Project Description

This project implements a complete machine-learning pipeline starting from raw data preparation and ending with experiment tracking using MLflow. The goal of the project is to build a gesture-recognition system capable of running in real-time.

The project captures an image of the user's hand gesture and processes it through a trained deep-learning model to recognize the gesture in real time. Once the model predicts the gesture, the system maps it to a specific predefined action, allowing seamless interaction without physical controls. This creates an intuitive and efficient way for users to control applications using only their hands.

Phases the project went through:

1. Data Collection and Preprocessing

- The project begins with gathering the dataset and organizing it under the data/ directory.
- The notebook **Data Preprocessing.ipynb** is used to clean, structure, and prepare the dataset for training.
- Preprocessing includes normalization, transforming data into the required format, and optionally segmenting input images using the segment_image.py script.
- All processed data is saved for later use in the training stage.

2. Model Training

- The **Model_Training.ipynb** notebook contains the full model-building and training workflow.
- A machine-learning model is trained on the processed dataset to classify or recognize hand gestures.
- After training, two artifacts are saved:
    - hand_gesture_norm_model.pkl (the trained model)
    - scaler.pkl (the normalization object used during preprocessing)
- These files are used later for real-time prediction and system integration.

3. Real-Time Detection and System Integration

- The project includes real-time interaction components:
    - **realtime_detection.py** handles real-time prediction using a webcam or input feed.
    - **realtime_gesture_controller.py** connects model predictions to actionable functions, allowing gesture-based control.
- The file **main.py** provides a runnable entry point to the integrated system.

4. Monitoring and MLflow Integration

- To ensure reproducibility and experiment tracking, MLflow is integrated using the **mlflow_log_model.py** script.
- Model parameters, metrics, and artifacts are logged to MLflow for experiment management.
- A custom monitoring component is included in **monitoring_module.py**, and runtime logs are stored in Monitoring Data.csv.

Overall Workflow Summary

1. Prepare and clean the dataset
2. Train a machine-learning model for gesture recognition
3. Save the trained model and scaler
4. Run real-time gesture detection and control
5. Track experiments, metrics, and model versions using MLflow