**Gesture Recognition Assignment**

Problem Statement:-

As a data scientist at a home electronics company which manufactures state of the art smart televisions. We want 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.

* 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.

**Understanding Dateset**:-The training data is made up of a few hundred videos that have been classified into one of five categories. Each video (typically 2-3 seconds long) is divided into 30 frames (images). These videos were created by various people performing one of the five gestures in front of a webcam, similar to what the smart TV will use. Videos come in two sizes: 360x360 and 120x160.

**Data Pre-Processing**:-To pre-process the data, a function called Generator was created. We will preprocess the images in the generator because we have images in two dimensions, as well as create a batch of video frames. To extract the best features and reduce image size, images were pre-processed with normalisation, standard scaling, and cropping. To identify the correct gesture, a set of images is fed into various CONO3D and CNN-RNN models.

**Model Development**: We worked on multiple models with varying batch sizes, epochs, and hyper parameters, as detailed in the table below. A convolutional 3D model with four convolutional layers outperformed the others in terms of validation accuracy. After each convolutional layer, batch-normalization and max-pooling layers were added to reduce overfitting; a dropout of.25 was added after the third convolutional layer to reduce overfitting. Finally, the output from these layers was flattened,

dropped out, and fed into the five-output soft-max layer.

Observations:-

|  |  |  |  |
| --- | --- | --- | --- |
| Experiment Number | Model | Result | Decision + Explanation |
| 1 | Conv3D | Epochs: 20  Train accuracy: 0.21  Validation accuracy: 0.42 | Since it is obvious that the model cannot learn, extra layers should be added. |
| 2 | Conv3D | Epochs: 20  Train accuracy: 0.60  Validation accuracy: 0.76 | The number of parameters has expanded dramatically even though the model's accuracy has improved relative to the previous model. lowering the number of the parameters. |
| 3 | Conv3D | Epochs: 20  Train accuracy: 0.96  Validation accuracy: 0.82 | The model appears to be over-fitting, further lowering the number of parameters. |
| 4 | Conv3D | Epochs: 20  Train accuracy: 0.18  Validation accuracy: 0.56 | The model is under-fitting as a result of reducing the number of parameters, increasing the number of epochs from 20 to 40, and deleting the Dropout layers from the model. |
| 5 | Conv3D | Epochs: 40  Train accuracy: 1  Validation accuracy: 0.93 | Reducing the amount of parameters and reintroducing dropout layers show that the model is over-fitting. |
| 6 | Conv3D | Epochs: 40  Train accuracy: 0.98  Validation accuracy: 1 | Clearly, the model is under-fitting. Increasing the amount of Dropout layers and using GlobalAveragePooling3D in place of Flatten layer |
| 7 | Conv3D | Epochs: 40  Train accuracy: 0.98  Validation accuracy: 0.92 | Despite using an alternative architecture, the model still appears to be over-fitting. |
| 8 | TimeDistributed Conv2D + LSTM | Epochs: 20  Train accuracy: 0.52  Validation accuracy: 0.68 | The low precision is probably caused by the few epochs, however due to time constraints, switching to a different architecture, raising the number of epochs from 20 to 40, was necessary. |
| 9 | TimeDistributed Conv2D + GRU | Epochs: 40  Train accuracy: 0.95  Validation accuracy: 0.83 | The model performs well, however the validation accuracy differs significantly from the train accuracy, necessitating the addition of some Dropout layers. |
| 10 | TimeDistributed Conv2D + GRU | Epochs: 40  Train accuracy: 0.85  Validation accuracy: 0.59 | Dropout layers were added, and this had a bad effect on the model. changing the model architecture and raising the epoch count to 50 |
| 11 | TimeDistributed Conv2D | Epochs: 50  Train accuracy: 0.99  Validation accuracy: 0.92 | Dropout layers are used to deal with the model's apparent over-fitting since it |
| 12 | TimeDistributed Conv2D | Epochs: 50  Train accuracy: 0.96  Validation accuracy: 0.91 | The model's accuracy ratings are excellent. deciding to use model 12 as the ultimate model |

Conclusion:-

Our final model was Model 12 (TimeDistributed Conv2D), which met our objectives with training accuracy of 0.96 and validation accuracy of 0.91.