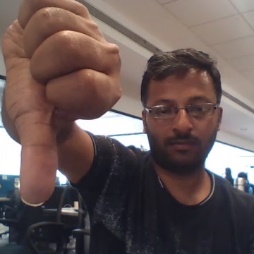
s

Group Case Study  
Deep Learning – Gesture Recognition

A circuit board

Description automatically generated

**Team Members**  
Palmari Sivasubramanian  
Bavani Ennares Maniammai

Stop

Thumbs Down

Thumbs Up

Right Swipe

Left Swipe

***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.

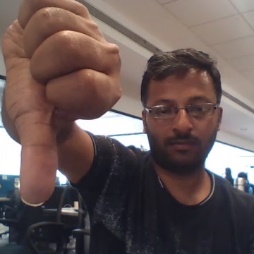
The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

* 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).

***UNDERSTANDING THE DATASET:***

* The training data consists of a few hundred videos categorized into one of the five classes. Each video (typically 2-3 seconds long) is divided into a **sequence of 30 frames(images)**. These videos have been recorded by various people performing one of the five gestures in front of a webcam - similar to what the smart TV will use.

Stop

Thumbs Down

Thumbs Up

Right Swipe

Left Swipe

* The data is in a zip file. The zip file contains a 'train' and a 'val' folder with two CSV files for the two folders. These folders are in turn divided into subfolders where each subfolder represents a video of a particular gesture. Each subfolder, i.e. a video, contains 30 frames (or images). Note that all images in a particulars video subfolder have the same dimensions but different videos may have different dimensions. Specifically, videos have two types of dimensions - either 360x360 or 120x160 (depending on the webcam used to record the videos). Hence, you will need to do some pre-processing to standardize the videos.
* Each row of the CSV file represents one video and contains three main pieces of information - the name of the subfolder containing the 30 images of the video, the name of the gesture and the numeric label (between 0-4) of the video.

***Two Architectures: 3D Convs and CNN-RNN Stack:***

* After understanding and acquiring the dataset, the next step is to try out different architectures to solve this problem.
* For analyzing videos using neural networks, two types of architectures are used commonly. One is the standard CNN + RNN architecture in which you pass the images of a video through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN. This is something you are already familiar with (in theory).
* The other popular architecture used to process videos is a natural extension of CNNs - a 3D convolutional network.
* *In this project, we tried both the architectures*

***Data Preparation – Image Preprocessing:***

***Import Necessary Libraries:***

* import numpy as np
* import os
* from scipy.misc import imread, imresize
  + To read the images from the disk by the generator function and to resize.
  + Data having different size of images. We tried running the model without resize. It shows size mismatch error and training did not happen
  + Later we imported this function to resize all the images to the same size
* import datetime
* import os
* from keras import backend as K
* import tensorflow as tf
* import random as rn
* from keras import backend as K
* from keras.models import Sequential, Model
* from keras.layers import Dense, GRU, Flatten, TimeDistributed, Flatten, BatchNormalization, Activation, Dropout,LSTM
* from keras.layers.convolutional import Conv3D,Conv2D, MaxPooling3D,MaxPooling2D
* from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau
* from keras import optimizers
* from keras.applications import ResNet50
* from keras.applications.resnet50 import preprocess\_input
* from keras.layers.recurrent import LSTM, GRU
* import keras
* import matplotlib.pyplot as plt

***Generator***

We are using generators because in this project, data is very large. We have images that runs in to GIGA BYTES (GB) and will not be able to load the entire data into the main memory for computation using keras fit method. To avoid that we have to use generators to bring that amount of data into main memory in which we can process. We no need feed entire data into main memory we can go batch wise

In this project we used custom generator.

**Sample Indexes:**

**img\_idx = [x for x in range(0, 30,2)]**

We have 30 images in each video which is quite high that leads to memory issue and slow execution training time. So, we are using sample indexes instead of using all 30 images

**Random Permutation:**

In CSV file we noticed, the videos belong to class1 are arranged sequentially in the first set followed by Class2, Class3, etc…

That kind of data biases network while training. To avoid such scenario, random shuffling as been done. So that it will not take the data belongs to particular class order.

