GESTURE RECOGNITION SYSTEM

In this project we have to build a model which can classify five human action gestures.
Let's understand different actions
Thumbs up: Increase the volume
Thumbs down: Decrease the volume
Left swipe: 'Jump' backwards 10 seconds
Right swipe: 'Jump' forward 10 seconds
Stop: Pause the movie

Data

2 csv and 2 folder of data is provided,

- Val csv
- Train csv
- Train folder
- Val folder

Inside the train folder we have 663 folders each folder contains 30 sequential images.
Inside the val folder we have 100 folders, each folder contains 30 sequential images.
Train csv and val csv contain name of folders without action

Approach and Steps

This problem can be solved by conv3d or,CNN+LSTM,CNN+GRU,tf+LSTM,tf+GRU.
We have tried different model architecture and moved accordingly, In final model we are able to achieve 90 in training and 82 in validation, in 35 epochs.

We have tried a different number of frames,i.e 15,18,20,24,30.Higher number leads t OOM error
We have tried different number of batch,i.e 663,256,128,64,32
For above 32 batch , We got an OOM error. Then we decided to stick under 32.
Image size chosen 120x120 and 96x96
96x96 is chosen because in transfer learning we used MobileNetV2 and generally, any state of art model performs well when we feed images of size on which they are trained on.

For other models we have chosen 120x120 or160x160, we got 2 sizes of images 360x360 and 120x160, hence we decided to go with either 120x120 or 160x160,.

In the training dataset we cropped and then resized it to 120x120 or 160x160. If transfer learning is used then according to the model on which it was trained.
For the val dataset we resize all images to 120x120,160x160 and according to the transfer learning model.

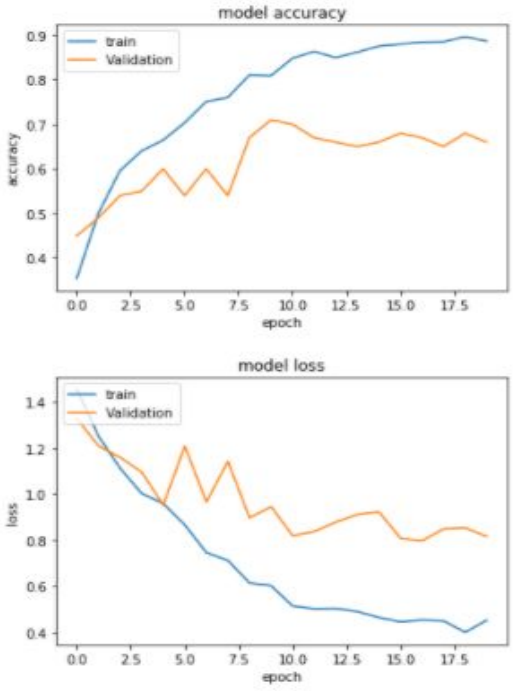
For normalisation,
We tried 2 type of normalization
Mean normalisation and dividing by 255. We didn't see much difference in base model performance. So we chose dividing by 255. To keep simple.

Optimizer we have chosen Adam.
Total of 3 of callbacks are used

- Model checkpoint
- ReduceLROnPlateau with patience 3 and monitor val loss
- EarlyStopping with patience 8 and monitor val loss.

CNN+LSTM Modelling

Base Model
3 time distributed convolution layer(16,32,64) neuron with (3,3) kernal size, relu activation.
Batch normalization after convolution
Then Maxpooling layer with pool size (2,2)
Flatten layer
Then LSTM layer 64 lstm cell
Then a dense layer with 64 neuron relu as activation.
Then dropout with .20

