Write-up of building the model for gesture recognition

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 below table shows the different model experimented for developing the above feature

Exp. No.	Model Architecture	Hyper parameters	Augmented	Metrics				Change in model + Observation
				Train Accuracy	Val Accuracy	Train Loss	Val Loss	
1	Conv 3D	Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	False	1	0.62	0.017	1.53	• Let's start with a simple network. • The training loss seems to be decreasing whereas validation loss is increasing after every epoch. • This clearly shows the model is not good and it is overfitting.
2			False	1	0.70	0.008	1.1375	• There are many ways to remove overfitting. Before that, let's deepen the model and observe the behaviour. • The training loss seems to be decreasing whereas validation loss is increasing after every epoch. • This clearly shows the model is still not good and it overfitting. • Though the model is overfitting, the metrics are better than model_1. Hence, let's improvise on this model_2.
3			True	0.97511	0.8	0.08835	0.84911	• Let's augment the training data to reduce overfitting on model_2. • The training loss seems to be decreasing and is almost 0 whereas validation loss is increasing after every epoch. • Though the metrics are high, the validation loss is increasing after every epoch and it is overfitting.
4			False	0.98944	0.75	0.0388	1.10293	• Let's add dropout layers to model_2 and see how it handles overfitting. • The training loss seems to be decreasing whereas validation loss

								is increasing after every epoch. • Though the metrics are high, the validation loss is increasing after every epoch and it is overfitting. • Though the model is overfitting, the gap between training and validation metrics got reduced which shows dropout layers has reduced the overfitting.
5		Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	True	0.99736	0.77	0.01784	1.41398	• Let's fuse model_3 and model_4, i.e., add dropout layers as well as augment the training data. • The training loss seems to be decreasing whereas validation loss is increasing after every epoch. • Though the model is overfitting, the gap between training and validation metrics got reduced. • This model is showing metrics similar to model_4.
6	**Conv 3D**		False	0.97285	0.31	0.11892	2.4139	• We observed that, with dropout layers or with both data augmentation and dropout layers, the metrics and behaviour of the model is very similar. • Let's improvise on model with dropout layers ,i.e. model_4, since it takes less training time and we get similar results. • Dropout layers, model_4 = ~625s (25 epochs * 25 s/epoch) • Dropout layers + Data augmentation , model_5 = ~960s (16 epochs * 60 s/epoch) • Let's add batch normalization layers to see if it improves the overall performance. • This model is overfitting a lot.
7			False	0.997	0.32	0.0406	2.0948	• Let's remove some dropout layer and batch normalization layer from model_6 and experiment. • This model is overfitting a lot.

8	**Conv 3D**	Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	False	0.7315	0.63	0.6465	0.9315	• Let's replace Flatten layer with GlobalAveragePooling3D layer to see the performance improvement in model_4. • Both the training loss and validation loss seems to be decreasing after every epoch. • Both training and validation metrics are overlapping. • The model metrics are low, but the model shows significant reduction in overfitting. This shows that global average pooling layer has significantly reduced the overfitting. • **model_8 could be considered as good model. Hence, Let's improvise on model_8.**
9			False	0.7496	0.25	0.6192	1.716	• Let's add batch normalization layers to model_8 to see the performance improvement. • This model is overfitting a lot.
10			False	0.8567	0.24	0.4005	2.5466	• Let's add more conv layers to model_9 to see the performance improvement. • To deepen the network, need to adjust the pooling layers to accomodate the depth. • This model is overfitting a lot.
11			True	0.60897	0.56	0.83352	1.00607	• Let's add augmented training data to model_8 and see how it improves the performance. • Both the training loss and validation loss seems to be decreasing after every epoch. • This model metrics are good, but the data augmentation did not improve the performance since we got better results without data augmentation in model_8.

- In Conv3D architecure based networks, we can consider **model_8** to be performing good with **87,429** total parameters and weights file size of **1090 KB**.

