Observations:

Experiment Number	Model	Result	Decision + Explanation
1	Conv3D	GPU exhausted error	Explanation : batch size was set to 16 and 32. To resolve this error we changed the batch size to 8 Decision : Change batch size to 8
2	Conv3D	Train_acc = 0.92 Val_acc = 0.84	Explanation : Model with 3 conv3d+Maxpool layers and 2 dense layers. We included dropout to each layer in order to overrule chances of overfitting. Looks like model is slightly overfitting, but performance looks good Decision : Since we trained for only 10 epochs, we decide to try for larger epoch size to see if model stabilizes
3	Conv3D	Train_acc: 0.94 Val_acc = 0.90	Explanation : Increasing epoch size to 20 has shown great results and the model looks more stable, but since we noticed slight overfitting in the previous model, we suspect that could still be at play here. Decision : Increase dropout values to resolve any overfitting
4	Conv3D	Train_acc : 0.75 Val_acc : 0.78	Explanation : By increasing the dropout and epoch size to 25 we can see that model accuracy has reduced but so has the difference between the train and val accuracies, this could be the solution to overfitting. We should try to train for greater epoch size to see if model improves Decision : Since learning is very slow we choose to increase epoch size to 40 to see if model improves

5	**Conv3D**	**Train_acc : 0.94** **Val_acc : 0.84**	**Explanation : We see that the training and val accuracy have improved but are similar to the model 2 performance. This could be the best performance possible with this model.** **Decision : The metrics of model 2 and 5 can be frozen as best metrics and we can now try a new architecture to see if we get better results**
6	**Conv2D+GRU**	**Train_acc : 0.98** **Val_acc : 0.97**	**Explanation : In this model we choose to take mobilenet as the conv2d network with pretrained weights from imagenet dataset. Since it would be very difficult to re-train a model to match the pretrained model metrics. Then we construct a gru network which takes the vector output from mobilenet. We add a 64 layer GRU cell in this network with dropouts. The final layers of the network are dense layers which will pass into a softmax to give the predicted class. We get great results for 10 epochs.** **Decision : Build Conv2D+GRU network, reduce epoch size and dropout values to see if performance is similar**
7	**Conv2D+GRU**	**Train_acc : 0.94** **Val_acc : 0.88**	**Explanation : Even after reducing epoch size and dropout values we can see that the performance is very good. These dropout values are promising and might perform better with greater epoch size** **Decision : Dropout values used look promising. We will try with larger epoch size to**

			see if model performance can be maximized
8	**Conv2D+GRU**	**Train_acc : 0.99 Val_acc : 0.97**	**<u>Explanation</u> : After increasing epoch size we can see that the model stabilizes and is performing very well. Performance of model 6 was also very good, so we could try running for same epoch size and increase dropout size <u>Decision</u> : Performance is very good, but we will also run an experiment by increasing dropout value to see how behaviour changes**
9	**Conv2D+GRU**	**Train_acc : 0.98 Val_acc : 0.96**	**<u>Explanation</u> : By increasing dropout values we see slightly lower performance than model 8. Even though there is lesser gap between training and val accuracy, the performance of model 8 is better <u>Decision</u> : Increasing dropout values have degraded the performance a little, model 8 should be considered the best**
Final Model	**Conv2D+GRU**	**Train_acc : 0.99 Val_acc : 0.97**	**<u>Explanation</u> : This model gives the best performance out of all models and also does not show any strong signs of overfitting. This architecture is clearly better performing than the Conv3d architecture <u>Decision</u> : Given the best metrics out of all expts, this model can be considered as best model**