Gesture recognition case study

Problem Statement

You want to develop a cool feature for a home electronics company which manufactures state of the art smart televisions, in which the smart-TV should be able to recognize five different gestures performed by the user which will help users control the TV without using a remote.

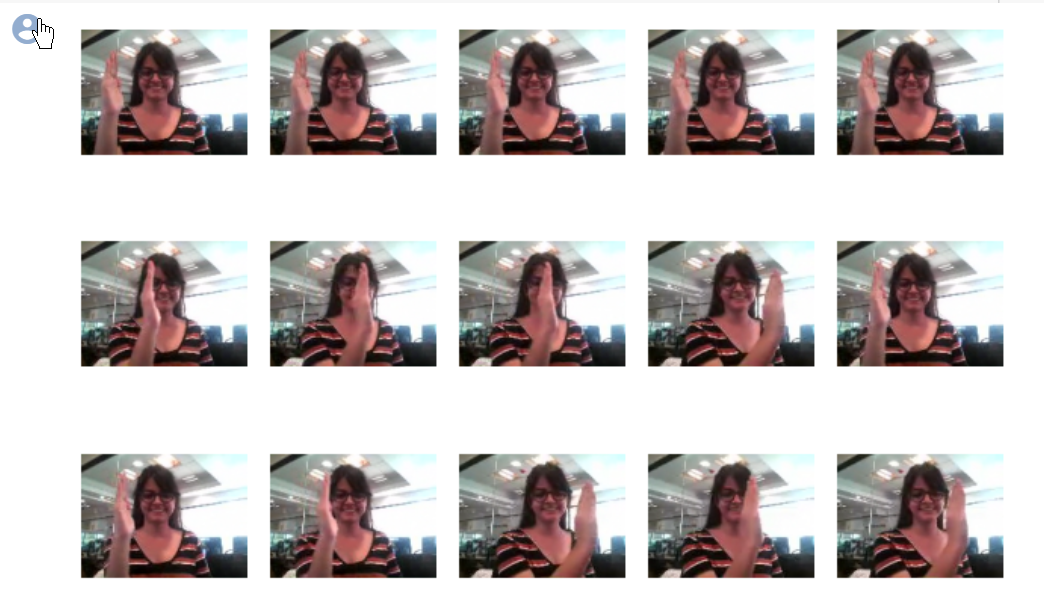
The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

* 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

Understanding the 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(images)**.

Example:



# Models with Conv3D:

Three template models are created with different combination of layers.

* conv3D\_template\_1: Total 4 convolution layers which includes 1 input layer and 3 hidden layers. All the 3 hidden conv layers are with batch normalization and max polling. Followed by flatten and two dense layers.
* conv3D\_template\_2: Total 5 convolution layers which includes 1 input layer and 4 hidden layers. All the 4 hidden conv layers are with batch normalization and max polling. Followed by flatten and two dense layers.
* conv3D\_template\_3: Similar to conv3D\_template\_1 version but change in position of dropout

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Model-type** | **Result** | **Decision + Explanation** | **Trainable Parameters** |
| **1** | * conv3D\_template\_1 * frame size=10, * image dimensions (100\*100) * kernal filters [16,32,64,128] * dropout = 0.15 * epoch = 20 * batch size = 20 | Training accuracy: **98.79%**  Validation accuracy: **71%** | Increase the number of frames and check if overfitting reduces | 2669125 |
| **2** | * conv3D\_template\_1 * frame size=15, * image dimensions (100\*100) * kernal filters [16,32,64,128] * dropout = 0.15 * epoch = 20 * batch size = 20 | Training accuracy: **99.25%**  Validation accuracy: **60%** | Overfit further increased. I will increase image dimensions to 120\*120 | 2669125 |
| **3** | * conv3D\_template\_1 * frame size=15, * image dimensions (120\*120) * kernal filters [16,32,64,128] * dropout = 0.15 * epoch = 20 * batch size = 20 | Training accuracy: **99.70%**  Validation accuracy: 59**%** | Still overfitting.  Lets try increasing the number of hidden layers | 3996229 |
| **4** | * conv3D\_template\_2 * frame size=20, * image dimensions (120\*120) * kernal filters [16,32,64,64,128] * dropout = 0.15 * epoch = 20 * batch size = 20 | Training accuracy: **99.25%**  Validation accuracy: 57**%** | Still no changes. Lets try increasing the image dimension to 160\*160 | 2043013 |
| **5** | * conv3D\_template\_2 * frame size=15, * image dimensions (160\*160) * kernal filters [16,32,64,64,128] * dropout = 0.15 * epoch = 20 * batch size = 20 | Training accuracy: **99.40%**  Validation accuracy: **74.89%** | Performance improved a bit. Let reduce the image dimension and try reduce the number of parameters | 3,714,181 |
| **6** | * conv3D\_template\_2 * frame size=25, * image dimensions (100\*100) * kernal filters [16,32,64,64,128] * dropout = 0.15 * epoch = 20 * batch size = 20 | Training accuracy: **98.79%**  Validation accuracy: **71%** | No change in the performance. Lets create a new template model. | 1,617,029 |
| **7** | * conv3D\_template\_3 * frame size=15, * image dimensions (120\*120) * kernal filters [16,32,64,128] * dropout = 0.15 * epoch = 20 * batch size = 30 | Training accuracy: **98.19%**  Validation accuracy: **34%** | Clear overfit. Let’s try with a different architecture | 15,945,093 |
|  |  |  |  |  |

