ML Ops

Major Assignment

1 Overview & Learning Objectives

Goal

This assignment will guide you through building a complete, automated MLOps pipeline. You will develop a PyTorch model, containerize it with Docker, build a CI/CD workflow with GitHub Actions to automate training and deployment tasks, and finally, optimize your model using quantization.

2 Dataset & Model

- **Dataset:** You will use the **Olivetti faces dataset** from `sklearn.datasets`.
- **Model:**
  1. You will train a **DecisionTreeClassifier** model using **scikit-learn**.

3 Submission & Grading

3.1 Submission Requirements

→ You must submit a **single PDF file** named `YourRollNo_Major.pdf`.

→ The first page must contain the **public link to your GitHub repository** and the **public link to your Docker Hub repository**.

→ The repository must contain all source code, the `Dockerfile`, the GitHub Actions workflow (`ci.yml`), other required files and a `README.md`.

→ All branches (`main`, `dev`, `docker_cicd`) must be present in the final repository.

→ Comprehensive documentation is required, including step-by-step screenshots covering both command-line interactions and GitHub workflows. A detailed analysis should also be included.

3.2 Grading Distribution (Total: 50 Marks)

Code + Git: 10 Marks

Documentation: 10 Marks

Viva: 30 Marks

4 Step-by-Step Workflow & Task Breakdown

You must follow this precise Git branching strategy.

4.1 Step 1: `main` Branch - Initial Setup

1. Initialize your GitHub repository with a `README.md` and a `.gitignore` file.

4.2 Step 2: `dev` Branch - Model Development

1. Create and switch to a new branch named `dev`.
2. Create a `train.py` script that loads the sklearn Olivetti faces dataset and uses the `train_test_split` function to split the data into 70% trainset and 30% testset. Next, train a scikit-learn `DecisionTreeClassifier` model on the trainset and save the model using `joblib` as `savedmodel.pth`.
3. **Create a `test.py` script** that loads the `savedmodel.pth` model, computes its accuracy on the test set and display the test accuracy.
4. **Configure the CI/CD workflow** in `.github/workflows/ci.yml` to run on `push`. The workflow must perform the following:

   Job: `check_working_repo`
   (a) Check out your repository's code.
   (b) Set up a Python environment and install dependencies.
   (c) Run the `train.py` script to generate the `savedmodel.pth` model file.
   (d) Run the `test.py` script to display the test accuracy of the saved model file.

5. **Do not merge** this branch into `main`.

4.3 Step 3: `docker_ci` Branch - Automation with Docker & CI/CD

1. Create and switch to a new branch named `docker_cicd` from `dev`.
2. **Develop a Flask web application** that will serve a web page to upload images and display the predicted class. It will use the trained model saved earlier.
3. **Create a `Dockerfile`** in the root directory.
4. **Build the Docker image** using the `Dockerfile` that runs the Flask web application using the saved trained model.
5. **Push the Docker image to Docker Hub**
6. **Deploy these containers using Kubernetes** while ensuring **3 replicas** are always running.
7. **Do not Merge** `docker_cicd` branch back into `main`.