Gesture Recognition – Deep learning

Problem Statement:

We need 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.

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

Exp. #	Model	Hyper-parameters	Result	Decision + Explanation
1	Conv3D	Batch size = 128, Ablation = 20, Augmentation = False, LR = 0.01, Seq Length = 10, Epoch = 20, Dim = 120x120	Train Accuracy: 0.15, Validation Accuracy: 0.15	The Model is not learning anything throughout the epochs, the loss is not decreasing. Reducing the batch size further.
2	Conv3D	Batch size = 32	Train Accuracy: 0.15, Validation Accuracy: 0.20	No improvement in the model, lets add more layers to the model so that it can learn from data.
3	Conv3D		Negative Dimension Error.	The new CNN kernel sizes are not compatible with the output of previous layers. Let's reduce the kernel size of new layers.
4	Conv3D		Train Accuracy: 0.20, Validation Accuracy: 0.20	Still there is no improvement in the model. Let's add Batch normalization layers after every CNN and dense layers.
5	Conv3D		Train Accuracy: 0.9062, Validation Accuracy: 0.2708	Model is able to over-fit on less data (Ablation data set), Let's Training on full data and increasing epochs to 50.
6	Conv3D	Ablation = None, Epoch = 50	Train Accuracy: 0.9062, Validation Accuracy: 0.70	Mode is having over-fitting as there is huge gap between training and validation accuracies. Let's add some dropouts that the model can be generalized.
7	Conv3D	Dropout = 0.2	Train Accuracy: 0.9896, Validation Accuracy: 0.7734	There is a bit of increase in the model validation accuracy and training accuracy also. Lets increase the drop out values from 0.2 to 0.5

8	**Conv3D**	**Dropout = 0.5**	**Train Accuracy: 0.9777, Validation Accuracy: 0.5391**	**After increase the dropout the model validation score further reduced and the model is over-fitted. Let's use 0.2 only remove a CNN layer to reduce the complexity of the model.**
9	**Conv3D**	**Dropout = 0.2**	**Train Accuracy: 1.00, Validation Accuracy: 0.77**	**Still the model is over-fitting. Let's use a Global Average Pooling instead of Flatten Layer.**
10	**Conv3D**		**Train Accuracy: 0.9509, Validation Accuracy: 0. 9062**	**The model is wonderful and the training and validation scores are good. The model has 710,533 trainable parameters. Let's try architectures too.**
11	**Time Distributed + GRU**		**Train Accuracy: 0.9554, Validation Accuracy: 0. 8203**	**The model is working quite well on validation dataset with less trainable parameters(98,885), Lets add some drop outs after each layer, so that both train and validation accuracies will be closure.**
12	**Time Distributed + GRU**	**Drop out = 0.2**	**Train Accuracy: 0.8720, Validation Accuracy: 0.6016**	**The model accuracy further deteriorated; Let's replace GRU with a plain Dense Layer Network and some Global Avg Pooling.**
13	**Time Distributed + Dense**		**Train Accuracy: 0.8780, Validation Accuracy: 0.8750**	**This is good model with training and validation accuracies with number of params 128,517. Let's use different architecture of model with time distributed and ConvLSTM2D.**
14	**Time Distributed + ConvLSTM 2D**		**Train Accuracy: 0.9673, Validation Accuracy: 0.9375**	**This is the best model so far we can get. The validation accuracy is good and the numbers of parameters are 13,589. The model size is also so small 226KB.**

Conclusion:

The Model built with Time distributed Conv2D and ConvLSTM2D (Experiment #14) gave better results compared to all the other models and also the model has very least number of parameters compared to other models.