# Models with Conv2D and RNN:

Fours template models are created with different combination of layers.

Transfer learning:

* ResNET50
* Vgg16
* resNet\_RNN\_template\_1: ResNET50 with 2 conv2d layers with batch normalization and maxpooling followed by flatter with GRU or LSTM and two dense layers.
* resNet\_RNN\_template\_2: ResNET50 with 1 conv2d(64 filter) layer with batch normalization and maxpooling followed by flatter with GRU or LSTM and two dense layers.
* resNet\_RNN\_template\_3: Similar to resNet\_RNN\_template\_2 version but with conv2d(16 filter) layer.
* vggNet\_RNN\_template\_4: VggNET16 with 1 conv2d(16 filter) layer with batch normalization and maxpooling followed by flatter with GRU or LSTM and two dense layers.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Model-type** | **Result** | **Decision + Explanation** | **Trainable Parameters** |
| **1** | * resNet\_RNN\_template\_1 * frame size=15 * gru - True * gru units - 256 * dense units - 128 * image dimensions = (100\*100) * ResNet training = False * dropout = 0.15 * epoch = 30 * batch size = 50 | Training accuracy: **100%**  Validation accuracy: **62%** | Still overfitting. Lets try with training few of ResNet50 layers | 24,428,069 |
| **2** | * resNet\_RNN\_template\_1 * frame size=15 * gru - True * gru units - 256 * dense units - 128 * image dimensions = (100\*100) * ResNet training = True * dropout = 0.15 * epoch = 30 * batch size = 30 | Training accuracy: **89.14%**  Validation accuracy: **62%** | No performance change. Lets reduce the number of hidden layers. | 10,883,621 |
| **3** | * resNet\_RNN\_template\_2 * frame size=15 * gru - True * gru units - 256 * dense units - 128 * image dimensions = (100\*100) * ResNet training = False * dropout = 0.15 * epoch = 30 * batch size = 30 | Training accuracy: **97.13%**  Validation accuracy: **93%** | Performance improved. Lets try to improve further by changing hidden layer conv2d filters from 64 to 16 kernals. | 24,999,365 |
| **4 Final Model** | * resNet\_RNN\_template\_3 * frame size=10 * gru - True * gru units - 256 * dense units - 128 * image dimensions = (120\*120) * ResNet training = False * dropout = 0.15 * epoch = 30 * batch size = 20 | Training accuracy: **99.25%**  Validation accuracy: 9**6%** | Performed improved. | 24,077,621 |
| **5** | * resNet\_RNN\_template\_3 * frame size=10 * gru - True * gru units - 256 * dense units - 128 * image dimensions = (120\*120) * ResNet training = True * dropout = 0.15 * epoch = 30 * batch size = 20 | Training accuracy: **98.19%**  Validation accuracy: **69%** |  | 10,533,173 |
| **6** | * vggNet\_RNN\_template\_4 * frame size=10 * gru - True * gru units - 256 * dense units - 128 * image dimensions = (120\*120) * vggNet training = False * dropout = 0.15 * epoch = 30 * batch size = 20 | Training accuracy: **98.94%**  Validation accuracy: **88%** | With vgg16, the performance is looks good. This model can further be improved. | 15,033,461 |

Because of upload size limitation, .h5 model is uploaded to https://drive.google.com/file/d/183Fau-\_voGfShM6qHesvbev0oCzIWBUC/view?usp=drive\_web