Layer (type) Output Shape Param # time_distributed_1 (TimeDist (None, 18, 120, 120, 18) 504 time_distributed_2 (TimeDist (None, 18, 120, 120, 18) 72 time_distributed_3 (TimeDist (None, 18, 60, 60, 18) 0 time_distributed_4 (TimeDist (None, 18, 60, 60, 32) 5216 time_distributed_5 (TimeDist (None, 18, 60, 60, 32) 128 time_distributed_6 (TimeDist (None, 18, 30, 30, 32) 0 time_distributed_7 (TimeDist (None, 18, 30, 30, 64) 18496 time_distributed_8 (TimeDist (None, 18, 30, 30, 64) 256 time_distributed_9 (TimeDist (None, 18, 15, 15, 64) 0 time_distributed_10 (TimeDis (None, 18, 14400) 0 lstm_1 (LSTM) (None, 64) 3703040 dense_1 (Dense) (None, 64) 4160 dropout_1 (Dropout) (None, 64) 0 dense_2 (Dense) (None, 5) 325 Total params: 3,732,197 Trainable params: 3,731,969 Non-trainable params: 228 None	

Trained model for 20 epochs
For the first 5 epoch we got good response val loss and train loss was decreasing
After 7 epoch we can see model start overfitting. Even after callbacks reduce LR. we didn't see any performance increase.

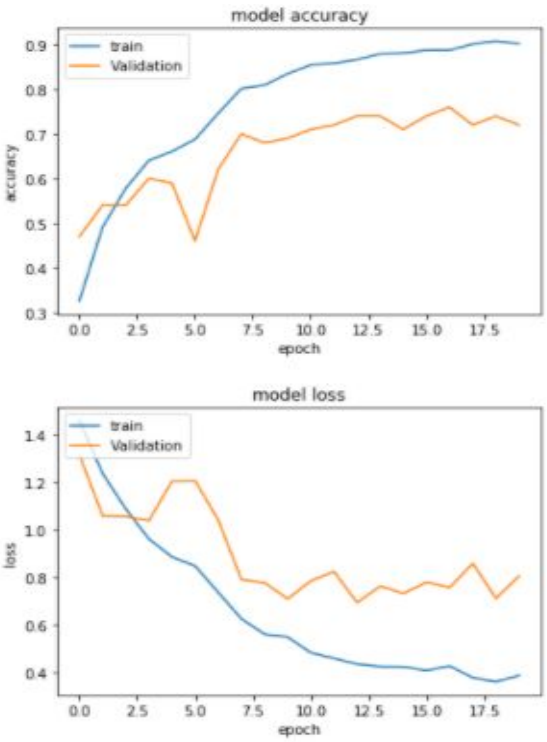
We decided to increase dropout from .20 to .25 and added one more dropout layer after LSTM
als we decided to add 1 more convolution layer with 128 neurons. kernel size (3,3)
LSTM cells also increased to 128 and dense layer neurons also to 128.

2nd Model CNN+LSTM

Layer (type)	Output Shape	Param #
time_distributed_1 (TimeDist	(None, 18, 120, 120, 18)	504
time_distributed_2 (TimeDist	(None, 18, 120, 120, 18)	72
time_distributed_3 (TimeDist	(None, 18, 60, 60, 18)	0
time_distributed_4 (TimeDist	(None, 18, 60, 60, 32)	5216
time_distributed_5 (TimeDist	(None, 18, 60, 60, 32)	128
time_distributed_6 (TimeDist	(None, 18, 30, 30, 32)	0
time_distributed_7 (TimeDist	(None, 18, 30, 30, 64)	18496
time_distributed_8 (TimeDist	(None, 18, 30, 30, 64)	256
time_distributed_9 (TimeDist	(None, 18, 15, 15, 64)	0
time_distributed_10 (TimeDis	(None, 18, 15, 15, 128)	73856
time_distributed_11 (TimeDis	(None, 18, 15, 15, 128)	512
time_distributed_12 (TimeDis	(None, 18, 7, 7, 128)	0
time_distributed_13 (TimeDis	(None, 18, 6272)	0
lstm_1 (LSTM)	(None, 128)	3277312
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16512
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 5)	645