An interesting thing to note here is the use of the infinite while loop. It is there in place so that the generator is always ready to yield a batch once next() is called once it is called at the start of training. Even after one pass over the data is completed (after the for loop is completed and the batch for the remainder datapoints is yielded), upon the subsequent next() call (at the start of the next epoch), the processing starts from the command 't=np.random.permuatation(folder\_list)'. In this way, the generator requires very less memory while training

**Resize Images:**

Data having different size of images. We tried running the model without resize. It shows size mismatch error and training did not happen

We cropped the image to 60\*60,

**imresize(image,(60,60))**

**Cropping Images:**

This is different to resize, while resize changes the aspect ratio of rectangular image. In cropping we will center crop the image to retain the middle of the frame.

**Normalization**

Photographs with poor contrast due to glare in the images will be taken care in the normalization

**temp = temp/127.5-1**

**Output:**

The generator able to take a batch of videos as input without any error. Steps like cropping, resizing and normalization performed successfully.

**Model:**

**Sample Conv 3D Model:**

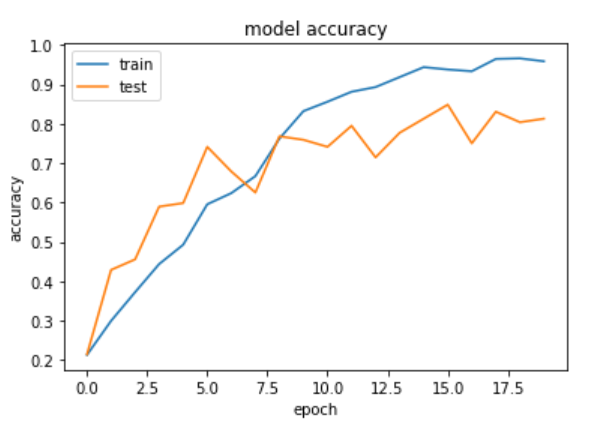


**Conv2D (LSTM + GRU):**



Developed multiple 3D and 2D models that is able to train without any errors. Many experiments have been done accuracy achieved. As suggested, start training on a small amount of data and then proceed further. Below are the experiment details.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| 1 | Conv2D (LSTM + GRU) | Accuracy: 0.54/Validation Accuracy: 0.32 | Tried with Affine, flipping the image and augmentation. But the batch size was high. |
| 2 | Conv2D (LSTM + GRU) | Accuracy: 0.99 /Validation Accuracy: 0.63 | It is overfitting on the training data. |
| 3 | Conv3D | No space left on device | Batch size 40, feature Map size 32 , epoch 30.  Tried with 1000, 5000 nodes in the Dense layer. This configuration generated the h5 files with each 500 MB. |
| 4 | Conv3D | ResourceExhaustedError: OOM when allocating tensor with shape [40,32,30,120,120] | Batch size 40, feature Map size 32 , epoch 30, cropped image size as 120\*120. The allotted resources exhausted and resulted in OOM error. |
| 5 | Conv3D | Accuracy: 0.61 | Cropped the image size to 120\*120, Batch size as 32 and with number of frames 30, Optimizer used is SGD. There are no dropouts in the convolution layer and FC layer |
| 6 | Conv3D | Accuracy: 0.71/ Validation Accuracy: fluctuating | Cropped the image size to 120\*120, Batch size as 32 and with number of frames 30, Optimizer used is SGD. Added dropouts in the convolution layer and FC layer. Used Learning rate as 0.1/0.01. Used Activation Relu. |
| 7 | Conv3D | Accuracy: 0.96 / Validation Accuracy: 0.65 | Cropped the image size to 60\*60, Batch size as 32 and with number of frames 15, Optimizer with Adam and learning rate as 0.001. Added dropouts and Batch Normalization in the layers. Used Activation Relu. |
| **Final Model** | **Conv3D** | **Accuracy: 0.95 / Validation Accuracy: 0.81** | **Cropped the image size to 60\*60, Batch size as 16 and with number of frames 15, Optimizer with Adadelta and learning rate as 0.001. Used 20 epoch. Added dropouts and Batch Normalization in both convolution and FC layers. Added dropouts in between the dense layers. Used Activation Relu.** |



**Experiment 8 Final Model using Conv 3D [Model 8]** gave us **test accuracy of 95% and validation accuracy of 81%** . The same model is submitted for the review.

**Submitted File:**

**H5 File Final**:

00020-0.11614-0.95833-0.71557-0.81250. h5

**Jupyter Notebook with Final 3D Conv Code**: Neural\_Network\_Gesture\_Recognition\_FinalModel

**Work Document**

Write-up