**Gesture Recognition - Project**

**Group Members:**

Manikesh Boyini

Hemal Patel

**Table of Contents:**

1. **Introduction**
2. **Problem Statement**
3. **Understanding the Dataset**
4. **Project Scope and Objectives**
5. **Methodology**
   * Data Preprocessing
   * Model Architecture
   * Model Training and Evaluation
6. **Experiments and Results**
7. **Final Model Recommendation and Conclusion**

**1. Introduction:**

The goal of this project is to develop a gesture recognition system for smart TVs that enables users to control their device without the need for a remote control. Instead, users will be able to perform five specific gestures, each corresponding to a TV command such as volume control, rewinding, fast-forwarding, and pausing the video. This gesture-based interaction aims to improve the user experience by offering hands-free, intuitive control, which enhances accessibility and convenience.

**2. Problem Statement:**

This project focuses on building a gesture recognition system using a webcam mounted on a smart TV to detect five distinct gestures. These gestures are mapped to specific TV control functions:

* **Thumbs Up**: Increase the volume
* **Thumbs Down**: Decrease the volume
* **Left Swipe**: Rewind the video by 10 seconds
* **Right Swipe**: Fast forward the video by 10 seconds
* **Stop**: Pause the video

The gesture recognition system must function in real time, accurately interpreting the gestures and executing the corresponding TV commands.

**3. Understanding the Dataset:**

The dataset for this project consists of several hundred video samples, categorized into five gesture classes. Each video consists of 30 frames and is recorded by different users under varying conditions. The dataset is organized into two main folders: "train" for training data and "Val" for validation data. Each video is stored in subfolders, with each subfolder containing 30 frames.

Each subfolder name represents a specific gesture (Thumbs Up, Thumbs Down, Left Swipe, Right Swipe, Stop), and each video is labelled with a numeric code (0-4) corresponding to these gestures.

**Data Challenges:**

* **Resolution Variability**: The videos are available in two resolutions: 360x360 and 120x160. To address this, standardization of resolution is necessary.
* **Inconsistent Lighting and Backgrounds**: Variations in lighting and background conditions require preprocessing and data augmentation to improve the model's robustness.

**4. Project Scope and Objectives:**

The scope of this project includes developing a machine learning model capable of recognizing gestures from video frames and mapping them to corresponding TV control commands in real-time. Key objectives are:

* Achieving high gesture recognition accuracy across various environments.
* Ensuring real-time performance for a smooth user experience.
* Preprocessing video frames to ensure consistency and robustness across various conditions.

**5. Methodology:**

**5.1 Data Preprocessing:**

To ensure uniformity across the dataset, all video frames were resized to a fixed resolution (e.g., 100x100). Additional preprocessing steps such as normalization, resizing, and data augmentation (e.g., random cropping) were applied to account for variability in resolution and environmental factors. These steps ensure that the model can handle different lighting and background conditions effectively.

**5.2 Model Architecture:**

Three different model architectures were explored to find the most effective approach for gesture recognition:

**Approach 1: 3D Convolutional Layers with 3D Max Pooling**

* **Description**: This model uses 3D convolutional and pooling layers to process videos as 3D tensors, capturing both spatial and temporal features across the frames.
  + **3D Convolutional Layers**: Capture spatiotemporal features.
  + **3D Max Pooling**: Reduces dimensionality across all three axes.
  + **Fully Connected Layers**: Classifies the output gesture.
* **Pros**: Effective in learning spatiotemporal features and has fewer parameters than RNNs.
* **Cons**: Limited in capturing long-term dependencies and uses more memory.

**Approach 2: CNN + RNN (LSTM or GRU)**

* **Description**: This approach uses a 2D CNN to extract spatial features from each frame, which are then fed into an LSTM or GRU to capture temporal dependencies.
  + **2D CNN Layers**: Extract spatial features from each frame.
  + **LSTM/GRU Layers**: Learn the temporal patterns across frames.
  + **Fully Connected Layers**: Classify the gesture.
* **Pros**: Good for learning long-term dependencies by handling spatial and temporal information separately.
* **Cons**: Higher complexity and requires careful tuning.

**Approach 3: CNN Transfer Learning (MobileNetV2 and VGG16) + LSTM**

* **Description**: In this model, pre-trained CNNs (MobileNetV2 or VGG16) are used to extract spatial features, followed by an LSTM layer to model the temporal sequence for gesture classification.
  + **Pre-trained CNN (MobileNetV2/VGG16)**: Efficient feature extraction.
  + **LSTM Layer**: Models the temporal sequence for gesture classification.
  + **Fully Connected Layers**: Output layer for classification.
* **Pros**: Efficient use of pre-trained features and works well with minimal data. MobileNetV2 is lightweight and suited for embedded devices.
* **Cons**: VGG16 may be too large for real-time use, and the approach requires more memory.

**5.3 Model Training and Evaluation:**

The model was trained using optimizers like SGD and Adam with controlled learning rates. The loss function used was categorical cross-entropy, and the model was trained for multiple epochs with a batch size of 32. To prevent overfitting, dropout layers and data augmentation were applied.

The model’s performance was evaluated based on accuracy and loss for both the training and validation sets, along with other factors such as training time, memory usage, inference speed, and latency to ensure real-time suitability.

**6. Experiments and Results:**

A series of experiments were conducted to fine-tune model parameters, optimize hyperparameters, and compare the performance of different architectures. Each experiment was evaluated on a fixed validation set to ensure consistency and reliability.

**7. Final Model Recommendation and Conclusion:**

After evaluating the different approaches, **Approach 3 (MobileNetV2 with transfer learning)** was found to be the optimal choice for the final model. This approach provides a balanced trade-off between accuracy, loss, and computational efficiency.

* **High Accuracy and Low Loss**: Achieved a maximum training accuracy of 0.96, a validation accuracy of 0.88, a minimum training loss of 0.12, and a minimum validation loss of 0.21.
* **Moderate Parameter Count**: Freezing the first 100 layers of MobileNetV2 reduces the number of parameters while still leveraging transfer learning effectively.
* **Efficient Training Time**: The use of global average pooling reduces the feature size, resulting in faster training without sacrificing performance.

**Future Work and Improvements:**

Future improvements could involve:

* Expanding the dataset.
* Exploring more efficient CNN architectures or advanced models.
* Testing the model in real-time scenarios to enhance robustness.

In conclusion, this project successfully developed a gesture recognition system that can interpret user gestures and control a smart TV in real-time. Further enhancements can improve its robustness and accuracy in practical applications.