12	**Conv2D + RNN**	Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	False	0.65762	0.72	0.71656	0.70857	• Let’s start with simple RNN model with LSTM cell. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model is slightly underfitting.
13			False	0.69985	0.67	0.62993	0.75315	• Let's add more conv units to model_12 to see if we can reduce underfitting. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model metrics are low, but the model shows reduction in underfitting. This shows that additional layer have worked. • model_13 could be considered as good model. Hence, Let's improvise on model_13.
14			False	0.825	0.28	0.4449	2.4414	• Let's add batch normalization layers in model_12 to see if there is any performance improvement. • The model is overfitting a lot.
15			True	0.84804	0.82	0.34843	0.48902	• Let's add augmented training data on model_13 to see if there is any performance improvement. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model is showing good accuracy metrics with fewer parameters. • **model_15 could be considered as good model.**
16			False	0.6848	0.73	0.6399	0.6942	• Let's replace LSTM cell with GRU cell in model_13 to see if the model performs good with even fewer parameters. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model is slightly underfitting.

17	**Conv2D + RNN**	Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	False	0.4857	0.48	1.0527	1.0835	• Let's reduce model complexity of model_16 and see if it improves the performance. • Both the training loss and validation loss seems to be decreasing after every epoch. • The model is good but the metrics are poor.
18			False	0.5279	0.58	0.9748	1.0639	• Let's add another GRU layer in model_17 and see if it improves the performance. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model is slightly underfitting. • The model did not perform well despite adding another GRU layer.
19			False	0.4887	0.53	1.0873	1.0566	• Let's add ConvLSTM2D layer and see the performance. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model is slightly underfitting.
20			False	0.45551	0.43	1.16585	1.2072	• Let's deepen model_19 with dense layers to see if we can overcome underfitting. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model has overcome underfitting issue, but has poor metrics still.
21			False	0.727	0.32	0.7256	1.7764	• Let's add batch normalization layers to model_20 and see if there is any performance improvement. • The model is overfitting a lot.
22			False	0.71192	0.7	0.6831	0.7096	• Let's remove the max pool layers in model_21 and see if it improves the performance. • Both the training loss and validation loss seems to be decreasing after every epoch. • The model is showing good accuracy metrics with fewer parameters.

23	**Conv2D + RNN**	Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	True	0.75603	0.75	0.6286	0.67698	• Let's add augmented training data to model_22 and see if it improves performance. • Both the training loss and validation loss seems to be decreasing after every epoch and closely overlapping. • The model is showing good accuracy metrics with very few parameters. • **model_23 could be considered as good model.**
24			False	0.7828	0.43	0.579	1.4493	• Let's tweak the network and see if it improves the performance. • Let's make the padding as 'valid'. • The model is overfitting a lot.
In Conv2D with RNN architecure based networks, we can consider below models to be performing good • **model_15** with **225,477** total parameters and weights file size of **2700 KB**. • **model_23** with **13,781** total parameters and weights file size of **254 KB**.								
25	**Transfer Learning + RNN**	Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	False	1	0.83	0.0066	0.59641	• Let's experiment on mobile net V2 for feature extraction and LSTM cell. • Training loss is decreasing but validation loss seems to be flattening and slightly increasing after few epochs. • Though the training accuracy is 1, the validation accuracy is 0.83. This shows the model is overfitting.
26			False	0.98944	0.81	0.03743	0.56922	• Let's add dropout layers to model_25 and see if we can overcome overfitting. • Training loss is decreasing but validation loss seems to be flattening and slightly increasing after few epochs. • Though the model is overfitting, the gap seems to have slightly reduced than the previous model.
27			False	1	0.85	0.00053	0.4477	• Let's add batch normalization layer to model_26 and check the performance improvement. • Let's add max pooling layer to model_26 and see if we can reduce overfitting. • Both the training loss and validation loss

	Transfer Learning + RNN							seems to be decreasing after every epoch. • Though the model is overfitting, the gap seems to have slightly reduced than the previous model.
28		Batch size = 32 Frame sequence = alternate frames from 5 to 25 (10 frames) Image size = (120,120)	False	1	0.88	0.00159	0.34987	• Let's replace LSTM cell with GRU cell in model_27 to see if we get similar performance with lesser parameters. • Both the training loss and validation loss seems to be decreasing after every epoch. • The change in metrics is very small after few epochs, this might be due to the saturation in learning as model was loaded with pre-trained weights. • The model shows overall good metrics. • **model_28 could be considered as good model.**
29			True	0.99284	0.81	0.02506	0.5734	• The previous model is overall good, but let's still experiment on model_28 with augmented training data. • Training loss is decreasing and validation loss is increasing after every epoch. • The model seems to be overfitting.

