Gesture Recognition Utilizing CNN-LSTM and 3D CNN Models

A) Problem Definition

The objective of this project was to develop a gesture recognition model capable of classifying five distinct hand gestures from video sequences. This model is intended for implementation on a smart TV, where gestures captured via a webcam can serve as remote control commands. The challenge was to create a model that generalizes well to unseen data while maintaining low computational costs for real-time use.

B) Dataset and Preprocessing

The dataset comprised video sequences, each containing 30 frames that captured various hand gestures performed by users. Key preprocessing steps included:

1. **Image Resizing**: Images were resized to different resolutions (120x120 and 160x160) to evaluate how varying image resolutions affect model performance.
2. **Normalization**: Pixel values were normalized to a range of [0, 1] by dividing by 255.
3. **Data Augmentation**: To mitigate overfitting, data augmentation techniques were applied in some models. These included random rotations, shifting, and cropping to enhance model generalization.

C) Model Architectures

Multiple architectures were explored, each employing different approaches:

1. **3D CNN:**
    - Utilized 3D convolutions to capture both spatial and temporal features from video frames.
    - These models were relatively simple yet effective for extracting spatial features.
    - Various filter sizes and resolutions were tested to optimize performance.

2. **CNN-LSTM:**
    - Implemented a CNN for feature extraction from individual frames.
    - An LSTM was applied on top of the CNN outputs to capture temporal dependencies between frames.
    - This hybrid approach effectively leveraged both spatial and temporal aspects of the gesture videos.

D) Experiments and Results

The following is a summary of the models discussed in the gesture recognition case study:

Model	Architecture	Key Features	Parameters	Training Accuracy	Validation Accuracy
ModelConv3D1	Conv3D	Multiple Conv3D layers, ReLU activation, BatchNormalization, MaxPooling3D, Dense	1,736,389	33.16%	21%
ModelConv3D2	Conv3D	Similar to ModelConv3D1 with added dropout layers	1,117,061	77.67%	23%
ModelConv3D3	Conv3D	Reduced filter size and image resolution	1,762,613	76.77%	22%
ModelConv3D4	Conv3D	Added more layers	2,556,533	74.35%	37%
ModelConv3D6	Conv3D	Reduced number of parameters	696,645	77.03%	27%
RNNCNN1	CNN + LSTM	TimeDistributed Conv2D layers, LSTM, Dense layers	1,657,445	94.66%	79%
ModelConv3D13	Conv3D	Similar to previous Conv3D models with data augmentation	696,645	77.45%	65%
ModelConv3D14	Conv3D	Reduced network parameters with data augmentation	504,709	78.17%	71%
RNNCNN2	CNN + GRU	CNN + GRU with data augmentation	2,573,925	98.43%	76%
RNNCNN_TL2	Transfer Learning	MobileNet + GRU, training all weights	3,693,253	99.32%	92%

1. ModelConv3D1

- Architecture: 3D Convolutional Neural Network (Conv3D)
- Parameters: 1,736,389
- Performance: Training Accuracy: 33.16%, Validation Accuracy: 21%

2. ModelConv3D2

- Architecture: Similar to ModelConv3D1 with added dropout layers.
- Parameters: 1,117,061
- Performance: Training Accuracy: 77.67%, Validation Accuracy: 23%

3. ModelConv3D3

- Architecture: Reduced filter size and image resolution.
- Parameters: 1,762,613
- Performance: Training Accuracy: 76.77%, Validation Accuracy: 22%

4. ModelConv3D4

- Architecture: Added more layers.
- Parameters: 2,556,533
- Performance: Training Accuracy: 74.35%, Validation Accuracy: 37%

5. ModelConv3D6

- Architecture: Reduced number of parameters.
- Parameters: 696,645
- Performance: Training Accuracy: 77.03%, Validation Accuracy: 27%

6. RNNCNN1

- Architecture: CNN + LSTM
- Layers: Time Distributed Conv2D layers followed by LSTM and Dense layers.
- Parameters: 1,657,445
- Performance: Training Accuracy: 94.66%, Validation Accuracy: 79%

7. ModelConv3D13

- Architecture: Similar to previous Conv3D models with data augmentation.
- Parameters: 696,645
- Performance: Training Accuracy: 77.45%, Validation Accuracy: 65%

8. ModelConv3D14

- Architecture: Reduced network parameters with data augmentation.
- Parameters: 504,709
- Performance: Training Accuracy: 78.17%, Validation Accuracy: 71%

9. RNNCNN2

- Architecture: CNN + GRU with data augmentation.
- Parameters: 2,573,925
- Performance: Training Accuracy: 98.43%, Validation Accuracy: 76%

10. RNNCNN_TL2

- Architecture: Transfer Learning with MobileNet and GRU.
- Parameters: 3,693,253
- Performance: Training Accuracy: 99.32%, Validation Accuracy: 92%

E) Final Model Selection

Chosen Model: RNNCNN1 (CNN + LSTM)
Reason: This model provided balanced performance, demonstrating a good trade-off between training and validation accuracy, with a manageable number of parameters.
Performance: Training Accuracy: 94.66%, Validation Accuracy: 79%

F) Key Learnings and Challenges

1. Overfitting: Overfitting presented a significant challenge in several models, particularly the 3D CNN models, which struggled to generalize beyond the training

set. Techniques such as dropout regularization and data augmentation were beneficial, but further efforts are needed to bridge the training-validation accuracy gap.

2. Strength of CNN-LSTM: The CNN-LSTM hybrid architecture proved to be the most effective. It successfully captured both spatial and temporal dependencies, which are crucial for gesture recognition tasks. The combination of CNN for spatial feature extraction and LSTM for temporal learning enhanced the model's performance on the validation set.

3. Computational Complexity: Higher image resolutions (160x160) and larger filter sizes (3,3,3) increased computational costs, resulting in longer training times and greater memory usage. Striking a balance between model complexity and generalization was a key focus throughout the experiments.

G) Future Work

1. Further Regularization: Implementing stronger regularization techniques (e.g., dropout, L2 weight regularization) to further reduce the gap between training and validation accuracy.

2. Optimize Model Size: Future efforts could focus on optimizing the model architecture for a low-memory footprint by exploring smaller CNN architectures or employing pruning techniques to reduce the number of parameters while maintaining performance.

3. Ensemble Learning: Investigating ensemble learning techniques that combine multiple models (e.g., 3D CNNs and CNN-LSTM hybrids) to enhance performance across various gesture types.

4. Transfer Learning: Utilizing pre-trained models (e.g., pre-trained CNNs for feature extraction) may decrease the amount of training data required while improving performance.

5. Data Augmentation: Exploring additional data augmentation techniques, such as motion blur and brightness changes, could simulate more realistic variations in gesture videos, thus enhancing model robustness.