Problem Statement:

We need to develop a cool feature in the smart-TV that can recognize five different gestures performed by the user which will help users control the TV without using a remote.

The following table consists of the experiments done to build a model to predict the gestures from the given data set

Exp No	Model	Hyperparameter	Result	Decision + Explanation
1	**Conv3D**	**Batch Size = 40** **Epochs = 20** **Dense neurons=64, dropout=0.25**	**Total params: 1117061** **Train accuracy: 0.9653** **val_accuracy: 0.2600**	**Model is clearly overfitting.** **Adding dropout layers**
2	**Conv3D**	**Batch Size = 20** **Epochs = 25** **dense_neurons=64, dropout=0.5** **image size 160*160**	**Total params: 1117061** **Train accuracy: 0.63** **val_accuracy: 0.2300**	**val_loss did not improve from 2.57189** **Early stopping at 11epoch** **Reduce the filter size and image resolution minor oscillations in loss, so lowering the learning rate to 0.0002**
3	**Conv3D**	**filter size (2,2,2)** **image 120 x 120,** **Batch Size = 30** **Epochs = 25** **Learning rate =0.0002**	**Total params: 1762613** **Train accuracy: 0.6538** **val_accuracy: 0.2200**	**val_loss did not improve from 1.86547** **Early stopping at 11epoch** **Adding more layers**
4	**Conv3D**	**Batch Size = 20** **No. of Epochs = 25** **dense_neurons=256** **dropout=0.5**	**Total params: 2556533** **Train accuracy: 0.8801** **val_accuracy: 0.7100**	**Let's try adding dropouts at the convolution layers**
5	**Conv3D**	**Batch Size = 20** **Epoch=15** **filtersize=(3,3,3)** **dense_neurons=256** **dropout=0.25**	**Total params: 2556533** **Train accuracy: 0.7836** **val_accuracy: 0.21**	**model is still Overfitting** **reduce the model size and see the performance**
6	**Conv3D**	**Batch size=20** **epochs=20** **dense neurons=128** **dropout=0.25**	**Total params: 696,645** **Train accuracy: 0.7813** **val_accuracy: 0.30**	**This low memory/compact model records a validation accuracy of 30%.** **val_loss did not improve from 2.12475** **Early stopping at 11epoch** **Reducing the number of parameters**
7	**Conv3D**	**Batch_size=20** **num_epochs=25** **dense_neurons=64** **dropout=0.25**	**Total params: 504,709** **Train accuracy: 0.75** **val_accuracy: 0.27**	**val_loss did not improve from 1.85189** **Lets us try new approach of using conv2d with LSTM/GRU models**
8	**Conv2D** **CNN- LSTM**	**batch_size=20** **epochs=20** **lstm_cells=128**	**Total params: 1,657,445**	**Comapre to other model till now this seems better**

		dense_neurons=128
		dropout=0.25
			Train accuracy: 0.94
			val_accuracy: 0.78
				Lets try some other models before finalizing
				lets augment the data with **slight rotation** as well and run the same set of models again

9	Conv3D with Augmentaion (similar to Model 2)	(3,3,3) Filter & 160x160 Image resolution	Total params: 3,638,981 Train accuracy: 0.78 val_accuracy: 0.30	Model overfitting Changing parameter
10	Conv3D with Augmentation (similar to Model 3)	(2,2,2) Filter 120x120 Image resolution	Total params: 1,762,613 Train accuracy: 0.66 val_accuracy: 0.38	Model overfitting Changing parameter Adding more layers

11	Conv3D with Augmentation (Similar to model 4)	batch_size=20,
		num_epochs=2
		filtersize=3,3,3
		dense_neurons=256
		dropout=0.5
			Total params: 2,556,533
			Train accuracy: 0.74
			val_accuracy: 0.78
				• Overfitting solved
				• But accuracy needs to improve
				• Adding dropouts to more layers

12	Conv3D with Augmentation (Similar to Model 5)	batch_size=20,
		epochs=25
		filtersize= 3,3,3
		dense_neurons=256
		dropout=0.25
			Total params: 2,556,533
			Train accuracy: 0.62
			val_accuracy: 0.22
				Badly overfitting
				Reducing network parameters

13	Conv3D with Augmentation (Similar to Model 6)	batch_size=20,
		epochs=25
		filtersize= 3,3,3
		dense_neurons=128
		dropout=0.25
			Total params: 2,556,533
			Train accuracy: 0.78
			val_accuracy: 0.66
				Reducing network parameters again

14	Conv3D with Augmentation (Similar to Model 7)	batch_size=20,
		epochs=30
		filtersize= 3,3,3
		dense_neurons=64
		dropout=0.25
			Total params: 504,709
			Train accuracy: 0.78
			val_accuracy: 0.73
				Comapre to other model till now this seems better
				Lets try some other models before finalizing

15	CNN LSTM with GRU
	(using Augmentation)
		batch_size=20
		num_epochs=20
		lstm_cells=128
		dense_neurons=128
		dropout=0.25
			Total params: 2,573,925
			Train accuracy: 0.92
			val_accuracy: 0.74
				overfitting

16	Transfer Learning Mobilenet+LSTM	batch_size=5
		num_epochs=20
		lstm_cells=128
		dense_neurons=128
		dropout=0.25
			Total params: 3,840,453
			Train accuracy: 0.97
			val_accuracy: 0.75
				overfitting

17	Transfer Learning Mobilenet+GRU	batch_size=5
		num_epochs=20
		lstm_cells=128
		dense_neurons=128
		dropout=0.25
			Total params: 3,693,253
			Train accuracy: 0.9805
			val_accuracy: 0.9500
				shows better performance in terms of accuracy, but has a higher parameter count, indicating a more complex model.

Conclusion:

Final model:

Model 17: Transfer Learning with GRU (Training All Weights)

- **Training Accuracy**: 98%
- **Validation Accuracy**: 95%
- **Total Parameters**: 3,693,253

Model 17 demonstrates excellent accuracy in both training and validation, achieving high performance with a considerable parameter count. The use of transfer learning with GRU while training all weights contributes to its strong results.