Total params: 3,393,509
Trainable params: 3,393,025
Non-trainable params: 484

None



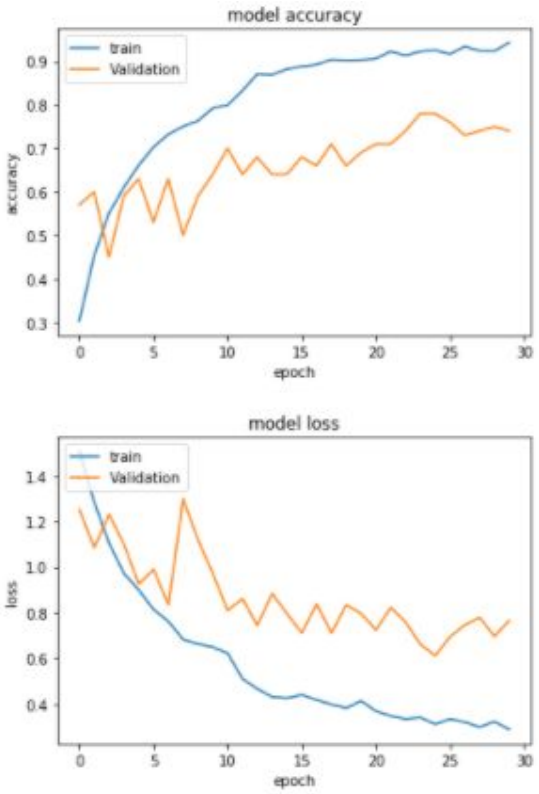
We are able to achieve 74 val accuracy and 87 train accuracy. We noticed that the model showed a tendency to learn.
We decided to increase epoch to 30 and also dropout .30,so we can reduce overfitting.

3rd model CNN+LSTM

Layer (type)	Output Shape	Param #
time_distributed_1 (TimeDist	(None, 18, 120, 120, 18)	504
time_distributed_2 (TimeDist	(None, 18, 120, 120, 18)	72
time_distributed_3 (TimeDist	(None, 18, 60, 60, 18)	0
time_distributed_4 (TimeDist	(None, 18, 60, 60, 32)	5216
time_distributed_5 (TimeDist	(None, 18, 60, 60, 32)	128
time_distributed_6 (TimeDist	(None, 18, 30, 30, 32)	0
time_distributed_7 (TimeDist	(None, 18, 30, 30, 64)	18496
time_distributed_8 (TimeDist	(None, 18, 30, 30, 64)	256
time_distributed_9 (TimeDist	(None, 18, 15, 15, 64)	0
time_distributed_10 (TimeDis	(None, 18, 15, 15, 128)	73856
time_distributed_11 (TimeDis	(None, 18, 15, 15, 128)	512
time_distributed_12 (TimeDis	(None, 18, 7, 7, 128)	0
time_distributed_13 (TimeDis	(None, 18, 6272)	0
lstm_1 (LSTM)	(None, 128)	3277312
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16512
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 5)	645

Total params: 3,393,509
Trainable params: 3,393,025
Non-trainable params: 484

None



We are able to achieve 78 in val with 92 in train data.
There is Improvement from the previous model.
We have decided one more convolution layer with kernel size=(3,3) with 256 neurons
Then batch normalization layer and max pooling layer with poolsize=(2,2).
The reason we decided to add convolution layer to extract more advanced features which can help in learning and again dropout is increased to .40.

