Gesture Recognition

# Problem Statement

A home electronics company which manufactures state of the art smart televisions, wants to develop a cool feature for a smart TV which is to recognize 5 different hand gestures which helps users control the TV without remote control.

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

# Details of the dataset

There are 663 videos recorded for training, each video is divided into 30 frames. There are 100 videos of 30 frames each to validate the accuracy of the solution.

# Possible Solutions

Using simple machine learning techniques like logistic regression can only be used for image classification, here we are dealing with large volumes of sequential data so we should be using more robust learning techniques which can deal with large amounts of sequential data, like neural networks.

2D Convolutional neural networks which are typically used for image processing cannot be used with sequence/time (the 3rd dimension), hence we should use a sequence learning Neural network architecture like RNN or use 3D convolution techniques. In this project we will use both.

First let us look at some of the pre-processing techniques that are applicable to this project.

# Pre-processing

* Crop
* Resize
* Generators.

# Solution 1: 3D Convolution

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment Type** | **Details** | **Validation Accuracy** | **Time per epoch** | **Analysis.** |
| 2 Conv layers  Ablation Experiment | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), flatten, dense(256), dense(256), softmax  Activation=relu | NA | NA | The model looks correct. |
| 2 Conv layers  Overfitting Intentionally | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), flatten, dense(256), dense(256), softmax, Activation=relu | NA | NA | Training accuracy was around 96, and there was significant gap between training and validation accuracy. |
| 2 Conv layers  Final Experiment without Batch Normalization and Dropouts | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), flatten, dense(256), dense(256), softmax, Activation=relu  Batch size = 30  Epochs = 20  Callbacks=checkpoint, Reduce LR on Plateau, Early Stopping  Image dimensions = 120,120 | 58% | 25s | The kernel crashed few times due to excess RAM usage while training in google colab – 12.72 GB RAM.  The training accuracy = 95%, overfitting observed. |
| 2 Conv layers  Final Experiment without Batch Normalization and Dropouts | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), flatten, dense(256), dense(256), softmax  , Activation=relu  Batch size = **10**  Epochs = 20  Callbacks=checkpoint, Reduce LR on Plateau, Early Stopping  Image dimensions = 120,120 | 68% | 25s | The kernel did not crash.  The training accuracy = 95%, overfitting treated to some extent but not completely |
| 3 Conv layers  Overfitting Intentionally | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), 128 conv(1,1,1), maxpool(2,2,2) flatten, dense(256), dense(256), softmax | NA | NA | Training accuracy was around 90, and there was significant gap between training and validation accuracy. |
| 3 Conv layers  Final Experiment without Batch Normalization and Dropouts | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), 128 conv(1,1,1), maxpool(2,2,2) flatten, dense(256), dense(256), softmax  Batch size = 10  Epochs = 30  Callbacks=checkpoint, Reduce LR on Plateau, Early Stopping  Image dimensions = 120,120 | 66% | 25s | The kernel did not crash.  The training accuracy = 99%, overfitting treated to some extent but not completely |
| 3 Conv layers  Final Experiment with Batch Normalization | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), dropout(0.25), batchnormalization, 64 conv(3,3,3), maxpooling(2,2,2), ), batchnormalization,128 conv(1,1,1), maxpool(2,2,2) , ),batchnormalization, flatten, dense(256), dense(256), softmax  Batch size = 10  Epochs = 30  Callbacks=checkpoint, Reduce LR on Plateau, Early Stopping  Image dimensions = 120,120 | 60% | 25s | The kernel did not crash.  The training accuracy = 94%, the model did not perform well. |
| 3 Conv layers  Final Experiment with Dropouts only | Architecture: [32 Conv(3,3,3), maxpool(2,2,2), 64 conv(3,3,3), maxpooling(2,2,2), 128 conv(1,1,1), maxpool(2,2,2) flatten, dense(256), dense(256), softmax  Batch size = 30  Epochs = 30  Callbacks=checkpoint, Reduce LR on Plateau, Early Stopping  Image dimensions = 120,120 | 49% | 13s | The kernel did not crash.  The training accuracy = 95%, the model did not perform well. |

Note:

* The total time for experiment does not matter much as we are using the callbacks for optimizing the run. The training will stop if the targets are hit, hence in the table above only time per epoch is mentioned.
* Tried the image resolution 120x120 and 80x80, the accuracy dropped for 80x80 while there was only less difference in the time taken – 3sec per step. Accuracy is considered higher priority than time hence switched back to 120x120. No change in accuracy for 180x180 beyond that the kernel crashed.
* Activation function **sigmoid, tanh** for the 2 layered networks did not overfit, hence dropped.
* The data augmentation was not done initially, after learning that the model is not learning beyond 65% accuracy the data augmentation techniques were applied.

# Solution 2: CNN + RNN

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Experiment Type | Details | Accuracy | Time | Analysis. |
| Ablation Experiment |  |  |  |  |
| Overfitting Intentionally |  |  |  |  |
| Final Experiment without Batch Processing and Dropouts |  |  |  |  |

# Conclusion