- In Transfer learning + RNN based networks, we can consider **model_28** to be performing good with **3,511,365** total parameters and weights file size of **23.3 MB**.

Additional experiments on input parameters:

Till now, we were modifying architecture to get good results.
Let's now take the best models we got till now, and play around input parameters and see if we can get still better results.

30	**Conv 3D**	Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (180,180)	False	0.6908	0.63	0.70205	0.73179	• Experiment on model_8 by increasing the number of frames to 20 and image resolution to (180,180). • The metrics are good, but it is not better than model_8.

31	**Conv 3D**	Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (120,120)	False	0.6787	0.62	0.7343	0.8293	• Experiment on model_8 by increasing the number of frames to 20. • The metrics are good, but it is not better than model_8.
32	**Conv2D + RNN**	Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (180,180)	False	0.67119	0.65	0.6526	0.66063	• Experiment on model_15 by increasing the number of frames to 20 and image resolution to (180,180). • The metrics are good, but it is not better than model_15.
33		Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (120,120)	False	0.67	0.7	0.6829	0.731	• Experiment on model_15 by increasing the number of frames to 20. • The model is slightly underfitting and it is not better than model_15.
34		Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (180,180)	False					• Experiment on model_23 by increasing the number of frames to 20 and image resolution to (180,180). • Facing resource OOM error due to limited availability.
35		Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (120,120)	False	0.7195	0.21	0.7535	2.9088	• Experiment on model_23 by increasing the number of frames to 20. • The model is overfitting a lot.

36	**Transfer Learning + RNN**	Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (180,180)	False	1	0.82	0.00825	0.57089	• Experiment on model_28 by increasing the number of frames to 20 and image dimensions to (180, 180). • The metrics are good, but it is not better than model_28.
37	**Transfer Learning + RNN**	Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (160,160)	False	1	0.89	0.00072	0.41466	• We could see there are some shapes of weight for which the model is pre-trained. Let's experiment on those shapes. • Experiment on model_28 by increasing the number of frames to 20 and image dimensions to 160 which is one of the standard imagenet weight input shape. • **The model is giving good results similar to model_28.**
38		Batch size = 32 Frame sequence = frames from 5 to 25 (20 frames) Image size = (128,128)	False	1	0.81	0.02165	0.67828	• Experiment on model_28 by increasing the number of frames to 20 and the image dimensions to 128 which is one of the standard imagenet weight input shape. • The metrics are good, but it is not better than model_28.

- From additional experiments, Experiment 37 is giving good results, but we can ignore this model, since model_28 with **lesser image resolution** and **lesser number of frames** was able to provide similar results.
- Hence, additional experiments did not give any better model.

Conclusion

From the above experiments and **38** different models built on different architectures, below are the good models from each variant,

Model	Architecture	Total Parameters	Weight file size	Training Accuracy	Validation Accuracy	Training Loss	Validation Loss
model_8	Conv3D	87,429	1090 KB	0.73152	0.63	0.64653	0.93146
model_15	Conv2D + RNN	225,477	2700 KB	0.84804	0.82	0.34843	0.48902
model_23	Conv2D + RNN	13,781	254 KB	0.75603	0.75	0.62860	0.67698
model_28	**Transfer learning + RNN**	**3,511,365**	**23.3 MB**	**1.0**	**0.88**	**0.00159**	**0.34987**

-> We can **ignore** model_8 as we have better models from other variants.

-> Each model is good in different use cases, as few shown below,

Memory Availability	Performance Needed	Suggested model
Abundant	High	model_28
Abundant	Moderate	model_28 or model_15
Moderate	Moderate	model_15
Less	Moderate	model_23

- For the assignment purpose, we can consider **model_28** with the above shown metrics as the **best** model.