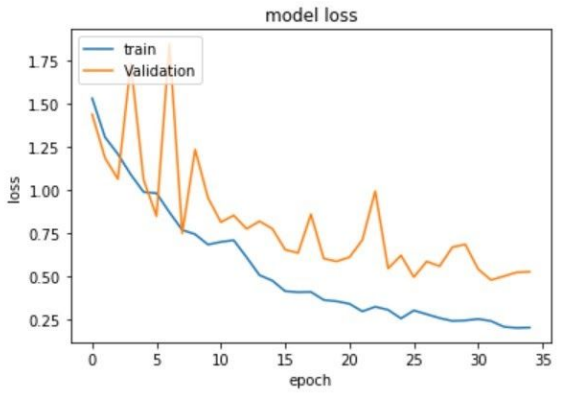
4th CNN+LSTM

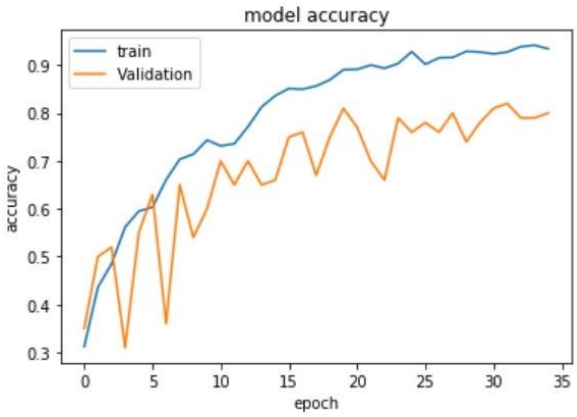
We are able to achieve 90 in train and 82 in val dataset. We can see it is a huge advancement from the previous.
Epoch is also increased to 35

Layer (type)	Output Shape	Param #
time_distributed_17 (TimeDis	(None, 18, 120, 120, 18)	504
time_distributed_18 (TimeDis	(None, 18, 120, 120, 18)	72
time_distributed_19 (TimeDis	(None, 18, 60, 60, 18)	0
time_distributed_20 (TimeDis	(None, 18, 60, 60, 32)	5216
time_distributed_21 (TimeDis	(None, 18, 60, 60, 32)	128
time_distributed_22 (TimeDis	(None, 18, 30, 30, 32)	0
time_distributed_23 (TimeDis	(None, 18, 30, 30, 64)	18496
time_distributed_24 (TimeDis	(None, 18, 30, 30, 64)	256
time_distributed_25 (TimeDis	(None, 18, 15, 15, 64)	0
time_distributed_26 (TimeDis	(None, 18, 15, 15, 128)	73856
time_distributed_27 (TimeDis	(None, 18, 15, 15, 128)	512
time_distributed_28 (TimeDis	(None, 18, 7, 7, 128)	0
time_distributed_29 (TimeDis	(None, 18, 7, 7, 256)	295168
time_distributed_30 (TimeDis	(None, 18, 7, 7, 256)	1024
time_distributed_31 (TimeDis	(None, 18, 3, 3, 256)	0
time_distributed_32 (TimeDis	(None, 18, 2304)	0
lstm_2 (LSTM)	(None, 128)	1245696
dropout_3 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 128)	16512
dropout_4 (Dropout)	(None, 128)	0
dense_4 (Dense)	(None, 5)	645

Total params: 1,658,085
Trainable params: 1,657,089
Non-trainable params: 996

None





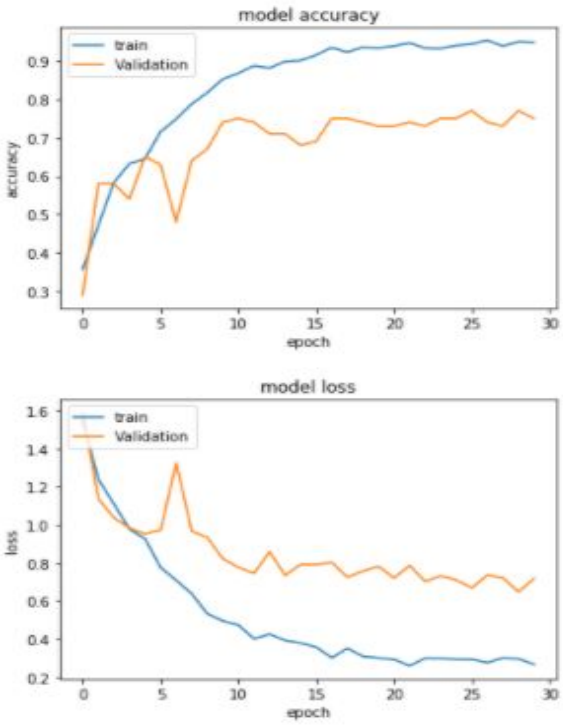
5th CNN+GRU

In CNN+LSTM we are able to achieve 90-82. For next model we tried same architecture,just only one change we made, instead of LSTM we used GRU
There is an advantage of GRU over LSTM like less number of gates, less training time,less number of parameters and sometimes GRU performed the same as LSTM.

