Gesture Recognition Project

Problem statement:

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

This table provides the details about different experiments performed during the project with model architecture details, hyperparameters details, results we got from each experiment and the decision + Explanations.

Exp. No	Model	Hyperparameters	Result	Decision + Explanations
1	Conv3D model with below structure: Conv3D layer (32 kernels) MaxPooling3D layer Conv3D layer(64 kernels MaxPooling3D layer Dense layer(256 nodes) Dense Layer(5 nodes)	batch_size = 64 num_images = 15 image_height = 120 image_width = 120 num_epochs = 10 num_train_sequences = 200	loss: 0.5087 categorical_accuracy: 0.8203 val_loss: 1.1403 val_categorical_accuracy: 0.5900	Trained with a subset of data (200 training sequence) and 10 epochs. Used only 200 sequences to see whether the generator and model setup works or not. The model is overfitting and needs more epochs and training data.
2	Conv3D model with below structure: Conv3D layer (32 kernels) MaxPooling3D layer Conv3D layer(64 kernels MaxPooling3D layer Dense layer(256 nodes) Dense Layer(5 nodes)	batch_size = 64 num_images = 15 image_height = 120 image_width = 120 num_epochs = 20 num_train_sequences = 663	loss: 6.3044e-04 categorical_accuracy: 1.0000 val_loss: 1.2433 val_categorical_accuracy: 0.7900	Increased the number of epochs to 20. Can clearly see that the model overfit as the training accuracy went to 1.000 but the val accuracy got stagnant at 0.77 – 0.80. Also, average time taken for an epoch training is around 100s.
3	Conv3D model with below structure: Conv3D layer (32 kernels) MaxPooling3D layer Conv3D layer(64 kernels MaxPooling3D layer Conv3D layer(128 kernels MaxPooling3D layer Dense layer(256 nodes) Dropout(0.25) Dense Layer(5 nodes)	batch_size = 64 num_images = 15 image_height = 120 image_width = 120 num_epochs = 15 num_train_sequences = 663 Num of training parameters = 7,653,637	Loss = 0.1157 Categorical-accuracy: 0.9593 Val_loss = 0.3779 Val_categorical_accuracy = 0.8400	Added additional conv3D layer with 128 kernel and a dropout layer after first dense layer. The val loss decreased and val_categorical_accuracy increased. Better than previous models but still a little overfitting.

4	Conv3D model with below structure: Conv3D layer (32 kernels) MaxPooling3D layer Dropout(0.3) Conv3D layer(64 kernels MaxPooling3D layer Dropout(0.3) Conv3D layer(128 kernels MaxPooling3D layer Dropout(0.3) Dense layer(256 nodes) Dropout(0.25) Dense Layer(5 nodes)	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 10 num_train_sequences = 663 Num of training parameters = 7,653,637	Loss = 0.1373 Categorical-accuracy = 0.9548 Val_loss = 0.4390 val_categorical_accuracy: 0.8700	Added dropouts after every conv3D layer to reduce overfitting and decreased batch size to 32. val_categorical_accuracy has increased to 87% and training accuracy to 95% showing better model.
5	Conv3D model with below structure: Conv3D layer (32 kernels) MaxPooling3D layer BatchNormalization layer Dropout(0.3) Conv3D layer(64 kernels MaxPooling3D layer BatchNormalization layer Dropout(0.3) Conv3D layer(128 kernels MaxPooling3D layer BatchNormalization layer Dropout(0.3) Dense layer(256 nodes) Dropout(0.25) Dense Layer(5 nodes)	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 10 num_train_sequences = 663 Num of training parameters = 7,653,637	Loss = 0.0572 categorical_accuracy: 0.9744 val_loss: 20.8559 val_categorical_accuracy: 0.1300	Added Batch normalization layer with all layers to see the behavior. Model completely overfitting. The val accuracy went down to 0.13 and val loss to 20.85 which is too high.
6	Conv3D model with below structure: Conv3D layer (32 kernels) MaxPooling3D layer Dropout(0.3) Conv3D layer(64 kernels MaxPooling3D layer Dropout(0.3) Conv3D layer(128 kernels MaxPooling3D layer Dropout(0.3) GlobalAveragePooling3D Dense layer(256 nodes) Dropout(0.25) Dense Layer(5 nodes)	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 20 num_train_sequences = 663 Num of training parameters = 313,605	loss: 0.5065 categorical_accuracy: 0.8145 val_loss: 0.4590 val_categorical_accuracy: 0.8200	Used GlobalAveragePooling3D layer instead of Flatten layer and observed the behavior. There is no overfitting as the training_accuracy and val accuracy are almost same.

