Hand Gesture Detection Using CNN Based Models

Introduction

In this Project, a model for human Hand Gesture Detection is implemented. The gestures are continuously monitored by the webcam mounted on the TV. The gestures are analysed, classified and accordingly, an action is taken.

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

Business Objective

Develop an AI model that can fit into the WebCam memory and classify the given gesture into one of the 5 categories mentioned above. The training/response time and the size of model are important parameters to be considered alongside the accuracy of the model.

Technical Goal

Using the available training data, develop a light AI model that can give decent accuracy. Keeping in view the requirement that the model should fit into webcam, the number of training parameters should be kept to least possible values.

Understanding Data

The training data consists of a few hundred videos categorised into one of the five classes. Each video (typically 2-3 seconds long) is divided into a sequence of 30 frames (30 RGB images). Various people performing one of the five gestures in front of different webcams have recorded these videos.
Since the webcams used are different, there could be change in the image specifications.
The CSV log files will contain the mapping of the video sequence to the unique label assigned to the gesture. There are 663 training samples and 100 validation samples.

Methodology

The approach is divided into following segments:

- Segment: 1 – Perform EDA on Training Data Video
- Segment: 2 - Define Data Generator and various Basic Models. Choose the Appropriate Model
- Segment: 3 - Apply Pre-processing techniques and Run the selected Model
- Segment: 4 - Tune the Hyper parameters of the model
- Segment: 5 - Run the Final Model and store the H5 file

At each of the segments, model parameters and behaviour has been studied and conclusions made. Please refer the attached Jupyter Notebook for the detailed stepwise approach. This document covers the approach at high level and the conclusions made at each of these segments for arriving at best recommended model.

Processing Segment 1

In this segment, sample-training video has been studied. Various pre-processing techniques are applied. Based on the outcome of analysis, the following observations and conclusions are made.

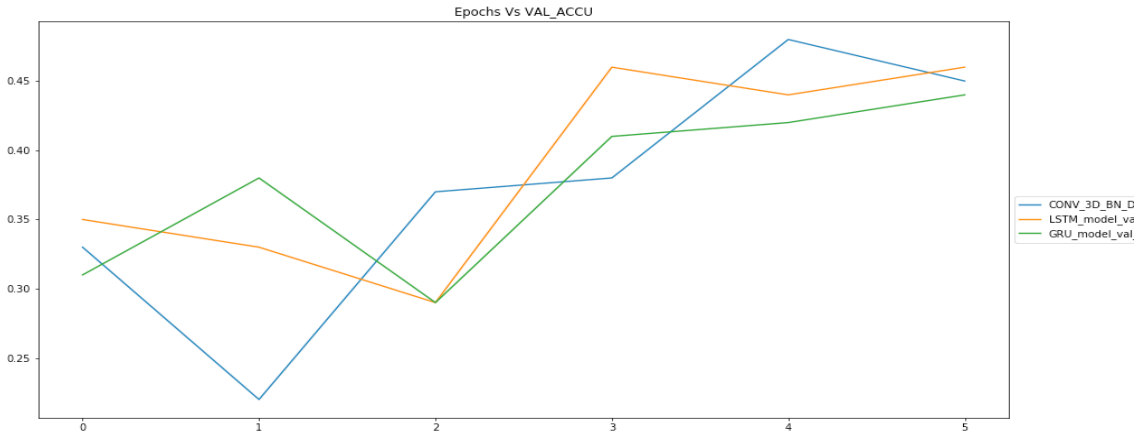
- CV2 MinMax Normalisation will be selected as pre-processing technique. Set the min and max as - 0 and 127.5. This technique will normalise the image will skewness is not introduced.
- Cropping leads to data loss - especially in some images and as such, resize will be done. Resizing maintains the aspect ratio of image
- Algorithmic learning might drop drastically due to edge detection, as the object that will need to be detection various according to the gesture.
- Affine, Perspective and Flip Transformations will aid the learning, as additional data points are augmented during training period.