Layer (type)	Output Shape	Param #
time_distributed_1 (TimeDist	(None, 18, 120, 120, 18)	504
time_distributed_2 (TimeDist	(None, 18, 120, 120, 18)	72
time_distributed_3 (TimeDist	(None, 18, 60, 60, 18)	0
time_distributed_4 (TimeDist	(None, 18, 60, 60, 32)	5216
time_distributed_5 (TimeDist	(None, 18, 60, 60, 32)	128
time_distributed_6 (TimeDist	(None, 18, 30, 30, 32)	0
time_distributed_7 (TimeDist	(None, 18, 30, 30, 64)	18496
time_distributed_8 (TimeDist	(None, 18, 30, 30, 64)	256
time_distributed_9 (TimeDist	(None, 18, 15, 15, 64)	0
time_distributed_10 (TimeDis	(None, 18, 15, 15, 128)	73856
time_distributed_11 (TimeDis	(None, 18, 15, 15, 128)	512
time_distributed_12 (TimeDis	(None, 18, 7, 7, 128)	0
time_distributed_13 (TimeDis	(None, 18, 7, 7, 256)	295168
time_distributed_14 (TimeDis	(None, 18, 7, 7, 256)	1024
time_distributed_15 (TimeDis	(None, 18, 3, 3, 256)	0
time_distributed_16 (TimeDis	(None, 18, 2304)	0
gru_1 (GRU)	(None, 128)	934272
dropout_1 (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 128)	16512
dropout_2 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 5)	645

Total params: 1,346,661
Trainable params: 1,345,665
Non-trainable params: 996

None



Our assumption was wrong val accuracy stuck after 74 and model start overfitting,we just trained for 30 epcoh.

Now we decided to try some Conv3d based architecture

Conv3d

We are able to achieve 90-82 with CNN+LSTM, can we do better than this with help of conv3d.

We decreased epochs to 20,to save time as well computational resource

Model -1 Conv3d

Base model of conv3d have

3 conv3d layer with 32,64,128 cells followed by batch normalization layer. For first layer kernel size is (3,3,3) and for other layers (2,2,2) And Max Pooling layer with pool size (2,2,2)

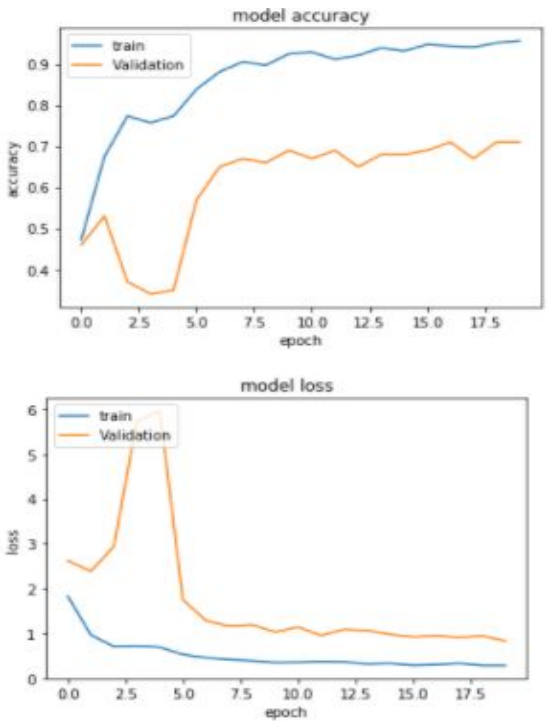
Then flatten layer, dense layer 256 neuron,dropout with .25 and again dense layer 128 and dropout with .25 at last dense layer with 5 cell and softmax activation

We saw significant increase of model parameter our best CNN+LSTM have 1.6 million parameter but for base conv3d model parameter increased to 11 million