7	Conv3D model with below structure: Conv3D layer (32 kernels) MaxPooling3D layer Dropout(0.3) Conv3D layer(64 kernels MaxPooling3D layer Dropout(0.3) Conv3D layer(128 kernels MaxPooling3D layer Dropout(0.3) GlobalAveragePooling3D Dense layer(256 nodes) Dropout(0.25) Dense Layer(5 nodes)	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 30 num_train_sequences = 663 Num of training parameters = 313,605	loss: 0.2617 categorical_accuracy: 0.9035 val_loss: 0.3087 val_categorical_accuracy: 0.8700	Used number of epochs as 30 and using GlobalAveragePooling3D layer. The model does not overfit and the val loss decreased to 0.3087. Also, the parameters are just 313,605
8	Con2D + GRU structure: TimeDistributed(conv2D)32 kernels. TimeDistributed(MaxPooling2D((2,2) BatchNormalization Dropout TimeDistributed(conv2D)64 kernels. TimeDistributed(MaxPooling2D((2,2) BatchNormalization Dropout GlobalAveragePooling2D GRU BatchNormalization Dropout	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 10 num_train_sequences = 663 Num of training parameters = 99,845	loss: 0.8066 categorical_accuracy: 0.6848 val_loss: 1.8142 val_categorical_accuracy: 0.2000	Used GRU + Conv2D. The result is overfitting. Although the number of paramaters decreased but the val loss increases to 1.81 and val accuracy moved down to 0.2. Trying to remove batchNormalization and also using GlobalAveragePooling3D in next experiment.
9	Conv2D + Dense TimeDistributed(conv2D)32 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout TimeDistributed(conv2D)64 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout TimeDistributed(conv2D)64 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 10 num_train_sequences = 663 Num of training parameters = 127,557	loss: 1.3933 categorical_accuracy: 0.3590 val_loss: 1.3233 val_categorical_accuracy: 0.4200	Model is underfitting. Used Conv2D with a dense layer only and no GRU. Need to train for more epochs. Will train for 30 epochs now.

	TimeDistributed(conv2D)128 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout GlobalAveragePooling3D			
10	Conv2D + Dense TimeDistributed(conv2D)32 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout TimeDistributed(conv2D)64 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout TimeDistributed(conv2D)64 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout TimeDistributed(conv2D)128 kernels. TimeDistributed(MaxPooling2D((2,2) Dropout GlobalAveragePooling3D	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 30 num_train_sequences = 663 Num of training parameters = 127,557	loss: 1.0058 categorical_accuracy: 0.5716 val_loss: 0.9679 val_categorical_accuracy: 0.5600	Model not giving better accuracy even after 30 epochs. Although, the model is not overfitting but still the model training will take a lot of time to converge. Hence, we will not use this model.
11	Conv2D + Conv2DLSTM + GlobalAveragePooling2D TimeDistributed(conv2D)32 kernels TimeDistributed(conv2D)32 kernels ConvLSTM2D (8 kernels) TimeDistributed(Dense)64 kernels GlobalAveragePooling2D	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 40 num_train_sequences = 663 Num of training parameters = 13,397	loss: 1.3517 categorical_accuracy: 0.3876 val_loss: 1.3254 val_categorical_accuracy: 0.4200	Model does not learn here as well even after 40 epochs. Will try creating another Conv2D layer and increasing the kernel size of the layers.
12	Conv2D + Conv2DLSTM + Flatten TimeDistributed(conv2D)32 kernels Max pooling TimeDistributed(conv2D)64 kernels Max pooling TimeDistributed(conv2D)128 kernels	batch_size = 32 num_images = 15 image_height = 120 image_width = 120 num_epochs = 20 num_train_sequences = 663	Loss: 0.0104 categorical_accuracy: 0.9970 val_loss: 1.0951 val_categorical_accuracy: 0.8000	The number of parameters were increased and also the model started overfitting. Validation loss increased and also there is a difference

