# DEEP LEARNING COURSE PROJECT – GESTURE RECOGNITION

* Vivek Trivedi
* Naresh Kumar Kuppili

*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.

* 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

Link to dataset: <https://drive.google.com/uc?id=1ehyrYBQ5rbQQe6yL4XbLWe3FMvuVUGiL>

*Details of dataset:*

Essentially, 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 - like what the smart TV will use.

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. The videos have two types of dimensions - either 360x360 or 120x160

Each row contains - 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.

*Goal:* To train a model on the 'train' folder which performs well on the 'val' folder as well. Check the final model's performance will be tested on the 'test' set and **the model to recognize 5 hand gestures**.

For analyzing videos using neural networks, below architectures are used commonly.

*Two types of architectures*

* **3D convolutional network -** 3D convolutions are a natural extension to the 2D convolutions. In this case, the input to a 3D conv is a video represented as a sequence of 30 RGB images. Considering the shape of each image is 100x100x3, for example, the video becomes a 4-D tensor of shape 100x100x3x30 which can be written as (100x100x30)x3 where 3 is the number of channels.

Hence, deriving the analogy from 2-D convolutions where a 2-D kernel/filter (a square filter) is represented as (fxf)xc where f is filter size and c is the number of channels, a 3-D kernel/filter (a 'cubic' filter) is represented as (fxfxf)xc (here c = 3 since the input images have three channels). This cubic filter will now '3D-convolve' on each of the three channels of the (*100x100x30*) tensor.

* **CNN + RNN architecture -** Conv2D network is to extract a feature vector for each image, and a sequence of these feature vectors is then fed to an RNN-based network.

**Options:**

1. You can use transfer learning in the 2D CNN layer rather than training your own CNN

2. GRU can be a better choice than an LSTM since it has lesser number of gates (and thus parameters)

*Generators:*

To feed data to the model in batches. It is probably the most important part of building a training pipeline. For this project, we are going to write generators from scratch. To simplify a generator yields a batch of data and 'pauses' until the fit\_generator calls next(). Also, a generator requires very less memory while training.

*Data preprocessing:*

* cropping the images
* resizing the images
* normalizing the images

*Architecture detailing:*

* We have use hyperparameter tuning such as implementing the decaying learning rate, change in dropout rate and increase the batch size during the training process.
* We also used ***ReduceLRonPlateau*** when the validation loss got stagnated.
* We have used the preferred optimizer as **Adam**
* The preferred activation function is **ReLU**
* During experimentation we initially used to train on a small segment of data and gradually scaling up to entire data
* To avoid overfitting, we took measures to implement batch normalization, pooling, dropout layers.

*Observations*

* The time taken for processing is directly proportional to number of trainable parameters.
* Due to GPU limitations, the experimentation was performed using smaller batch size.
* Transfer learning helped in improving the accuracy of the model.
* Conv2D + LSTM with GRU cells seem to perform better than Conv3D.
* We observed underfitting problem through the phases of the experimentation.

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| 1 | Conv3D | Training accuracy: 0.9483 Validation accuracy: 0.4615  Total params: 11,200,901  Trainable params: 11,200,901  *The model is underfitting* | Used only 58 records were used for initial model validation experiment with batch size as 12.  The plots showcase that the training loss gradually reduced to minima whereas the loss doesn’t seem to get reduced on the validation data  *Next action*: adding batch normalization and dropouts |
| 2 | Conv3D | Training accuracy: 0.8448  Validation accuracy: 0.2692  Trainable params: 11,201,061  Non-trainable params: 160  *The model is underfitting.* | The model was still underfitting as the model doesn’t seem to perform well on the validation data.  *Next action*: Improve the model by adding more convolution layers. |
| 3 | Conv3D | Training accuracy: 0.7586  Validation accuracy: 0.1923  Trainable params: 20,800,709  Non-trainable params: 0  *The model is underfitting.* | We able to gradually reduce the loss on training and validation data but the model is still underfitting.  *Next action*: Increase the batch size and add more records for training |
| 4 | Conv3D | Training accuracy: 0.8235  Validation accuracy: 0.4444  *The metric has stopped improving* | The accuracy doesn’t seem to improve as expected so we can try reducing the learning rate  *Next action*: To reduce LR (0.01) on plateau after every 5 epochs. |
| 5 | Conv3D | Training accuracy: 0.2294  Validation accuracy: 0.2037  *The model is underfitting.* | As we were running the model on ablation data, maybe the accuracy metric dropped drastically on training and validation data  *Next action*: Running model with learning rate decay on full data and 10 epochs and batch size = 64 |
| 6 | Conv3D | Training accuracy: 0.2081  Validation accuracy: 0.2300  *The model is underfitting.* | *Next action*: Run model without learning rate decay for 10 epochs on complete dataset with batch size as 32 |
| 7 | Conv3D | Training accuracy: 0.8763  Validation accuracy: 0.3600  *The model is underfitting.* | *Next action*: Run for epochs = 20 |
| 8 | Conv3D | Training accuracy: 0.9442  Validation accuracy: 0.4100  *The model is underfitting.* | *Next action*: Try  LSTM and Conv2D |
|  |  |  |  |
| 1 | Conv2d + LSTM | Training accuracy: 0.9412  Validation accuracy: 0.4074  Trainable params: 11,429,413  Non-trainable params: 0  *The model is underfitting.* | Performing the training and validation on ablation data  *Next action*: Add batch normalization and dropouts |
| 2 | Conv2d + LSTM | Training accuracy: 0.8529 - Validation accuracy: 0.2963  *The model is underfitting.* | *Next action*: Run on full data with 30 epochs along with ReduceLROnPlateau |
| 3 | Conv2d + LSTM | Training accuracy: 0.6697  Validation accuracy: 0.2100  Trainable params: 999,813  Non-trainable params: 480  *The model is underfitting.* | *Next action*: Use ResNet to convolve and then using LSTM |
| 4 | ResNet + LSTM | Training accuracy: 0.3167  Validation accuracy: 0.2900  Trainable params: 1,266,661  Non-trainable params: 23,587,840  *The model is underfitting.* | *Next action*: Using 18 pictures from every video instead of 12 and with batch size 64 along with mobilenet model |
| 5 | MobileNet + LSTM | Training accuracy: 0.8959 Validation accuracy: 0.5800  Trainable params: 609,541  Non-trainable params: 3,230,912  *The model was performing better* | *Next action*: Use GRU with MobileNet |
| 6 | GRU with MobileNet | Training accuracy: 0.93967  Validation accuracy: 0.81  Trainable params: 1,053,701  Non-trainable params: 3,230,912  The model seems to be a good fit | The model was able to learn well on training data and perform optimally on validation data |
| Final Model | GRU with MobileNet |  |  |

*Summary:*

* After performing all experiments, we have observed that MobileNet with GRU performed better compared to other models used during experimentation phase