MLFA Lab Assignment: Feedforward Neural Network & CNN Model

Total Points: 40

Objectives:

Build and optimize a Neural Network model for multiclass classification on the MNIST dataset.

- Perform multiclass classification on the MNIST dataset (download it from here) using data augmentation to address the imbalanced classes.
- Implement hyperparameter tuning to optimize the model's performance.
- Utilize data loaders to efficiently manage the dataset and improve training efficiency.
- Plot the training and validation loss graphs over epochs.
- Calculate the number of parameters and FLOPs (floating-point operations) in the model.
- Generate a confusion matrix and calculate precision and recall scores.
- Identify and describe the different layers used in the neural network model.

Tasks

Question 1

Dataset:

The MNIST dataset comprises 28x28 grayscale images of handwritten digits (0-9).

(Dataset: https://drive.google.com/file/d/1YDqR-i0eSo_LGwTiwxbriLH_90MYyKNB/view?usp=sharing)

1. Data Augmentation (3 Points):

- Use *torchvision.transforms* to define a series of transformations to augment the training data.
- Consider transformations like random rotation, random horizontal flip, and random crop.
- Apply the transformations only to the training set, not the validation or test sets.

2. Hyperparameter Tuning (4 Points):

- Keep number of hidden layers as 2, batch size as 32 and epochs as 10. Use ReLU as the activation function.
- Define a grid of hyperparameters to tune, namely, hidden layers size (number of units in each hidden layer) and learning rate.
- Track the model's performance on the validation set during training and select the hyperparameters that yield the best results.

3. Data Loaders (3 Points):

- Use *torch.utils.data.DataLoader* to create data loaders for the training, validation, and test sets. Split the training data from MNIST into validation & training set.
- Set appropriate batch sizes for each data loader.

4. Model Training and Evaluation (4 Points):

- Train the model using the augmented training data and the selected hyperparameters.
- Evaluate the model's performance on the validation and test sets using accuracy and loss (*Cross Entropy Loss*).
- Visualize the training and validation loss graphs over epochs.

5. Model Analysis (3 Points):

- Calculate the number of parameters, FLOPs of the model using a library like *flopth*, *torchprofile*.
- Generate a confusion matrix and calculate precision and recall scores using *torchmetrics*.

6. Model Architecture (3 Points):

- Describe the different layers used in the neural network model, including their purpose and functionality.
- Explain how the layers are connected and how they contribute to the overall model architecture.
- Mention the description and explanation in the IPython Notebook (.ipynb).

QUESTION 2

CNN Model for MNIST Classification

Objective Build and optimize a Convolutional Neural Network (CNN) model for multiclass classification on the MNIST dataset. Compare the performance, including accuracy, precision, recall, number of trainable parameters, and FLOPs, with a feedforward neural network model.

Use the **same MNIST dataset and Data Loaders** setup as in Question 1 (Feedforward Neural Network approach). This includes any preprocessing or data augmentation steps applied to the dataset.

1. Model Architecture (7 Points):

Design a CNN model with the following specifications: consider a Convolutional Neural Network (CNN) that starts with a **convolutional layer**, followed by a **max-pooling layer**. Another convolutional layer comes next, with a second max-pooling layer following it. The sequence concludes with a fully connected layer for classification into one of 10 categories.

- Two convolutional layers followed by a linear (fully connected) layer.
- Utilize ReLU activation functions for the convolutional layers.
  Describe the layers used in your CNN model, including:
- The purpose and functionality of each layer.
- The connection between layers and their contribution to the model's overall architecture.

This description should be included in your IPython Notebook (.ipynb).

2. Hyperparameter Tuning (6 Points):

- Focus on tuning the number of filters in each convolutional layer and the learning rate.
- Keep the number of epochs as 10 and the batch size as 32, as set in Question 1.
- Track the model's performance on the validation set during training. Select the hyperparameters that yield the best results.

3. Model Training and Evaluation (7 Points):

- Train your CNN model using the augmented training data and the selected hyperparameters.
- Evaluate the model's performance on the validation and test sets. Report the accuracy, loss, precision, and recall.
- Visualize the training and validation loss graphs over epochs.
- Compare the accuracy, precision, and recall scores with those obtained from the feedforward neural network model in Question 1. Discuss any observed differences.
- Calculate and compare the number of trainable parameters and FLOPs in both models (CNN and feedforward neural network).

Submission Guidelines

A .zip file containing the python source code (No PDF report required to submit).

The final name should follow the template: _.zip. For example, if your roll no is 15CE30021, the filename for Assignment 6 will be: Assign-6_15ce30021.zip

1. Include an IPython Notebook (.ipynb) file containing the code implementations and outputs of your experiments.
   The first two lines of this file should include your name and roll number for identification purposes.
2. Submit a single Python code file (.py) containing the implementations of your model and experiments.
   The first two lines of this file should include your name and roll number for identification purposes.

Both .py and .ipynb files need to be submitted to receive credit on the assignment.

NOTE: Plagiarism will be penalized.