Layer (type)	Output Shape	Param #
conv3d_1 (Conv3D)	(None, 16, 118, 118, 32)	2624
max_pooling3d_1 (MaxPooling3	(None, 8, 59, 59, 32)	0
batch_normalization_1 (Batch	(None, 8, 59, 59, 32)	128
conv3d_2 (Conv3D)	(None, 6, 57, 57, 64)	55360
max_pooling3d_2 (MaxPooling3	(None, 3, 28, 28, 64)	0
batch_normalization_2 (Batch	(None, 3, 28, 28, 64)	256
conv3d_3 (Conv3D)	(None, 1, 26, 26, 128)	221312
batch_normalization_3 (Batch	(None, 1, 26, 26, 128)	512
flatten_1 (Flatten)	(None, 86528)	0
dense_1 (Dense)	(None, 128)	11075712
batch_normalization_4 (Batch	(None, 128)	512
dropout_1 (Dropout)	(None, 128)	0
dense_2 (Dense)	(None, 5)	645

Total params: 11,357,061
Trainable params: 11,356,357
Non-trainable params: 704

None



We decided to increase kernel size in other convolution layers to (3,3,3), but we got error.

```
ValueError: Negative dimension size caused by subtracting 2 from 1 for 'max_pooling3d_3/MaxPool3D' (op: 'MaxPool3D') with input
shapes: [?,1,26,26,128].
```

We increased kernel size only for one layer with 64 cells and were able to solve this issue.

2nd Conv3d

Change in kernel size

Parameter decreased by half

We didn't find any good advancement in val accuracy.

For Next conv3d we decided to add one more layer and added argument padding='same' in all conv3d layers so negative dimension size error can be passed.

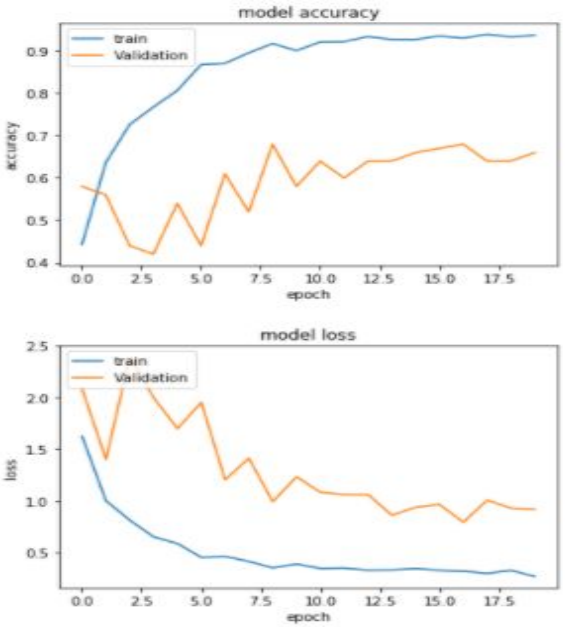
All kernel size is increased to (3,3,3)

For both dense layer we increased neurons to 256.

Layer (type)	Output Shape	Param #
conv3d_1 (Conv3D)	(None, 16, 118, 118, 32)	2624
max_pooling3d_1 (MaxPooling3	(None, 8, 59, 59, 32)	0
batch_normalization_1 (Batch	(None, 8, 59, 59, 32)	128
conv3d_2 (Conv3D)	(None, 6, 57, 57, 64)	55360
max_pooling3d_2 (MaxPooling3	(None, 3, 28, 28, 64)	0
batch_normalization_2 (Batch	(None, 3, 28, 28, 64)	256
conv3d_3 (Conv3D)	(None, 2, 27, 27, 128)	65664
max_pooling3d_3 (MaxPooling3	(None, 1, 13, 13, 128)	0
batch_normalization_3 (Batch	(None, 1, 13, 13, 128)	512
flatten_1 (Flatten)	(None, 21632)	0
dense_1 (Dense)	(None, 256)	5538048
batch_normalization_4 (Batch	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 128)	32896
batch_normalization_5 (Batch	(None, 128)	512
dropout_2 (Dropout)	(None, 128)	0
dense_3 (Dense)	(None, 5)	645

