* and *test accuracy* of your model for each *epoch* (this is the *accuracy plot*).

Question 3: [9p] Complete ***Table 1*** (use the accuracy values obtained in the final epoch):

Table 1

Training accuracy [%]	Testing accuracy [%]	Total time for training [s]	Total time for inference [s]	Average time for inference per image [ms]	GPU memory during training [MB]

Problem 2 [40p]: Overfitting, Dropout and Normalization

Question 1: [8p] Run the code from ***simpleFC.py*** after specifying the target device for the model, images and labels. Draw the loss and accuracy plots (like in **Problem 1, Question 2** above). Does this model overfit? Explain.

Question 2: [14p] In the ***__init__()*** method from the ***SimpleFC*** class, define once ***nn.Dropout(probability)*** and apply this new operation in the ***forward()*** method after every layer of the model except the last one.

[1] Modified National Institute of Standards and Technology database (MNIST)

Run four different experiments using the values [0.0, 0.2, 0.5, 0.8] as probabilities for dropout. Draw one loss plot for each experiment. What do you observe from these plots? Which dropout probability gives the best/worst results (the best results have no overfitting, i.e., almost equal train and test loss values)? Explain.

Question 3: [18p]. Complete ***Table 2*** by running the same code as in **Problem 2, Question 2**, but using only the best dropout rate. In ***Table 2***, replace X with the dropout rate which led to the best results in ***Problem 2, Question 2***. For the row labelled "+ norm", keep the same dropout probability and add normalization to both training and testing datasets. Compare and contrast these normalized and unnormalized experiments and explain the differences.

Table 2

Dropout	Training accuracy [%]	Testing accuracy [%]	Total time for training [s]	First epoch when the model reaches 96% training accuracy
X				
X + norm				

Problem 3 [40p + 10Bp]: CNNs and Model Complexity

Question 1: [20p] Implement the normalized MNIST dataset from **Problem 2, Question 3** in ***simpleCNN.py***. Using the same file, write the necessary code to complete ***Table 3*** and then run it with this dataset. Is there any difference between the estimated (total) size of the model from ***torchsummary*** and the saved version of the model (i.e., file size on disk)? Explain.

Table 3

Model name	MACs	FLOPs	# parameters	***torchsummary*** (total) size [KB]	Saved model size [KB]
SimpleCNN					

Question 2: [20p] Create your own CNN with at least two convolutional layers and train it for 25 epochs on the normalized MNIST dataset. Try making this model more efficient (e.g., smaller number of parameters and/or smaller model size) compared to the SimpleCNN model from **Problem 3, Question 1**. Plot the loss and accuracy curves as a function of the #epochs. Extend ***Table 3*** further with a new row with your model name (e.g., "myCNN") and show the new results.

BONUS Question 3: [10Bp] Explain your choice for the model architecture. Does your model overfit? How does your proposed model compare against SimpleCNN in terms of training and testing accuracy? In a real scenario, would you prefer using your proposed model or SimpleCNN? Why or why not?

Submission Instructions

Include your solutions 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. Your code files, named suggestively (e.g., **p1_q1.py** for **Problem 1 Question 1** code).
3. A readme.txt file describing all your items in the zip file.

Good luck!