**Deep Learning – Gesture Recognition**

**Submitted by: Sandesh Kandagal and Neelima Boddapati**

**Problem statement (in brief):**

As Data scientists, we want to develop a feature in smart-TVs (home electronics) to recognize 5 different gestures performed by the user. This feature will help them to control the TV without using a remote.

|  |  |
| --- | --- |
| Gesture | Control |
| Thumbs up | Increase the volume |
| Thumbs down | Decrease the volume |
| Left swipe | 'Jump' backwards 10 seconds |
| Right swipe | 'Jump' forward 10 seconds |
| Stop | Pause |

**Understanding the data:**

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.

**Data preprocessing:**

The data generator will help in preprocessing the images of different dimensions in batches. Using the data generator data preprocessing steps such as image cropping, resizing and normalization were performed.

**Exploring different batch sizes:**

* Batch sizes: 32, 64, 128 and 256 were explored for model building.
* Observations:
  + The kernel repeatedly crashed when using Batch sizes 128 and 256 and hence were not used.
  + All models exhibited better performance with Batch size 32 than 64. Hence, batch size 32 is used in final model evaluation below.

**Modeling:**

The following models were built on the data

* Conv3D
* Conv2D + RNN LSTM
* Conv2D + GRU LSTM
* Transfer Learning imagenet +LSTM

**Parameters held constant:**

* Batch size: 32
* Epochs: 30
* Input shape: 15, 80, 80, 3
  + 15 (frames), 80 (width), 80 (height), 3 (channels)
* All models have consecutive layers and with Relu activation function in all layers
* Output layer: Dense (5) with Softmax activation function
* MaxPooling (2,2)
* padding='same'
* BatchNormalization

**Model summary:**

| **Model  Name** | **Model** | **Model parameters** | **Parameters** | **Result**  **(Best accuracy metrics)** | | | **Observations & Decision** |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Epoch** | **Training** | **Validation** |  |
| Model 1 | Conv3D | Hidden layers:  3 layers (**16, 32, 64**), Kernel size: (3,3,3)  FC layer: Dense (512)  Dropout: 0.5  Optimizer: **Adam** | 3,350,853 | 30 | 93.59 | 93.75 | **Observations:** The model performs well with good training and validation accuracy.  **Decision:** Explore if changing the optimizer (Model 2) improves accuracy and proceed with model building (Model 3) using the better optimizer. |
| Model 2 | Conv3D | Hidden layers:  3 layers (**16, 32, 64**), Kernel size: (3,3,3)  FC layer: Dense (512)  Dropout: 0.25  Optimizer: **SGD** | 3,350,853 | 24 | 99.32 | 81.25 | **Observations:** Changing the optimizer to SGD resulted in overfitting on train data and drop in validation accuracy.  **Decision:** Explore if increasing no of neurons while using Adam as optimizer (Model 3) will improve the accuracy. |
| Model 3 | Conv3D | Hidden layers:  3 layers (**32, 64, 64**), Kernel size: (3,3,3)  FC layer: Dense (512)  Dropout: 0.5  Optimizer: **Adam** | 3,449,157 | 24 | 93.53 | 81.25 | **Observations:** There is no improvement with increase in no of neurons.  **Decision:** Exploring if changing the kernel size (Model 4) will improve model performance. |
| Model 4 | Conv3D | Hidden layers:  3 layers (**16, 32, 64**), Kernel size: (**5,5,5**)  FC layer: Dense (512)  Dropout: 0.5  Optimizer: **Adam** | 3,606,437 | 23 | 74.4 | 68.750 | **Observations:** There is no improvement with changing the kernel size. **Decision:** Hence, the best Conv3D model is Model 1. Explore if Conv2d + LSTM will improve model performance. |
| conv2d1 | CONV2d +RNN LSTM | 3 Conv2D groups: (32, 64, 128)  Kernel size: (3,3)  LSTM:  LSTM (128)  Dense (64)  Dropout: 0.5  Optimizer = Adam | 6,917,029 | 29 | 81.60 | 56.25 | **Observations:** This model performance is poor compared to Conv3D models. Validation losses are fluctuating without a concurrent gain in validation accuracy.  **Decision:** Explore if CONV2d +GRU LSTM (Conv2d2) performs better. |
| conv2d2 | CONV2d +GRU | 3 Conv2D groups: (32, 64, 128)  Kernel size: (3,3)    GRU (128)  Dense (64)  Dropout: 0.5  Optimizer = Adam | 5,262,501 | 29 | 87.88 | 75 | **Observations:** The training and validation accuracy is better than the Conv2d1.  **Decision:** Explore if Transfer learning will produce better results. |
| conv2d3 | Transfer Learning imagenet +LSTM | Transfer learning using imagenet  LSTM (128)  Dense (64)  Dropout: 0.5  Optimizer = Adam | 3,831,877 | 11 | 84.48 | 81.250 | **Observations:** Achieved good accuracy on train and validation data owing to the transfer learning. It has very few trainable parameters (600,965) compared to Model 1 which has similar performance (3,350,629).  **Decision:** Will use this model as the final model for this dataset |

**Final decision:**

Considering a scenario wherein the company is limited by resources, we suggest to use the Transfer Learning imagenet +LSTM model with model parameters as identified in conv2d3 as the Final model because of it’s low trainable parameters and satisfiable performance.

Whereas in a scenario where the company is not limited by resources, we suggest the Convolution 3D model with model parameters as identified in model 1 as the Final model.