DEEP LEARNING COURSE PROJECT-GESTURE RECOGNITION

The following write-up demonstrates the finding and intuition behind the decision of the best model to perform gesture recognition.

Generator function: Here, we have assumed that consecutive frames give the same information, we have skipped one frame between two frames and picked up images for each video, i.e., out of 30 images/frames per video we have picked up 15.

We have built two models: One is the standard **CNN+RNN** architecture and the other one is the **3D-Convolutional network or Conv3D**.

Conv3D

Experiment 1: We started with a basic network structure with 5 Conv3D layers with pooling and no dropouts or batch normalization. Our intention was to begin with a large model first to see if we get good accuracy, and then try to reduce model size to achieve same or similar accuracy. The results are summarized in the table below:

Image Size	100 X 100
Batch Size	30
Number of Epochs	150
Total number of parameters	4,043,085
Total number of Trainable parameters	4,043,085
Training Accuracy	0.96
Validation Accuracy	0.50

Experiment 2: From the training and validation accuracies obtained in Experiment 1, we observed that the model seems to overfit. So, we added Batch Normalization between convolution layers, and Dropout layer after pooling. This led to an increase in validation accuracy. The results are as summarized in the table below:

Image Size	100 X 100
Batch Size	30
Number of Epochs	150
Total number of parameters	4,043,597
Total number of Trainable parameters	4,043,341
Training Accuracy	0.95
Validation Accuracy	0.70

Experiment 3: After using some regularization techniques performance of model increased significantly compared to previous experiment but the model is still not good enough, hence we increased the layers in the model to see if we get better results. We also changed the pooling size in the middle convolution layer to pass more information to the next convolution layers. The results are as summarized in the table below:

Image Size	100 X 100
Batch Size	30
Number of Epochs	150
Total number of parameters	6,527,237
Total number of Trainable parameters	6,526,933
Training Accuracy	0.94
Validation Accuracy	0.86

Experiment 4: After increasing the number of layers in the model, the validation accuracy shoots up and we get a good accuracy for validation set, but still there is a significant amount of gap between the training and validation accuracy. With the idea that giving more information to the existing model might improve accuracy of the model, we increased the image size from 100x100 to 120x120. The results are as summarized in the table below:

Image Size	120x120
Batch Size	30
Number of Epochs	150
Total number of parameters	11,402,117
Total number of Trainable parameters	11,401,557
Training Accuracy	0.9743
Validation Accuracy	0.97

Experiment 5: In Experiment no 3, we changed the pooling size to pass more information to next layer, which increased size of the model significantly. Now we keep all the parameters unchanged and revert the pooling size to reduce the size of model and see if we get similar accuracy. The results are as summarized in the table below:

Image Size	120x120
Batch Size	30
Number of Epochs	150
Total number of parameters	6,527,237
Total number of Trainable parameters	6,526,933
Training Accuracy	0.959
Validation Accuracy	0.966

Experiment 6: We got good results after experiment 5. Let us further reduce the model size by removing the last convolution layer with 128 parameters. The results are as summarized in the table below:

Image Size	120x120
Batch Size	30
Number of Epochs	150
Total number of parameters	3,314,437
Total number of Trainable parameters	3,314,133
Training Accuracy	0.9841
Validation Accuracy	0.95

Experiment 7: As observed in the previous experiment, the model size reduced significantly and we got pretty good results. Let's reduce the model size further by reducing the number of nodes in the Dense layer to see if get good results. The results are as summarized in the table below:

Image Size	120x120
Batch Size	30
Number of Epochs	150
Total number of parameters	1,708,037
Total number of Trainable parameters	1,707,733
Training Accuracy	0.9725
Validation Accuracy	0.9833

We tried reducing the size further, but the performance of the model deteriorated. Since this model gave us the best results, we decided to go with the model at the end of Experiment 7.

CNN+RNN

Experiment 1: By leveraging the experience we got from the 3D convolutional model, we started by reducing the number of nodes in convolution layer since we will be adding a GRU layer which will further increase the size of model. The results are as summarized in the table below:

Image Size	120x120
Batch Size	30
Number of Epochs	150
Total number of parameters	3,394,589
Total number of Trainable parameters	3,394,333
Training Accuracy	0.97
Validation Accuracy	0.95

Experiment 2: We achieved a good accuracy value on both the training and validation set. We tried increasing the number of nodes and adding kernel regularizer in the GRU layer to see if we can get better results in terms of accuracy. The results are as summarized in the table below:

Image Size	120x120
Batch Size	30
Number of Epochs	150
Total number of parameters	3,428,237
Total number of Trainable parameters	3,427,885
Training Accuracy	0.9609
Validation Accuracy	0.975

Experiment 3: We observed that the performance did not improve significantly, thus we decided to revert to the previous structure. We already have the GRU layer which also has many nodes thus increasing the size of model. Hence, we reduce the number of nodes in the Dense layer. The results are as summarized in the table below:

Image Size	120x120
Batch Size	30
Number of Epochs	150
Total number of parameters	1,739,677
Total number of Trainable parameters	1,739,421
Training Accuracy	0.9609
Validation Accuracy	0.9750

Thus, from the results obtained above we conclude the 3D Convolutional model (obtained after Experiment no 7) to be the best and final model, as it gives the highest accuracy score and has fewer number of model parameters.