**GESTURE RECOGNITION PROJECT**

General experiments and observations :

* For selection of batch size, we started with batch size of 50, however, with this the Kernel dies and we cannot run the model
* Reducing to a batch size of 40 was tried however, with this every epoch is very slow further reducing it to 30 gives an optimal batch size
* For all model we first tried with a subset to check how the model performs and if it allows to overfit on the training data
* With SGD optimizer the training loss was constant and not reducing over Epochs
* The images of size 160x120 (width x height) were initially cropped in the generator function before resizing as image[:120, 20:140]. That is, 20 from each side. However, it was giving poor training results as compared to resizing without cropping. This could probably be because of loss of information on either side when the images are cropped. The gestures being cut towards the ends. Thus, we did not crop the images.
* Batch normalization was introduced in the model to increase the training speed
* Dropouts have been introduced in the models in trying to tackle overfitting
* Maxpooling layers introduced to pool the data and reduce the trainable parameters entering into the subsequent layers

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| Initial Experiments in trying to get a model | CNN + GRU(RNN) | 1. Kernel dies 2. Epochs are slow 3. 6 hidden layers without pooling 4. Model starts to train | 1. Batch size 50 2. Batch size 40 3. Too many trainable parameters in the network and thus training not possible 4. Reducing layers and introducing pooling layers and batch normalization |
| **Model 1 :**  Batch size: 30  Epochs : 25  Image size: 100 \* 100  Frames per video : 15  Optimiser : Adam  Normalization : minmax | CNN + GRU(RNN) | Accuracy : 70%  Check for a better model | This base CNN+RNN model was first tried on a small subset of the data to check if it trains well. Following are the observations:   * It was observed that with SGD optimizer and a learning rate of 0.001 there was no reduction in the training loss across Epochs. * The same when switched to Adam optimizer with same learning rate shows reduction in training loss across Epochs. * Also, if we crop the 160x120 image to 120x120 and then resize we see a reduction in model performance as compared to when it is only resized * Training loss: 0.0462 and val loss : 1.20 * There is lot of difference between training and validation accuracy indicating that the model is overfitting. * Hence, we experiment with other parameters in the generator function * However, model overfits |
| **Model 2 :**  Batch size: 30  Epochs : 25  Image size: 100 \* 100  Frames per video : 15  Optimiser : Adam  Normalization : percentile | CNN + GRU(RNN) with change in normalization technique | Accuracy : 61%  Check for a better model | Base model run with percentile normalization instead of minmax. Following are the observations:   * The training time remains similar * The train and validation accuracy are further apart now for the same number of epochs thus the model with percentile normalization overfitting * However, model overfits |
| **Model 3 :**  Batch size: 30  Epochs : 35  Image size: 100 \* 100  Frames per video : 15  Optimiser : Adadelta  Normalization : minmax | CNN + GRU(RNN) with optimizer change and normalization technique back to minmax | Accuracy: 67%  Check for a better model | Base model run again with minmax normalization as this gave better results and now varying the optimizer   * In earlier models we see that the validation loss stops improving over epochs thus introduce the earlystopping callback to stop the model when validation loss stops improving for 10 epochs * With change in optimizer we see the training loss converging faster and improving training accuracy * Thus, we go ahead with minmax normalization and adadelta optimizer * However, model overfits |
| **Model 4 :**  Batch size: 30  Epochs : 35  Image size: 120 \* 120  Frames per video : 30  Optimiser : Adadelta  Normalization : minmax | CNN + GRU(RNN) with change in image size and frames used for training. Keeping other parameters same as in above model | Accuracy : 72%  Check for a better model | Since in above models we still see overfitting we increase the input frames and check the model performance. Following are the observations:   * We try training by increasing the image size and frames per video * The number of trainable samples increases on increasing the number of frames per video and image size due to which the time taken to run each Epoch also increases * However, this increase shows a positive effect on the accuracy and hence we continue with this image size and frames per video * However, model still overfits |
| **Model 5:**  VGG16  Batch size: 30  Epochs : 35  Image size: 120 \* 120  Frames per video : 30  Optimiser : Adadelta  Normalization : minmax | Transfer Learning (VGG16) + GRU(RNN) | Accuracy : 74%  Decent model, Check for a better one | We use the VGG16 model and GRU with image size, normalization, optimizer and frames per video as seen best from the above model. Observations :   * We have lot of parameters with inclusion of VGG16 model however weights of most of these are trained already. * We thus get advantage of the pretrained parameters plus a few additional parameters from our GRU and dense layer * It takes time like above model 4 but gives us better accuracy * However, model still overfits |
| **Model 6:**  ResNet50  Batch size: 30  Epochs : 35  Image size: 120 \* 120  Frames per video : 30  Optimiser : Adadelta  Normalization : minmax | Transfer Learning (RESNET50) + GRU(RNN) | Accuracy: Very Low | We use the RESNET50 model and GRU with image size, normalization, optimizer and frames per video as seen best from the above model. Observations:   * Restriction on image size being at least 197x197 for resnet50 model which increases the parameters as seen in models above * We have a lot of parameters however, as compared to VGG16 the model here has lot of trainable parameters as well * The model when trained on a subset does not seem to train well |
| Initial experiments in building a Conv3D | Conv3D | Similar observations as seen while developing CNN+GRU(RNN) model |  |
| **Model 7:**  Batch size: 25  Epochs : 35  Image size: 100 \* 100  Frames per video : 15  Optimiser : Adadelta  Normalization : minmax | Conv3D | Accuracy : 64%  Check for a better model | We use a 3D convolutional network with kernel size (5,5,5)   * This leads to more trainable parameters * We run the model on a subset of dataset and observe not so good accuracy hence we engage in playing with layers and filters to achieve better accuracy with optimal number of parameters |
| **Model 8:**  Batch size: 25  Epochs : 35  Image size: 100 \* 100  Frames per video : 30  Optimiser : Adadelta  Normalization : minmax | Conv3D  With image size and frames per video change | Accuracy : 77% | Explorations for getting the right layers and filters for a good model   * Added layers which increased the training parameters so then tried to optimize it against accuracy and loss values * Introduced dropouts after every pooling layer to fix overfitting issues however this reduced performance and over fitting exist * Place dropouts at select locations to obtain the best accuracy model |
| **Final Model (Model 8)**  Batch size: 25  Epochs : 35  Image size: 100 \* 100  Frames per video : 30  Optimiser : Adadelta  Normalization : minmax | Conv3D | Accuracy : 77% | Number of hidden layers  Pooling layers  Filter sizes  Kernel size  Batch normalization  Dropouts  Learning rates  All the above adjusted for better accuracy |