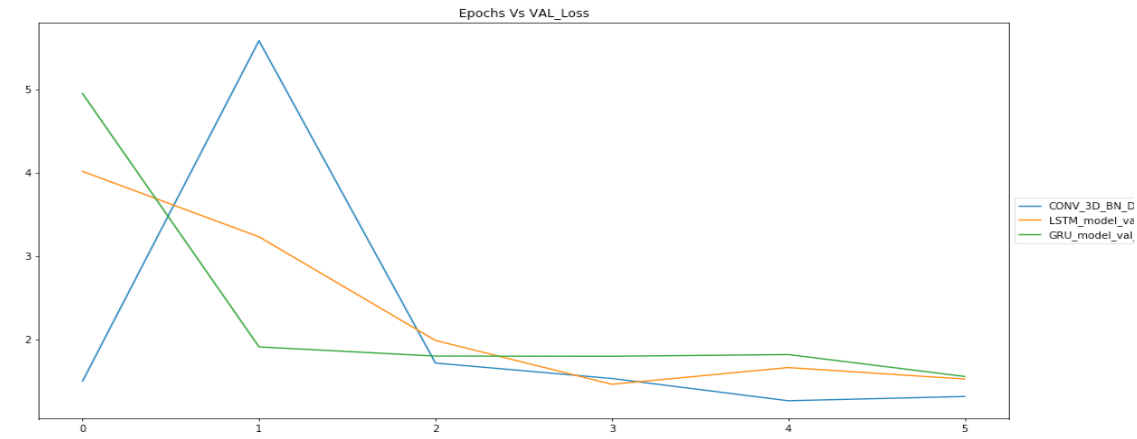
Processing Segment 2 – Basic Model Selection

Three basic models have been setup and the training data was run through the models.

- CONV3D model outperforms the LSTM and GRU model interms of learning.
- CONV3D model has least set of weights and hence better learning can be achieved and is also very economical for training.
- GRU model has higher loss compared to the LSTM. However, over the number of epochs, there is subtle difference between the two.

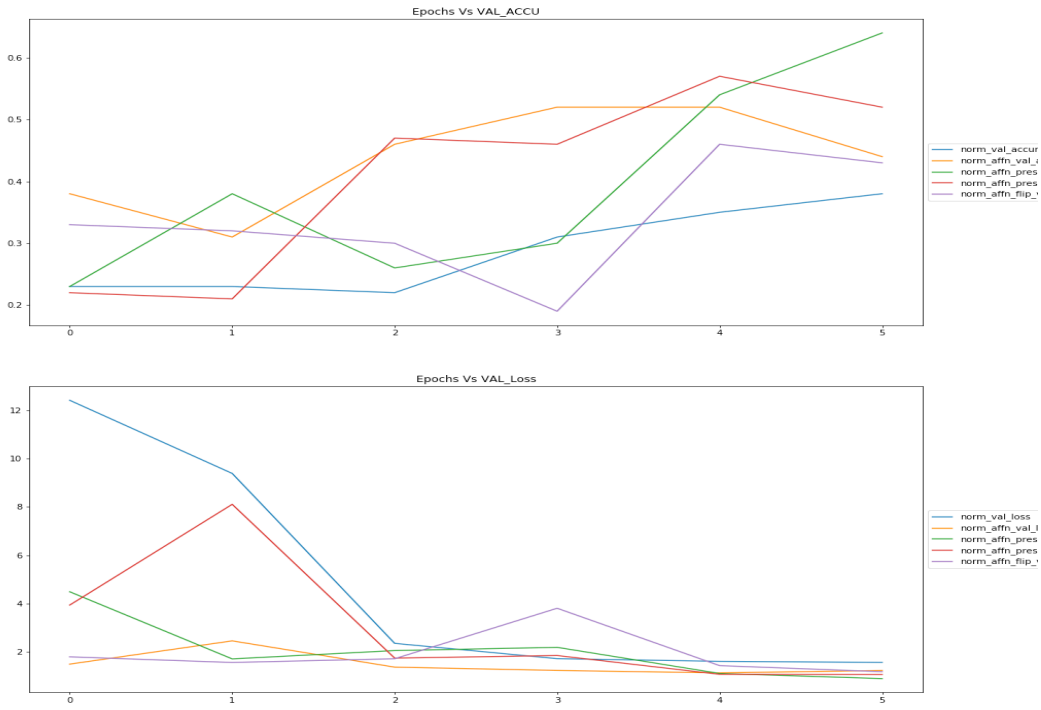
Based on these observations, CONV3D model will be selected for further processing.





