For our base Models, we started with the suggested architectures of Conv3D + MaxPooling3D and Conv2D+RNN which also included a GRU layer.

Before Model building, we modified the given generator skeleton code to work with our dataset and pick up residual data to complete one pass on the data. This generator also included the preprocessing the image data such as resizing and normalizing pixel values on each channel.

For the Convolutional 3D Base Model (Model 1), we chose the base as 8-16-32-64 layers with 256-128 dense and final dense layer of 5 neurons with SoftMax activation. We chose the ADAptive Moment Estimation (ADAM) as our optimizer which is modification to SGD and works similarly.

For our other Convolutional 2D Base Model (Model 2), we chose a similar layer architecture but used TimeDistributed and included a GRU before final Dense layer.

We applied BatchNormalization and Dropout layers in our base models as well to provide some regularization.

The Base Models yielded moderate results with the indication of overfitting and with that note we started out the experiments.

In Experiment 1, we tried playing around with input constants like Batch Size, Image Size, Epochs and Window Size. The Models used were similar to Base Models only difference being the change in mentioned constants. This experiment didn’t yield good results as overfitting was still present in both the Models (Model 3 and Model 4).

In Experiment 2, the aim was to reduce the overfitting so we tried reducing Model complexity in both the Base Models. Base Models were modified in such a way that both the Models had around half the number of parameters than their Base Model counterparts.

This approach yielded good results with the Conv3D model (Model 5) as its performance on validation dataset increased significantly and the accuracies in train and validation sets were comparable. The Conv2D model (Model 6) was still overfitting the data as the train accuracy reached 100% whereas validation accuracy remained at 70%.

Finally in Experiment 3 we tried to improve the performance of Conv2D Model by employing Transfer Learning (Model 7). We used a MobileNet + GRU architecture for this experiment. MobileNet was chosen as it is quite lightweight when compared to other pre trained CNNs like AlexNet, GoogleNet, VGC16 and the overall aim was a less complex model. However, this model ended up with highest number of parameters but the performance was improved significantly with respect to other Conv2D Models we created.

In conclusion, we had two best performing models, the first was Conv3D + MaxPooling3D with reduced complexity which we created in Experiment 2 and the Transfer Learning MobileNet + GRU. Both these Models had a similar performance but the former had a significantly lower number of parameters than the latter so we chose Model 5 as our final Model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment Number** | **Model Number** | **Model Architecture** | **Result** | **Decision + Explanation** |
| **0 (Base)** | **1** | **Conv3D + MaxPooling3D** | **Accuracy on Train: 96%**  **Accuracy on Validation: 87%** | **Base Model. Difference in accuracies of train and validation. Presence of Slight Overfitting** |
| **0 (Base)** | **2** | **Conv2D + RNN + GRU** | **Accuracy on Train dataset: 100%**  **Accuracy on Validation dataset: 70%** | **Base Model. Train accuracy reaches 100%, significant difference in train and validation accuracies. Overfitting** |
| **1** | **3** | **Conv3D + MaxPooling3D** | **Accuracy on Train dataset: 70%**  **Accuracy on Validation dataset: 32%** | **Tried changing Batch Size, Window Size, Image Size, Epochs, Total number of Frames. Presence of clear overfitting with poor performance on validation set.** |
| **1** | **4** | **Conv2D + RNN + GRU** | **Accuracy on Train dataset: ~95%**  **Accuracy on Validation dataset: 56%** | **Tried changing Batch Size, Window Size, Image Size, Epochs, Total number of Frames. Results similar to Model Number 3** |
| **2** | **5** | **Conv3D + MaxPooling3D** | **Accuracy on Train dataset: ~97%**  **Accuracy on Validation dataset: 96%** | **Reduced complexity on Model 1. Halved the number of parameters from Model 1. Gives good performance of 95+% accuracies in train and validation sets.** |
| **2** | **6** | **Conv2D + RNN + GRU** | **Accuracy on Train dataset: 100%**  **Accuracy on Validation: 70%** | **Even after reducing complexity of Model 2, this architecture still tends to overfit train data. We will try to modify this architecture by applying Transfer Learning.** |
| **3** | **7** | **Conv2D (MobileNet) + GRU** | **Accuracy on Train dataset: 100%**  **Accuracy on Validation dataset: 95%** | **Applied Transfer Learning here. Model still slightly overfitting but performance improved on validation set, when compared to other GRU Models.** |