Total params: 5,697,669
Trainable params: 5,696,453
Non-trainable params: 1,216

None

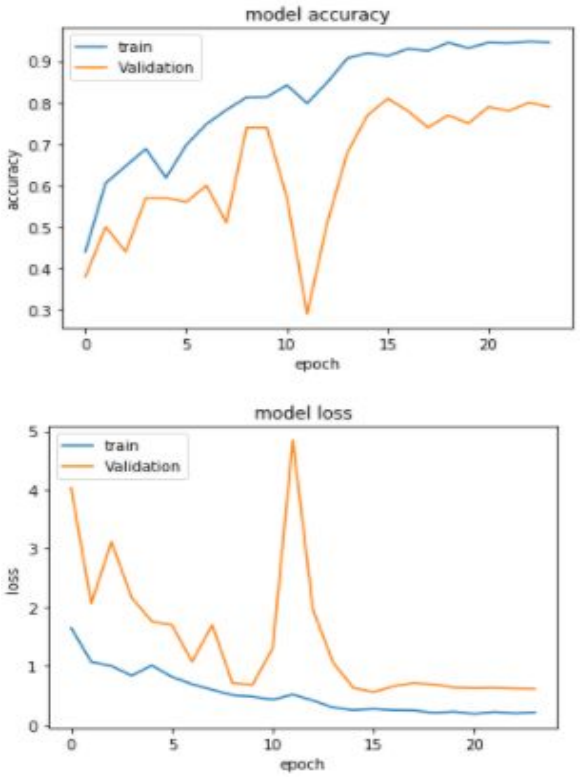


Final Conv3d
We changed image size to 160x160,number of frames to 20 and added one more conv3d layer 16 neuron at starting
Epoch number is increased to 25

Layer (type)	Output Shape	Param #
conv3d_1 (Conv3D)	(None, 20, 160, 160, 16)	1312
activation_1 (Activation)	(None, 20, 160, 160, 16)	0
batch_normalization_1 (Batch	(None, 20, 160, 160, 16)	64
max_pooling3d_1 (MaxPooling3	(None, 10, 80, 80, 16)	0
conv3d_2 (Conv3D)	(None, 10, 80, 80, 32)	13856
activation_2 (Activation)	(None, 10, 80, 80, 32)	0
batch_normalization_2 (Batch	(None, 10, 80, 80, 32)	128
max_pooling3d_2 (MaxPooling3	(None, 5, 40, 40, 32)	0
conv3d_3 (Conv3D)	(None, 5, 40, 40, 64)	55360
activation_3 (Activation)	(None, 5, 40, 40, 64)	0
batch_normalization_3 (Batch	(None, 5, 40, 40, 64)	256
max_pooling3d_3 (MaxPooling3	(None, 2, 20, 20, 64)	0
conv3d_4 (Conv3D)	(None, 2, 20, 20, 128)	221312
activation_4 (Activation)	(None, 2, 20, 20, 128)	0
batch_normalization_4 (Batch	(None, 2, 20, 20, 128)	512
max_pooling3d_4 (MaxPooling3	(None, 1, 10, 10, 128)	0
flatten_1 (Flatten)	(None, 12800)	0
dense_1 (Dense)	(None, 256)	3277056
batch_normalization_5 (Batch	(None, 256)	1024
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 256)	65792
batch_normalization_6 (Batch	(None, 256)	1024
dropout_2 (Dropout)	(None, 256)	0
dense_3 (Dense)	(None, 5)	1285

Total params: 3,638,981
Trainable params: 3,637,477
Non-trainable params: 1,504

None



We got 81 in val and 89 train. So change we made last time helped us.

For next set we try transfer learning and GRU,LSTM

Transfer Learning with LSTM

We already have a state of art model which performed very well in the imagenet dataset.
We can use them as well for feature extraction
Let's see some state of art model

VGG19

VGG19 has 143,667,240 and .h5 file sizes vary near 550 mb. Which is huge and as we are targeting smart tv for deployment of this model. Due to size and number of parameter it will require high RAM to do the task. So we rejectected VGG19