Processing Segment 3: Image Pre-processing Techniques

- Various pre-processing techniques are applied based on the Flags passed to the function.
- In case a single channel image processing is requested, the generator function will pick up Red channel
- The key Data Augmentation techniques applied are – Image Resizing, Normalisation, Affine Transformation, Perspective Transformation and Flip Transformation.
- The Validation accuracy and Validation loss are captured from 'fit_generator' function for model valuation and these values are plotted across the Epochs.
- Cropping has been avoided to preserve the data loss.
- From the epochs vs Accuracy curve, it is evident that red line(model with norm_affn_presp_flip) and green line(model with norm_affn_presp) have higher learning rates. The accuracy increased rapidly.
- These data augmentation techniques have helped the learning.
- As per the epoch vs Loss curve, the loss in green line is slightly less than that compared to red line. This could be attributed to over fitting in the red line model. The flip technique applied, might potentially be leading to over fitting over number of epochs.
- We can apply two of the three augmentation techniques and this can help control the over fitting.

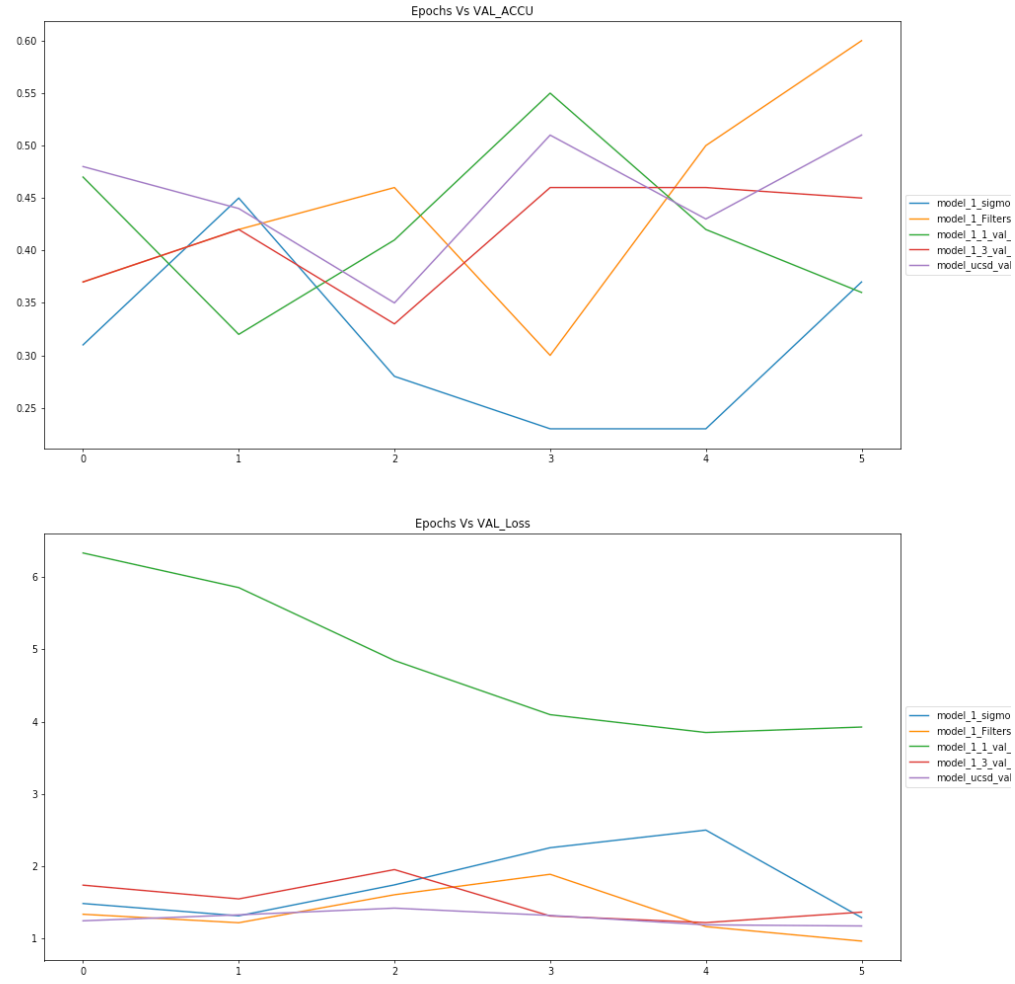


Processing Segment 4: Hyper Parameter Tuning

In this segment, various hyper parameters for the model are tuned. The behaviour of the Network for each of the key hyper parameter is experimented and results logged.

Parameter	Observation
Optimiser Function (SGD vs ADAM)	• From the epochs vs loss curve, it is evident that the variations in the loss is higher in case of model where 'ADAM' optimiser is used. • At the end of Epochs, the magnitude of loss is less in case of model where 'ADAM' optimiser is used. • Thus, we can conclude that model with 'ADAM' optimiser is able to learn faster and better. This Optimiser will be selected for further processing.
Learning Rate - [0.0001,0.001,0.002, 0.003,0.005,0.01,0.1]	• From epochs vs accuracy curve, the top 4 models are those with learning rates - 0.001,0.002,0.003 and 0.1 • From the epochs vs loss curve, the top 4 models are those with learning rates - 0.002,0.001,0.003 and 0.1 • The learning rate 0.1 is too high and often will lead to gradient decent explosion and global minima can never be reached. • The model with 0.002 has a higher loss during epochs 2 and 5. However, loss gradually decreased towards end of epochs. • This characteristic of the model implies that the model has effectively learnt during training. Learning rate - 0.002 is used.
Activation Function – Sigmoid vs Relu	• The model performs better with 'ReLu'activation function .
