SEG3904 Project Progress

Project Title: Hand Gesture Calculator

Overview:

The purpose of this project is to develop a Hand Gesture Calculator that interprets numerical hand gestures (0-9) using a webcam and performs basic arithmetic operations (addition, subtraction, multiplication, and division). The project will involve implementing computer vision techniques and machine learning algorithms to detect and classify hand gestures in real-time. The application will be developed using Python, leveraging libraries such as OpenCV and TensorFlow. This project will help in understanding how to develop and deploy real-time machine learning models for practical use cases.

Project Progress:

Week 1 (Sept. 3 - Sept. 10, 2024):

- Project Planning: Defined the project goals and requirements for the Hand Gesture Calculator.
- Environment Setup: Installed necessary tools, including Python, OpenCV, TensorFlow, and Keras.
- Initial File Structure: Set up the basic project structure and Git version control.

Week 2 (Sept. 10 - Sept. 17, 2024):

- Data Collection: Started collecting hand gesture images (0-9) using the webcam.
- Script Development: Created image_capture.py to capture and save images in gesture-specific directories.
- Data Organization: Set up directories for each gesture and begin the collection process.

Week 3 (Sept. 17 - Sept. 24, 2024):

- Data Preprocessing: Converted the collected images to grayscale and resized them to a uniform size (128x128).
- Script Development: Created preprocess_images.py to automate image conversion and resizing for consistency.

Week 4 (Sept. 24 - Oct. 1, 2024):

- Data Preparation: Loaded the preprocessed images, labeled them, and split the dataset into training and testing sets.
- Script Development: Created prepare_data.py to organize, label, and split the data into training and testing sets (90/10 split).

Week 5 (Oct. 1 - Oct. 8, 2024):

- Model Design: Built a Convolutional Neural Network (CNN) model using TensorFlow/Keras for gesture recognition.
- Model Training: Trained the model using the collected and preprocessed gesture data.
- Script Development: Created model.py to define, train, and save the trained model.

Week 6 (Oct. 8 - Oct. 15, 2024):

- Real-Time Gesture Recognition: Implemented a real-time gesture recognition system using the webcam.
- Script Development: Created main.py to predict gestures using the trained model.

Week 7 (Oct. 15 - Oct. 22, 2024):

- UI Enhancement: Added on-screen buttons for arithmetic operations (+, -, *, /).
- Operation Implementation: Enhanced the system to allow users to select operations via clickable buttons, and perform calculations using recognized hand gestures.

Week 8 (Oct. 22 - Oct. 29, 2024):

- Process Rework: Integrated the image preprocessing and loading steps directly into model.py for improved workflow.
- Code Optimization: Streamlined the data pipeline for better performance and maintainability.

Week 9 (Oct. 29 - Dec. 5, 2024):

- Data Collection: Collected additional gesture images to improve accuracy, focusing on diverse lighting, angles, and hand positions.

Week 10 (Nov. 5 - Nov. 12, 2024):

- Model Optimization: Implemented techniques such as data augmentation, a learning rate scheduler, and regularization to improve model accuracy and prevent overfitting.
- Testing: Evaluated the model's performance using the expanded dataset

Week 11 (Nov. 12 - Nov. 19, 2024):

- Integration of MediaPipe: Integrated MediaPipe for hand landmark detection, replacing the image-based approach.
- Landmark Extraction: Extracted 3D hand landmarks (x, y, z) for gestures 0-9 using extract_landmarks.py.
- Landmark Dataset Creation: Generated the gesture_landmarks_v3.pkl file containing extracted landmarks for training.

Week 12 (Nov. 19 - Nov. 26, 2024):

- Model Redesign: Updated the model architecture to process hand landmarks instead of raw images, leading to improved accuracy and efficiency.
- Model Training: Trained the updated model using the new landmark-based dataset.

- Model Evaluation: Achieved better results with faster training and inference times.

Week 13 (Nov. 26 - Dec. 3, 2024):

- UI Development: Created the first version of the graphical user interface (GUI) using OpenCV:
    - Displayed the live webcam feed.
    - Added buttons for operations (+, -, *, /).
    - Displayed real-time gesture recognition, selected operations, and results.
- Data Collection: Added more diverse images for gestures to reduce confusion between similar gestures (e.g., 5 and 9).
- Directory Reorganization: Cleaned up and restructured project directories for better clarity.

Week 14 (Dec. 3- Dec. 10, 2024):

- UI Enhancement: Upgraded the GUI using PyQt5 for better functionality and appearance.
    - Integrated real-time gesture recognition into the PyQt5 interface.
    - Added visual indicators for gestures, operations, and results.
    - Improved responsiveness and performance of the user interface.

Week 15 (Dec 10 - Dec. 17. 2024):

- Final Testing and Bug Fixing: Tested the entire system for stability, accuracy, and edge cases.
- Project Finalization: Prepared the project for demonstration and submission
    - Polished the codebase.
    - Updated the README file with detailed instructions and documentation.
- Self-Assessment and Future Work: Reflected on the learning process and outlined future steps for improvement.

Self Assessment of Learning and Future Work Items:

Self Assessment of Learning:

- I gained significant knowledge throughout the course of this project, particularly in integrating machine learning concepts with real-time computer vision applications.
- Initially, I was unfamiliar with MediaPipe and how it could be used to extract 3D hand landmarks, but I learned how to leverage its capabilities effectively.
- At the beginning, I was focused on training a CNN model with raw image data. While functional, the approach lacked accuracy and efficiency.
- Realizing the need for a better method, I researched alternative solutions and discovered landmarks which then allowed me to discover MediaPipe and its pre-trained hand

tracking system which could extract critical hand features, reducing computational overhead.

- I learned to preprocess and structure the extracted landmarks into a format suitable for training a neural network, including handling missing data through zero-padding.
- I also improved my understanding of model architecture design by creating a feed-forward neural network optimized for classification tasks with dropout layers and batch normalization to prevent overfitting.
- Throughout the process, I learned how to debug and optimize a project by identifying issues like inconsistent input dimensions, handling cases where MediaPipe only detected one hand, and ensuring smooth gesture recognition for both one and two-handed gestures.

Future Work Items:

- Collect more diverse and extensive training data under different lighting conditions and hand orientations to further improve model robustness.
- Extend the project to recognize additional gestures beyond numbers (e.g., control gestures like "AC", "+", or custom signs).
- Enhance the application's efficiency to reduce latency for real-time gesture recognition, particularly on lower-end devices.