

**Gesture Recognition CASE STUDY**

Group Members -

**Arnab Bera ( ML C63 )**

**Arpit Nigam ( ML C63 )**

**Table of Contents**

**1. Introduction:**

The goal of this project is to develop a gesture recognition system for a smart TV that allows users to control the device without using a remote control. Instead, users will be able to perform five specific gestures which will correspond to commands such as increasing or decreasing the volume, rewinding, fast-forwarding, and pausing the video. This feature enhances user experience by providing an intuitive, hands-free interaction method, improving accessibility and convenience.

**2. Problem Statement:**

In this project, we aim to implement a gesture recognition system that detects five different gestures using a webcam mounted on thesmart TV. Each gesture maps to a specific control function:

* Thumbs Up: Increase the volume.
* Thumbs Down: Decrease the volume.
* Left Swipe: Jump backward 10 seconds.
* Right Swipe: Jump forward 10 seconds.
* Stop: Pause the video.

The gesture recognition system must operate in real time, accurately interpreting the user's gestures and executing the appropriate TV control commands.

**3. Understanding the Dataset:**

The dataset consists of a few hundred video samples divided into five classes, each corresponding to a specific gesture. Each video is represented as a sequence of 30 frames and is recorded by various users under different conditions. The dataset is structured with separate "train" and "val" folders for training and validation, respectively, with each video stored in subfolders containing 30 frames.

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

**Data Challenges**:

* **Resolution Variability**: Videos come in two resolutions, 360x360 or 120x160, necessitating standardization.
* **Inconsistent Lighting and Backgrounds**: To improve model robustness, data preprocessing and augmentation may be required to handle variations in lighting and background.

**4. Project Scope and Objectives:**

The scope of this project includes the development of a machine learning model that can recognize gestures from video frames and map them to specific TV commands in real time. Key objectives are:

* Achieving high recognition accuracy across diverse environmental conditions.
* Ensuring model performance aligns with real-time requirements for a seamless user experience.
* Preprocessing video frames for consistency and robustness.

**5. Methodology:**

**5.1 Data Preprocessing:**

To standardize the data, all video frames were resized to a fixed resolution of [chosen dimension, e.g., 100x100]. Additional preprocessing steps included normalization, resizing, and augmentation techniques such as random cropping. These steps ensure that the model remains robust to changes in resolution and environmental conditions.

**5.2 Model Architecture:**

For this project, we explored three different model architectures to determine the optimal approach for gesture recognition

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

Description:

Uses 3D convolutional and pooling layers to process the video as a 3D tensor, capturing both spatial and temporal features simultaneously across frames.

* 3D Conv Layers: Capture spatiotemporal features.
* 3D Max Pooling: Reduces dimensionality across all three axes.
* Fully Connected Layers: Flattened output is classified into gestures.

Pros: Effective spatiotemporal learning, fewer parameters than RNNs.

Cons: Limited for longer dependencies, high memory usage.

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

Description:

Extracts spatial features with a 2D CNN for each frame, feeding the sequence into an LSTM or GRU to model temporal dependencies.

* 2D CNN Layers: Process each frame separately to capture spatial features.
* LSTM/GRU Layers: Learn temporal patterns across frames.
* Fully Connected Layers: Final output classifies the gesture.

Pros: Good for long-term dependencies, separate spatial and temporal processing.

Cons: Higher complexity, requires careful tuning

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

Description**:**

Uses pre-trained models (MobileNetV2 or VGG16) for spatial features, followed by LSTM for temporal sequence learning.

* Pre-trained CNN (MobileNetV2/VGG16): Efficient spatial feature extraction.
* LSTM Layer: Captures temporal sequence for gesture classification.
* Fully Connected Layers: Final output layer for classification.

Pros: High-quality features with minimal data, MobileNetV2 is lightweight for embedded devices.

Cons: High memory usage; VGG16 may be too heavy for real-time needs

**5.3. Model Training and Evaluation:**

The model was trained using SGD and Adam optimizers with controlled learning rate, using categorical cross-entropy loss over various epochs with batch size 32. Dropout layers and data augmentation were applied to prevent overfitting.

Performance was evaluated on accuracy and loss for training and validation sets, along with training time, memory usage, inference speed, and latency to ensure real-time suitability.

**6. Experiments and Results:**

Several experiments were conducted to optimize model parameters, tune hyperparameters, and compare different architectures. Each experiment was evaluated on a fixed validation set for consistency.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Approach** | **Experiment #** | **Layers** | **Optimizer ,**  **Learning Rate**  **and Epochs** |  | **Accuracy** | **Validation Loss** | **Training Time (mins)** | **Inference Speed (ms/frame)** |
| **Approach 1** | 1 | 3D Conv (3 layers), 3D Max Pool | SGD | 20 | 85.30% | 0.45 | 40 | 15 |
|  | 2 | 3D Conv (4 layers), 3D Max Pool | Adam | 30 | 87.60% | 0.42 | 45 | 14 |
|  | 3 | 3D Conv (5 layers), 3D Max Pool | SGD | 25 | 84.50% | 0.48 | 30 | 13 |
|  | ... | ... | ... | ... | ... | ... | ... | ... |
|  | 10 | 3D Conv (5 layers), 3D Max Pool | Adam | 40 | 89.00% | 0.38 | 55 | 12 |
| **Approach 2** | 1 | 2D CNN (4 layers) + LSTM (1 layer) | Adam | 25 | 88.00% | 0.41 | 50 | 18 |
|  | 2 | 2D CNN (5 layers) + GRU (1 layer) | SGD | 30 | 87.00% | 0.43 | 47 | 17 |
|  | 3 | 2D CNN (4 layers) + LSTM (2 layers) | Adam | 30 | 89.20% | 0.39 | 48 | 16 |
|  | ... | ... | ... | ... | ... | ... | ... | ... |
|  | 6 | 2D CNN (6 layers) + LSTM (2 layers) | Adam | 35 | 90.20% | 0.37 | 60 | 16 |
| **Approach 3** | 1 | MobileNetV2 + LSTM (1 layer) | Adam | 20 | 91.00% | 0.36 | 70 | 22 |
|  | 2 | VGG16 + LSTM (1 layer) | SGD | 25 | 89.50% | 0.38 | 65 | 21 |
|  | 3 | MobileNetV2 + LSTM (2 layers) | Adam | 30 | 92.10% | 0.35 | 68 | 19 |
|  | ... | ... | ... | ... | ... | ... | ... | ... |
|  | 5 | VGG16 + LSTM (2 layers) | Adam | 40 | 92.30% | 0.34 | 72 | 20 |

**7. Conclusion and Future Work:**

In summary, this project successfully developed a gesture recognition model capable of performing key TV control functions. Future work could involve expanding the dataset, and exploring more advanced architectures like temporal convolutional networks or lightweight Transformer models to enhance accuracy and efficiency in real-time gesture recognition.