# Gesture Recognition Using Neural Networks

## Problem Statement

A home electronics company, which manufactures state of the art smart televisions, is looking to develop a feature that recognises five different hand gestures performed by the user via a webcam mounted on the TV, to control it without a remote.

The five hand gestures are as follows.

| **Gesture** | **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 |

The goal of this deep learning project is to build a neural network that can evaluate the hand gestures and map them to associated commands.

## Project Pipeline

The project is executed as per the following steps:

1. Data Reading/Data Understanding
2. Data Generator
3. Model Building & Training
   1. 3D Convolutional Neural Network
   2. 2D Convolutional Neural Network + Recurrent Neural Network
4. Model Selection

## Data Understanding

Upon inspection, it is seen that the training data is divided into 2 directories 'train' and 'val'. Each subdirectory of 'train' and 'val' contains 30 frames of videos of people making hand gestures. Videos either have dimensions of 360x360 or 120x160.

Each row of each CSV file represents 1 video and contains 3 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.

## Data Generator

The data generator is defined using two functions ‘process\_folder’ and ‘generator’, which perform the following operations:

1. ‘process\_folder’ - reads images in a folder, crops those with size 120x160, resizes and normalises them, and returns a set of image frames
2. ‘generator’ - packages sets of image frames into batches

The following is a sample of 2 sets of images of 3 frames from a batch of 8.



## Model Architectures

Two types of architectures are used for the task of classifying videos:

1. 3D Convolutional Neural Network
2. 2D Convolutional Neural Network + Recurrent Neural Network

### 3D Convolutional Neural Network

This architecture processes the videos using 3D convolutional layers and classifies them with dense layers.



### 2D Convolutional Neural Network + Recurrent Neural Network

This architecture processes the videos using time distributed 2D convolutional layers and classifies them using a recurrent layer.



## Model Building & Training

12 experiments with various model configurations of the 2 architectures are performed during the model building and training process. Experiments 1 through 5 test different layer configurations and hyperparameters of the 3D CNN architecture and experiments 6 through 10 test them for the 2D CNN + RNN architecture. Experiments 11 and 12 use different frame numbers and image sizes.

| **Architecture** | **Exp. No.** | **Result (Best Epoch)** | **Parameters** | **Explanation** |
| --- | --- | --- | --- | --- |
| 3D CNN | 1 | Training Accuracy - 0.8244  Validation Accuracy -0.5000 | 6,482,821 | Model 1 contains 2 3D convolutional layers and 2 dense layers. |
| 2 | Training Accuracy -0.8244    Validation Accuracy - 0.7188 | 2,903,045 | An additional 3D convolutional layer is added to model 1 to improve flexibility, but this results in overfitting. |
| 3 | Training Accuracy - 0.7247  Validation Accuracy - 0.5312 | 2,903,045 | A dropout layer is added to model 2 to counter overfitting, but validation accuracy decreases. |
| 4 | Training Accuracy - 0.7366  Validation Accuracy -0.4766 | 1,629,253 | The dropout layer is removed and an extra 3D CNN layer is added right after the first layer of model 3. This results in lower validation accuracy. |
| 5 | Training Accuracy - 0.6979  Validation Accuracy - 0.7188 | 1,424,453 | Model 5 is an experiment with the extra 3D CNN layer added after the second layer. Performance is average. |
| 2D CNN + RNN | 6 | Training Accuracy - 0.1801  Validation Accuracy -0.2109 | 2,428,789 | Model 6 contains two time-distributed 2D convolutional layers and 1 GRU layer. |
| 7 | Training Accuracy - 0.7470  Validation Accuracy -0.7812 | 1,132,533 | An additional time-distributed 2D convolutional layer is added to model 6. This results in good validation accuracy and a generalisable model. |
| 8 | Training Accuracy - 0.6786  Validation Accuracy - 0.7031 | 1,132,533 | A dropout layer to model 7 right before the GRU layer to counter overfitting, but validation accuracy decreases. |
| 9 | Training Accuracy - 0.6280  Validation Accuracy - 0.2500 | 1,133,493 | The dropout layer is removed and batch normalisation is added to all layers. This results in overfitting with just a few epochs and loss of generalisability. |
| 10 | Training Accuracy - 0.6741  Validation Accuracy - 0.7109 | 696,565 | Model 10 tests an additional CNN layer before the GRU layer in model 7. |
| 3D CNN | 11 | Training Accuracy - 0.7917  Validation Accuracy - 0.7031 | 2,084,853 | Model 11 uses an image size of 160x160 in the same configuration as model 7, but this does not improve performance. |
| 2D CNN + RNN | 12 | OOM Error during training | 1,175,797 | Model 12 uses 30 frames and images of size 160x160 in model 10, which has the lowest number of parameters. |

### Note

Early stopping is used in most cases because validation loss increases after a few epochs

## Model Selection

Model 7, which is a 2D CNN + RNN model with 1,132,533 parameters is the best performing model. Early stopping is used to isolate the best weights and the parameters are as follows:

* Training loss - 0.66613
* Training accuracy - 0.74702
* Validation loss - 0.66864
* Validation accuracy - 0.78125