GESTURE RECOGNITION CASE STUDY

# PROBLEM STATEMENT

As a data scientist at a home electronics company which manufactures state of the art smart televisions. We 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.

|  |  |
| --- | --- |
| GESTURE ACTION | |
| THUMBS UP | **INCREASE THE VOLUME** |
| THUMBS DOWN | **DECREASE THE VOLUME** |
| LEFT SWIPE | **JUMP 10 SECONDS BACK** |
| RIGHT SWIPE | **JUMP 10 SECONDS AHEAD** |
| STOP | **PAUSE THE MOVIE** |

# OBJECTIVE

Our job is to train several models in the 'train' folder to predict the action taken in each sequence of video, and to see which models perform well in the 'val' folder. The final test folder has been reserved for review; the final model's performance will be evaluated on the 'test' set.

# POSSIBLE ARCHITECTURE

1. 3D Convolution Neural Network
2. CNN + RNN Architecture

# DATA PRE-PROCESSING

1. Images were resized and cropped. This was done primarily to guarantee that the NN only detects the motions and does not focus on the background noise in the image.
2. The photos have been normalized. Normalizing an image's RGB values can sometimes be a quick and easy approach to remove distortions produced by lights and shadows.

# NN ARCHITECTURE AND DEVELOPMENT THROUGHOUT THE PROJECT

1. We started off with creating a base model which had 4 Convolution Layers, 2 Dense Layers and One Softmax Layers.
2. Our first aim was to figure out the batch size and number of the frames that should be taken for the model. We tried the following combinations:
   1. Frames as 16 and Batch Size as 32
   2. Frames as 30 and Batch Size as 64
   3. Frames as 16 and Batch Size as 32
   4. Frames as 30 and Batch Size as 64
3. Once we chose the frame and Batch Size, we moved to the Model Building Process.
4. We tried different combinations of the model with different combinations of hyperparameters. We played with the number of layers, number of filters, number of dense layers, the learning rate, optimizer etc.
5. Then, since we weren’t getting the desired result we went with the CNN Architecture.
6. Finally, we tried the MobilNet Architecture as Transfer Learning Architecture where we got appreciable result.

# OBSERVATIONS

1. Based on the facts, we concluded that batch size has no effect on training duration but does have an effect on the number of frames. We will choose a batch size of 64 because it appears to be the most efficient. It is mentioned in the Jupyter notebook that because it takes a long time to run, making it difficult for the notebook to run entirely, hence we used the videos with 16 frames.
2. The model overfitted until we came up with transfer learning. Maybe, the weights weren’t getting properly initialized and the model was too complex to get the proper weights.
3. The transfer learning approach helped us in overcoming the overfitting problem and the predefined weights associated with the transfer learning architecture helped us in getting the desired result.

# MODEL PERFORMANCES

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **MODEL** | **EXP NO** | **RESULT** | **P** | **TRAINING ARAMETERS** | **DECISION** |
| **Conv3D**  **CNN + LSTM** | 1 | Training Accuracy: 93.58% ValidationAccuracy:50% | 891,833 | | The model overfits due to which we need to change hyperparameters. |
| 2 | Training Accuracy: 42.24%  Validation Accuracy: 62.5% | 891,833 | | The accuracy increased but this time model was underfitting |
| 3 | Training Accuracy: 74.86%  Validation Accuracy: 62.5% | 890,853 | | The model achieved decent accuracy but there is scope of improvement |
| 4 | Training Accuracy: 52.9% Validation Accuracy:37.5% | 890,853 | | We tried with RMSprop optimizer but no improvement. |
| 5 | Training Accuracy: 47%  Validation Accuracy: 37% | 890,853 | | We tried increasing the number of layers and filters in the Convolution Layers |
| 6 | Training Accuracy: 68%  Validation Accuracy: 38% | 2,443,045 | | We tried tuning the hyperparameters. |
| 7 | Training Accuracy: 74%  Validation Accuracy: 62.5% | 3,115,397 | | We combined CNN + RNN to get the result and got decent results. |
| **MobileNet** | 8 | Training Accuracy: 81.81%  Validation Accuracy: 87.5% | 1,413,125 | | We applied mobilenet and our accuracy increased and gave us a good result. |
| **(Transfer** |
| **Learning +** |
| **LSTM)** |

**SCOPE OF IMPROVEMENT**

* 1. **Deeper Data Understanding**: The video clips were shot in a variety of settings, including varied backdrops, lighting, people, and cameras. Further investigation of the existing photographs may yield further information about them and increase the variety of the dataset. This additional knowledge may be used to the model's advantage within the generator function, increasing its stability and accuracy.
  2. **Using Transfer Learning**: Identifying the initial feature vectors with a pre- trained ResNet50/ResNet152/Inception V3 and sending them to an RNN for sequence information before passing them to a softmax layer for gesture categorization. (This was done, but owing to a shortage of time and disc capacity in the Google Colab platform, other pre-trained models could not be evaluated.)
  3. **Tuning hyperparameters**: Experimenting with different hyperparameter combinations such as activation functions (ReLU, Leaky ReLU, mish, tanh, sigmoid), various optimizers such as Adagrad() and Adadelta() can assist design better and more accurate models. Experimenting with other combinations of hyperparameters such as filter size, paddings, stride length, batch normalization, dropouts, and so on can help boost performance even further.