The following table shows the models tried with their corresponding input parameters, output and also the decision of changes for the next model. Overall we tried with Conv3D, time distributed Conv2D + LSTM, time distributed Conv2D + GRU, pre-trained Conv2D with ResNet + LSTM as different categories. In each of these categories various architectures were tried to increase accuracy, decrease over-fitting and manage memory. Please note that various other combinations were tried with simple changes like drop out values, batch sizes, sample frames and kernel sizes. The significant of these are listed in the following table.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Exp# | Model | Input | Brief on architecture | Result | Decision + Explanation |
| 1 | Conv3D | Epochs: - 20  Batch size: -64  Image size: -160x160  Sample frames=30  Optimizer: - Adam | 4 conv layers with 8, 16, 32, 64 cells with kernel size 3, batch normalization and 2 dense layers | ResourceExhaustedError- OOM when allocating tensor with shape [64,16,30,160,160] | Reduce the size of image and no. of frames to reduce the input |
| 2 | Conv3D | Epochs: - 15  Batch size: -64  Image size: -100x100  Sample frames=20  Optimizer: - Adam | 4 conv layers with 8, 16, 32, 64 cells with kernel size 3, batch normalization and 2 dense layers | Train Acc:- 67%  Val\_acc:- 61% | As the accuracy is not satisfactory, decided to:   * Increase img size to 120x120 * Change the architecture * Increase the epochs |
| 3 | Conv3D | Epochs: - 20  Batch size: -64  Image size: -120x120  Sample frames=20  Optimizer: - Adam | 4 conv layers with 16, 32, 64, 128 cells with kernel size 3, batch normalization and 2 dense layers of 64 and drop out of 0.25 after each dense layer | Train Acc:- 98%  Val\_acc:- 79% | The model is good but over-fitting.  Next approach is to try with augmentation to counter over-fitting. |
| 4 | Conv3D | Epochs: - 20  Batch size: -64  Image size: -120x120  Sample frames=20  Optimizer: - Adam  With data augmentation | Same as above | Dst tensor is not initialized | This error message is generated when there is not enough memory to handle the batch size. So, let us reduce the batch size and try again |
| 5 | Conv3D | Epochs: - 20  Batch size: -20  Image size: -120x120  Sample frames=20  Optimizer: - Adam  With data augmentation | Same as above | Train Acc:- 83%  Val\_acc:- 71% | With Data Augmentation (Zoom\_range), there is a drop in validation accuracy. So, we have decided not to perform data augmentation. Next approach is to try reducing the filter size from (3,3,3) to (2,2,2) and see if there is an improvement in val acc |
| 6 | Conv3D | Epochs: - 20  Batch size: -64  Image size: -120x120  Sample frames=20  Optimizer: - Adam | 4 conv layers with 16, 32, 64, 128 cells with **kernel size 2,** batch normalization and 2 dense layers of 64 and drop out of 0.25 after each dense layer | Train Acc:- 88%  Val\_acc:- 76% | This is a good model. Trying to increase the val accuracy to above 80.  Changes made to this:   * Tried cropping the image and resizing and this reduced the accuracy. Hence **decided to not crop images** and only resize * Tried changing the normalization technique from ‘/255’ to max/min and percentile methods. Both of them did not result in any improvement. **So, going with ‘/255’ for normalization.**   Next approach is to try by modifying the architecture **by increasing layers between each maxpooling.** |
| 7 | Conv3D | Epochs: - 20  Batch size: -20  Image size: -120x120  Sample frames=20  Optimizer: - Adam | 8 conv layers with 2 each of 16, 32, 64, 128 cells with kernel size 2, batch normalization and 2 dense layers of 64 and drop out of 0.25 after each dense layer | epoch 9- 85, 79  epoch 10- 90,82  epoch 11- 89,81 --> This is considerable  epoch 12- 92,81 | This is a good enough model. **But trying by changing the image index to drop 10 frames in between (like 1 or 2 after every 3 frames) to select 20 frames distributed across 0 to 30.** |
| 8 | Conv3D | Epochs: - 20  Batch size: -20  Image size: -120x120  Sample frames=20  Optimizer: - Adam | 8 conv layers with 2 each of 16, 32, 64, 128 cells with kernel size 2, batch normalization and 2 dense layers of 64 and drop out of 0.25 after each dense layer | Dst tensor is not initialized | This error message is generated when there is not enough memory to handle the batch size. So, reducing the batch size and trying again |
| 9 | Conv3D | Epochs: - 20  Batch size: -20  Image size: -120x120  Sample frames=20  Optimizer: - Adam | 8 conv layers with 2 each of 16, 32, 64, 128 cells with kernel size 2, batch normalization and 2 dense layers of 64 and drop out of 0.25 after each dense layer | epoch 17- 96, 83  epoch 18- 98,85  epoch 19- 97,85 --> This is the best  epoch 20- 99,84 | This has so far resulted in the **best model.**  Next approach is to try with different optimizers. |
| 10 | Conv3D | Epochs: - 20  Batch size: -20  Image size: -120x120  Sample frames=20  Optimizer 1: adadelta  Optimizer 2: sgd | Same as previous | No improvement | So staying with Adam optimizer.  Parking the Conv3D for now and trying with CNN+RNN approach. |
| 11 | Time distributed Conv2D + LSTM | Epochs: - 50  Batch size: -64  Image size: -50x50  Sample frames=20  Optimizer: - sgd | 4 layers of time distributed conv2D with 16, 16, 32, 32 cells, kernel size 2, batch normalization + **256 cell LSTM** + 64 cell dense layer with regularization and dropout | Train Acc:- 79.6%  Val\_acc:- 80% | Considerable accuracy.  Next approach is to try adding more layers and see if the accuracy improves. |
| 12 | Time distributed Conv2D + LSTM | Epochs: - 35  Batch size: -64  Image size: -120x120  Sample frames=20  Optimizer: - sgd | Adding more layers to the previous | Train Acc:- 98%  Val\_acc:- 81% | Overfitting.  Tried with increasing dropouts but is resulting in lesser accuracies for both train and val.  Next approach is to try with GRU |
| 13 | Time distributed Conv2D + GRU | Epochs: - 20  Batch size: -64  Image size: -120x120  Sample frames=20  Optimizer: - adam | Same as 11 but with GRU 128 cell | Train Acc:- 96%  Val\_acc:- 73% | Overfitting.  Tried with increasing dropouts but is resulting in lesser accuracies for both train and val.  Next approach is to try with pre-trained conv2D with ResNet50 |
| 14 | Pre-trained ResNet50 + LSTM  (ResNet50 not trainable) | Epochs: - 35  Batch size: -64  Image size: -120x120  Sample frames=20  Optimizer: - adam |  | Train Acc:- 99%  Val\_acc:- 18% | Very poor val accuracy.  Next approach is to make resnet50 weights trainable |
| 15 | Pre-trained ResNet50 + LSTM  (ResNet50 trainable) | Epochs: - 35  Batch size: -64  Image size: -120x120  Sample frames=20  Optimizer: - adam |  | Train Acc:- 94%  Val\_acc:- 75% | Not good enough.  The pre-trained models have higher memory requirement and are not satisfactory enough.  So deciding to go with row 9 i.e., conv3d as final model. |