**Gesture Recognition Analysis**

*By:*

**Deepak Hiremath**

Problem Statement:

Imagine you are working as a data scientist at a home electronics company which manufactures state of the art smart televisions. You want to develop a cool feature in the smart-TV that can recognize five different gestures performed by the user which will help users control the TV without using a remote.

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

* 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

Each video is a sequence of 30 frames (or images).

Model and Explanation:

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment** | **Model** | **Result %** | **Observation + Decision** |
| **Conv3D**  1. | *Initially tried with some simple runs to get the understanding of data and find the epochs, batch size, image size to start with, then we started with below model.*  Conv3D  Epoch=15  Batch\_size=40  optimiser = 'adam'  image: 160\* 160 | Train Accuracy:97.13  Val Accuracy: 20.00 | - Models shows high training accuracy and low validation accuracy  - Model is overfitting.  - So we need to do data augmentation to handle this issue |
| 2. | *We are trying to handle the overfitting with augmentation and more epochs*  *We also increased dropout from 0.25 to 0.5 to see if helps in underfitting*  - Data Augmentation - to address overfitting  - Batch Size 20  - Epoch from 15 to 25  - Dropout from 0.5 to 0.25Conv3D  Epoch=25  Batch\_size=20  Dropout=0.5  Image=160\*160  optimiser = 'adam' | Train Accuracy:85  Val accuracy:64 | - Model is not overfitting and we get a good training and validation accuracy of around 64%.  -Observed a very nice trend , an improvement due to data augmentation.  -More the epochs we got better results |
| 3. | *Next we will try some more methods,*  *reduce the filter size*  *reduce image resolution.*  Epoch=30  Batch size=32  Filter size=2,2,2  Image=120\*120  Dropout=0.25  optimiser = Adam  LR 0.0002 | Train Accuracy:67  Val accuracy:55 | This time we didn’t get good model, we tried with different combination, here we get some extra problem the loss didn’t get reduce constantly. I see that the model didn’t improve compared to previous model.  But it didn’t overfit |
| 4. | *-Adding more layers, trying to see if model accuracy increases if we add more layers.*  CNN layers=8  Epoch=30  Filter size=(3,3,3)  Image=120\*120  Batch size=20  optimiser = Adam  LR 0.0002 | Train Accuracy:74  Val accuracy:73 | In this model after making some changes in image size I was able to manage a model better than previous one.  With more layers we see drastic performance improvement. We get validation accuracy |
| 5. | *-Building a low memory foot print model by reducing the parameters, reducing neurons*  Epoch=25  Filter size=(3,3,3)  Image=120\*120  Batch size=20  optimiser = Adam  LR= 0.0002  Dropout=0.25 | Train Accuracy:74  Val accuracy:71 | This was a very good learning, with less params we are able to achieve a better model, but it didn’t improve even with multiple experiments of filter sizes.  We realized we cannot improve more so we started to use LSTM. |
| RNN  6. | **CNN+LSTM**  **5 CNN 2D layers**  **1 LSTM layer of 128 cells**  num\_epochs=25  batch\_size = 32  Image=120\*120  Batch size=20  optimiser = Adam  LR= 0.0002  Dropout=0.25  Filter=2 | Train Accuracy:92.91  Val Accuracy:76 | * For LSTM-CNN model we observed a best validation accuracy of 76% and training accuracy of around 92%. * As we see overfitting, let us augment the data with a small amount of rotation and with the same model. |
| 7. | *Augment the above RNN CNN model*  neurons=256  num\_epochs=30  batch\_size = 20  Image=160\*160  Batch size=20  optimiser = Adam  LR= 0.0002  Dropout=0.25  Filter=3 | Train Accuracy:79.68  Val Accuracy:78.7 | * Model has given a good accuracy of around 79% for both training and validation. |
| 8. | *Mobilenet + LSTM model*  LSTM cells:128  num\_epochs=20  batch\_size = 5  Image=120\*120  Batch size=20  optimiser = Adam  LR= 0.0002  Dropout=0.25 | Train Accuracy:97  Val Accuracy:78 | * Model training accuracy 97% and validation accuracy 78% * We are not training the mobilenet weights and we see validation accuracy is very poor. * Let's train them as well and observe if there is performance improvement. |
| 9 | *Mobilenet+GRU*  *- Transfer Learning with GRU and training all weights*  *-GRU cells=128*  - Cells: 64  - epochs: 20  - batch size: 5  - image size:120\*120  *-* Dropout=0.25 | Train Accuracy:98.5  Val Accuracy:93 | * This has the best performance * This is our final model |

**Conclusion:**

I tried lot of combinations of model, Finally I got best fit in ModelNet+ GRU Model and this is my final model as well.

1. We started building our model with exercise, experiment on getting the optimal parameters like image resolution, frame size and batch sizes.
2. We have tried different models starting with a base model of CNN only and slowly building up by including RNN layers. We have also included transfer learning to check its performance in gesture recognition.
3. In all models, we checked different parameters and tested their performance on the available data. The CNN 2D models were generally overfitting with high training accuracy of around 90% and low validation accuracy of around 71% while a CNN 3D model gave around 79% with no overfitting.
4. CNNs with RNN proved better than only CNN models. Both CNN 2D and CNN 3D models with RNN showed similar accuracies.
5. Transfer Learning. Lastely We trained the models using transfer learning from MobileNet with GRU cells without and with training of transfer learning weights. The model with trained transfer learning weights provided the best accuracy. We achieved a training accuracy of ***98.5%*** and a validation accuracy of ***93%***

## Learnings:

1. As seen in this study, transfer learning has proved to be better than any other models including complex ones with LSTM and GRU with CNN. Any machine learning model should be quickly deployed. So, using available resources like transfer learning is important
2. In the absence of a suitable transfer learning model, we should explore more options and layers of the different available CNN and RNN models including various activation functions and residual network options.