Functional Requirements

1. Introduction

- This document outlines the functional requirements of a CIFAR-10 dataset based microservice based application that trains , evaluates, predicts, stores, and visualises the performance metrics like accuracy, precision, recall and f1 score of a machine learning model.
- **Scope** : Python FastAPI for microservice development, MongoDB for data storing and retrieving , Python Streamlit for frontend visualisation.

2. API Endpoints

2.1 Welcome to the Application API

- Endpoint: /
- Method: GET
- Description: Welcome to the Model Training Microservice!
- Input : Nothing
- Output: “Welcome to the Model Training Microservice!”

2.2 Train Model API

- Endpoint: /train
- Method: POST
- Description: Trains a CNN model using CIFAR-10 dataset
- Input: Model parameters epochs,batch_size,validation_split,learning_rate as query params for endpoint
- Sample Input: train?epochs=40&batch_size=8&validation_split=0.1&learning_rate=0.0005
- Output: JSON response with model train completion and training metrics(test_accuracy,test_loss,validation_accuracy,validation_loss)

- Validation : Check the input parameters in query params and set the value. Otherwise it takes default set parameter value for each. If apply different query param names or more than 4 query params API endpoint works without problem.

2.3 Test Model API

- Endpoint: /evaluate
- Method: POST
- Description: Test the trained model and calculate the metrics accuracy,precision,recall,f1 score.
- Input: Set the CIFAR-10 dataset testing dataset inside the code level
- Output: JSON response about metrics
    - ❖ Accuracy
    - ❖ F1 Score
    - ❖ Precision
    - ❖ Recall

2.4 Retrieve Training Metrics API

- Endpoint: /metrics
- Method: GET
- Description: Retrieves the stored performance metrics from MongoDB
- Input: number of latest records can apply in query params
    - metrics?n=5

- Output: JSON response containing a “n” number of records of stored metrics with timestamps.

2.5 Retrieve Training Details API

- Endpoint: /training-details
- Method: GET
- Description: Retrieves the stored training parameter details from MongoDB
- Input: number of latest records can apply in query params
    - metrics?n=5

- Output: JSON response containing a “n” number of records of stored training parameter details with timestamps.

2.6 Inference Training Model API

- Endpoint: /predict
- Method: POST
- Description: Predict the CIFAR-10 testing randomly selected image using trained model
- Input: Randomly select in testing dataset image in CIFAR-10 dataset.
- Output: JSON output with predicted class,true label,class index and probabilities each class

3. Frontend Visualization

The frontend is developed using Python streamlit for data visualization.

- **Table** : shows the latest 10 testings metrics
- Line charts and Plots for each metric trends over time(Accuracy, Precision, Recall, F1 score)

4. Database

System uses the MongoDB NoSQL database for data storage.

- **Metrics collection :**
  - **Fields:** id, timestamp,accuracy,f1 score,precision,recall

- **Training_details collection:**
  - **Fields:** id, timestamp, epochs, batch_size, learning_rate, validation_split, test_accuracy, test_loss, validation_accuracy, validation_loss

5. Assumptions

- CIFAR-10 dataset is preprocessed.
- System developed in a Linux Environment.

6. Constraints

- Application is designed for single machine setup.

Non-Functional Requirements

1. Performance

- The system shall process API requests and responses under normal load conditions.
- Frontend must render visualisations after retrieving metrics from the database.

2. Scalability

- The microservice supports vertical scaling to accommodate more computational resources.

3. Availability

- Backend,Database and frontend must run well without problems to visualise the charts and tables through the frontend UI.

4. Reliability

- API requests and responses are guaranteed to operation activities status.
- Application must recover within a few minutes.

5. Usability

- Streamlit frontend shall have an informative user interface that gives brief idea about the activities progress in technical manner.
- Error messages guide the user well.

6. Maintainability

- Codebase shall follow the Python coding standard.
- Application must include separate code scripts for each main activity.
- Developers can add new feature or API endpoint with minimal code changes.

7. Portability

- System must be containerized using docker to facilitate deployment in the different environments.