ResNet50

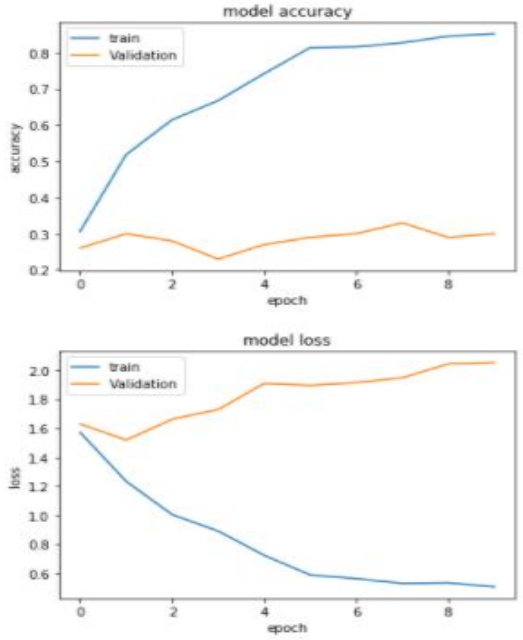
ResNet50 is one of the best state of art model out there. It has 25,636,712 parameter,again it is large number for parameter.

MobileNetV2

MobileNetV2 is good for low ram device like mobile. It has around 3,538,984

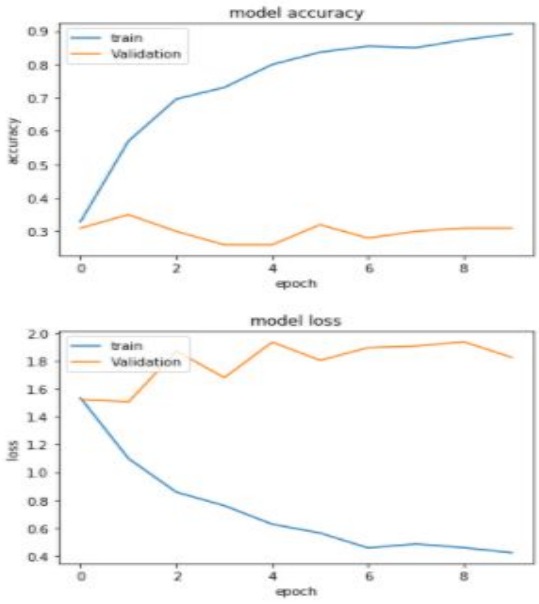
We chose mobile net v2 for transfer learning
We tried both LSTM and GRU with MobileNetV2

LSTM

_____ Layer (type) Output Shape Param # ========== time_distributed_1 (TimeDist (None, 18, 3, 3, 1280) 2257984 _____ time_distributed_2 (TimeDist (None, 18, 3, 3, 1280) 5120 _____ time_distributed_3 (TimeDist (None, 18, 1, 1, 1280) 0 _____ time_distributed_4 (TimeDist (None, 18, 1280) 0 _____ lstm_1 (LSTM) (None, 128) 721408 _____ dropout_1 (Dropout) (None, 128) 0 _____ dense_1 (Dense) (None, 128) 16512 _____ dropout_2 (Dropout) (None, 128) 0 _____ dense_2 (Dense) (None, 5) 645 ========== Total params: 3,001,669 Trainable params: 741,125 Non-trainable params: 2,260,544 _____ None	

We got even worst performance then our base CNN+LSTM and conv3d model model

GRU

_____ Layer (type) Output Shape Param # ========== time_distributed_1 (TimeDist (None, 18, 3, 3, 1280) 2257984 _____ time_distributed_2 (TimeDist (None, 18, 3, 3, 1280) 5120 _____ time_distributed_3 (TimeDist (None, 18, 1, 1, 1280) 0 _____ time_distributed_4 (TimeDist (None, 18, 1280) 0 _____ gru_1 (GRU) (None, 128) 541056 _____ dropout_1 (Dropout) (None, 128) 0 _____ dense_1 (Dense) (None, 128) 16512 _____ dropout_2 (Dropout) (None, 128) 0 _____ dense_2 (Dense) (None, 5) 645 ========== Total params: 2,821,317 Trainable params: 560,773 Non-trainable params: 2,260,544 _____ None	

Again very worst performance
Next, What we can do is to train mobilenet with LSTM or GRU.But due to computational restraint we didn't.

Final Model and reason selection
For the final model we have chosen 5th CNN+LSTM with 82 val accuracy. It has less number of parameters as compared to conv3d model and it will be a good smart tv for task..

Done By-
Arvind Kumar Patel
Subhasis Pattanayak