Assignment 4

In this assignment, you will explore the concept of transfer learning by using a pre-trained model on the MNIST dataset and applying it to recognize your own handwritten digits. This exercise will give you practical experience with transfer learning, fine-tuning, and working with image data.

Dataset Preparation:

- Begin by familiarizing yourself with the MNIST dataset of handwritten digits. This dataset is publicly available and widely used for digit recognition tasks.
- Create your own handwritten digit dataset. Write the numbers 0 through 9 twelve times on a sheet(s) of paper, take a photo or scan it, and preprocess these images into separate images for each digit (28x28 pixels, grayscale, like the MNIST dataset).
- Use 8 images per digit for training, 2 image per digit for validation, and 2 image per digit for testing. This will give you a training set of 80 images and validation and test sets of 20 images each.
- The validation set should be used for tuning the model's layers, dropout rates, and learning rate adjustments if needed, while the test set should remain untouched until the end of training for a final, unbiased evaluation.

Model Setup:

- Use a pre-trained model for digit recognition trained on the MNIST dataset. You may use a pre-trained PyTorch model.
- Load the pre-trained model and freeze its initial layers to retain the learned features. Experiment with fine-tuning just the last layer versus a few layers. In the report, highlight the differences you observed in performance and training stability. Explain the reasons for those differences.

Training and Fine-tuning:

- Implement transfer learning by training the model on your handwritten digit dataset.
- Perform hyperparameter tuning (e.g., learning rate, batch size) to optimize the model for your data.
- Train two models one that finetunes only the last layers whereas the other finetunes two or more layers.

Evaluation:

- Evaluate the following models' performance using accuracy as a metric. Compare the results and fill the corresponding entries in the given Excel sheet "Assignment 4_Results.xlsx".

1. The pre-trained PyTorch model evaluated on the standard MNIST test set (10,000 images).
2. The pre-trained PyTorch model evaluated on your test set (20 images).
3. Your model with only the last layer finetuned evaluated on your test set (20 images).
4. Your model with *n* last layer finetuned evaluated on your test set (20 images).

- In the report, analyze the results and document any patterns or insights observed.

Report:

Write a brief report discussing:

- Your dataset details.
- The name of the pre-trained model (e.g. AlexNet, GoogLeNet, ResNet) and why you selected that.
- Your hyperparameter tuning process, i.e., rationale behind choosing specific values for learning rate, batch size and epochs. Also describe any challenges faced in tuning the hyperparameters (e.g., if certain values led to overfitting or instability) and how you addressed them.
- Comparison, analysis, insights and observations from the results in "Assignment 4_Results.xlsx".

The report should not be more than 2 pages of text (excluding the images).

Deliverables:

Submit a zip file containing:

- A folder named "Code" with the complete source code, ensuring it's executable on another system without additional modifications. This can include Python files (.py) or Jupyter Notebooks (.ipynb).
- Three folders named "Training Set", "Validation Set", and "Test Set", respectively, that contain your image dataset.
- A PDF file named "Assignment 4_Report" for documentation.
- An Excel file named " Assignment 4_Results".
- A requirements.txt file listing all the dependencies.

Resources:

- MNIST Dataset, PyTorch's builtin MNIST Dataset
- Pre-trained Models
- Tutorial on Transfer Learning (replace with the most appropriate resources based on your preferred framework, such as PyTorch or TensorFlow).

Instructions:

- Use the folder structure provided with the assignment description on MLS. Also, make sure to follow all naming conventions and formats.

- You may use high-level functions from OpenCV, NumPy, or other libraries. Provide a requirements.txt file listing all the dependencies.
- Store all images as ".jpg" for consistency.
- Use Python version > 3.6 and OpenCV version > 3.4
- All code that you submit should either be original or, if sourced from the internet, properly cited as indicated earlier.
- Your code must run on the grader's computer without any changes. Specifically, do not use any hardcoded path, or any piece of code that is dependent on anything not specifically noted in requirements.txt. Non-compliance may result in a zero mark in the assignment.
- Your code should be well-commented to explain your logic. A clear and well-structured code can significantly aid in understanding and help the grader follow your logic.
- Submit all materials to MLS in a single zip file.

Marking Scheme

Task	**Marks**
Quality of Dataset	10
Code Quality and Results Accuracy	20
Report	10
Total	**40**

A rubric for detailed marks distribution is attached with the project description on MLS.