Gesture Recognition Project

Problem Statement

A smart tv manufactures wants to add gesture based controls to their TVs.

To start with, the following 5 gestures are planned to be undstood by the TV:

- Thumbs Up to increase volume
- Thumbs Down to decrease volume
- Left Swipe to move 10 seconds back
- Right Swipe to move 10 seconds ahead
- Open Palm (Stop) to pause

The hardware and software to capture and take action based on the gestures already exists with the manufacturer, our focus will be on `Recognising the Gestures`.

Data

- The data we have been provided with to train our model consists of images / frames taken in a sequence (videos that are already broken down into images) for various individuals showing the above mentioned hand gestures.
- The data is labelled with the different classes (gestures) that need to be identified.

Approach

To do this, we will be using `Deep Learning`. Specifically, we will be trying two approaches:

- Approach 1: 3D CNN Model
- Approach 2: A CNN + RNN Model

Experiments

S. No.	Model	Result	Reason	Decision
1	Conv3D	Runtime Exception during training	Generator has less batches than what the model expects while training	Send the correct input for ablation experiment by fixing the range so that model does not expect more batches than the generator is capable of yielding
2	Conv3D	Validation Accuracy: 20%	Model is not deep enough to learn	Change model configuration to improve accuracy
3	Conv3D	Model throws ValueError exception	The MaxPooling layer had a filter size of $2 \times 2 \times 2$ while the Conv3D layer just before it convolved the image to $1 \times 38 \times 38$ size due to which the dimensions were becoming negative and the model not was being accepted by keras.	Ensure that video frame dimensions and kernel sizes are considered carefully. Try with $1 \times 2 \times 2$ filter in Conv3D Layer.
4	Conv3D	Out of Memory Error when allocating Tensor	Too many parameters	Add pooling layers to reduce number of parameters
5	Conv3D	Kernel Died	batch size too large	reduce batch size
6	Conv3D	Out of Memory	GPU Ram is limited	disable GPU usage

S. No.	Model	Result	Reason	Decision
7	Conv3D	Validation accuracy 19%	using low number of epochs and large batch to reduce training time but score is impacted greatly	use a different architecture (stop Conv3D experiments)
8	CNN + RNN (ResNet+GRU)	Training Time too high	GPU was disabled	reduce batch size
9	CNN + RNN (ResNet+GRU)	Memory Error	Even just loading ResNet was consuming 3.2GB out of 4GB of GPU Ram	use different model for transfer learning
10	CNN + RNN (MobileNet)	ValueError	When setting `include_top=True` and loading `imagenet` weights, `input_shape` should be (224, 224, 3). Received: input_shape=(80, 80, 3)	load model without weights. Initialize weights externally.
10	CNN + RNN (MobileNet+GRU)	Memory Error	Even just loading MobileNet was consuming 3GB out of 4 GB of GPU Ram	disable GPU, use CPU+Main Memory only
11	CNN + RNN (MobileNet+GRU)	Validation Accuracy - 87%	Feature extraction is significantly better with pre-trained models (transfer learning). GRUs are good at understanding sequential data.	Increase number of epochs to get better accuracy
12	CNN + RNN (MobileNet+GRU)	Validation Accuracy - 92%	epochs were increased and batch size was increased to 50	Try freezing layers to improve training time
13	CNN + RNN (MobileNet+GRU)	Validation Accuracy - 70%	the pre-trained layers of MobileNet	unfreeze the pre-trained layers and improve model architecture

S. No.	Model	Result	Reason	Decision
14	CNN + RNN (MobileNet+GRU)	Validation Accuracy - 93%	MobileNet layers were trained. CNN Layers were added after MobileNet Layers. Batch Normalization and Dropouts were used	Increase batch size to improve generalizability
15	CNN + RNN (MobileNet+GRU)	Validation Accuracy - 92%	architecture was similar to previous one, only batch size was increased to 200 for faster training	final model as this has least validation loss

Result

- The model is performing well as far as validation accuracy is concerned.
- On the whole validation data, 92% accuracy is observed.
- The model size is around 40MB which is small enough for Smart TVs.
- The model has total `3,392,821` parameters, out of which `3,370,693` are trainable.