	Max pooling ConvLSTM2D (128Kernels) TimeDistributed(Dense)64 kernels Flatten Layer Dense layers	Num of training parameters = 1,777,669		between train accuracy and validation accuracy.
13	**Conv3D model with below structure:** **Conv3D layer (32 kernels)** **MaxPooling3D layer** **Dropout(0.3)** **Conv3D layer(64 kernels** **MaxPooling3D layer** **Dropout(0.3)** **Conv3D layer(128 kernels** **MaxPooling3D layer** **Dropout(0.3)** **GlobalAveragePooling3D** **Dense layer(256 nodes)** **Dropout(0.25)** **Dense Layer(5 nodes)**	**batch_size = 32** **num_images = 20** **image_height = 120** **image_width = 120** **num_epochs = 40** **num_train_sequences = 663** **Num of training parameters = 313,605**	**loss: 0.2606** **categorical_accuracy: 0.9035** **val_loss: 0.3139** **val_categorical_accuracy: 0.8900**	**As experiment 7 gave the most promising results. Now training it with 40 epochs and 20 image frames.**

Result:

For best model we got in experiment 13,

- Training categorical loss = 0.2606

- Training categorical accuracy = 0.9035

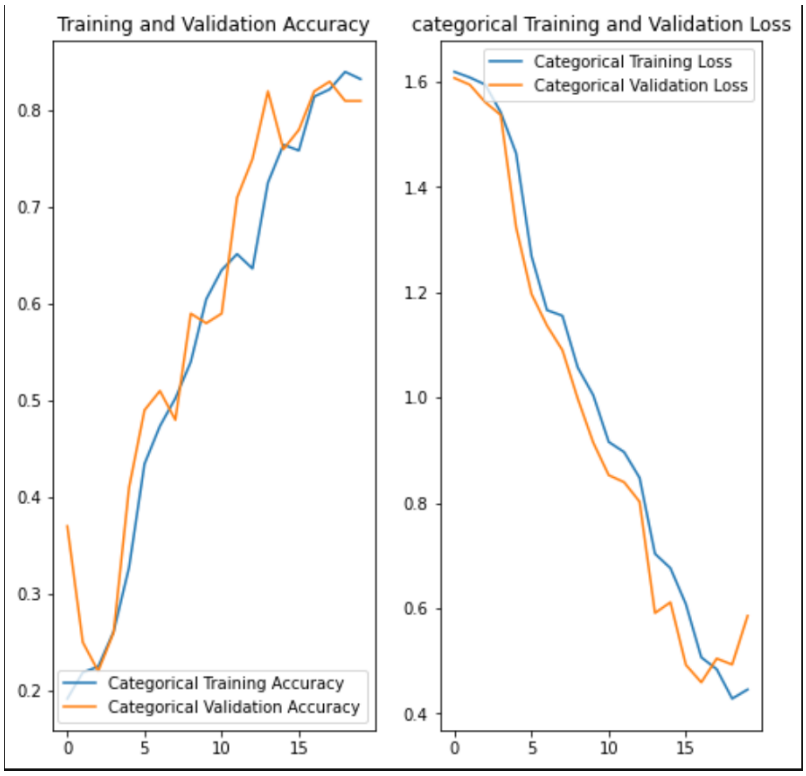
- Validation categorical loss = 0.3139

- Validation categorical accuracy: 0.8900

The model is not overfitting as both the training and validation accuracy is similar. Also, the model is having less validation loss from all other models. The number of trainable parameters are 313,605.

Best model I got is when using Conv3D + MaxPooling + Dopout + GlobalAveragePooling3D layers

The plots are as follows:


Training and Validation Accuracy
Categorical Training Accuracy
Categorical Validation Accuracy
categorical Training and Validation Loss
Categorical Training Loss
Categorical Validation Loss