Number of Filters	• The higher the number of filters the better is the model performance. The model will extract less number of abstract features. However, in certain scenarios, the model might not perform well (For example, in the case of poor image quality or low light, the model with higher kernel size, might not be able to extract the maximum abstract features and therefore, it can render poor results).

Batch Normalization, Drop outs and L2 Regularization	• Adding the dropout at the end of Convolution Layers and Dense layers has reduced the over fitting. • Adding L2 regularization slightly impacted the maximum accuracy
Increase Kernal Sizes	• This could not be experimented as the GPU runs out of memory
BottleNeck Layers	• Addition of bottle neck layer resulted in faster model learning
Co-joint Spatial and Time axis Layers	• The CNN layers have been separated for processing the Space (Image height and width) and the Time axis. This resulted in a stable and fast model. This is implemented based on research paper - https://arxiv.org/abs/1712.04851
Image Channel Selection (R vs RGB)	• The Red channel has been used to test model performance. For single channel, the model learning rate is slower. Therefor RGB is used for final processing, • In order to experiment the Grey Image processing, the data generator function will need to be modified and due to time constraints, this is not implemented in this study.
Number of Images per Video	• The optimal number of images selected is 24. The First 3 and the last 3 images of each of the videos can be discarded. Decent accuracy can be achieved.
CONV3D vs CON2D+GRU	• The number training parameters in the CONV2D+GRU are much higher than CONV3D. This results in a very heavy model and hence, the CONV3D model is recommended. • Also, the problem of over fitting is more predominant in CONV2D+GRU model



Note: Please Refer Jupyter notebook for detailed plots for each of the tuning parameters

Processing Segment: 5 - The Final Model

A CONV3D model with CNN layers processing the space(image height and width) and Time axis co-jointly has been used for classification.

The number of epochs has been increased drastically to generalise the model training. Batch Size of 10 (x4 = 40) has been considered for the batch run. Batch run above this value will lead to system resource errors,

Based on above considerations, the model that has **training accuracy of 0.7376 and validation accuracy of 0.84** will be selected. Please note that Keras call-back Model checkpoint functionality is enabled only at final model execution, to store the best performing H5 file.

NOTE: There is a small drop in the training accuracy compared to Validation accuracy and this is due to the Batch Normalisation and Dropouts added with in model.