**Deep Learning Assignment**

**Neural Networks Project - Gesture Recognition**

**DSC32**

**Ashit Aggarwal**

**Shubhham Agarwal**

12th April 2022

# **Problem Statement**

## 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). Data can be accessed through the following link:

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

## What is 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.

**A picture containing text, person, group, people

Description automatically generated**

## Objective

Train a model on the 'train' folder which performs well on the 'val' folder as well (as usually done in ML projects). We have withheld the test folder for evaluation purposes - your final model's performance will be tested on the 'test' set

# **Model Solutioning**

For analysing 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).

## **3D Convolution Network**

3D convolutions are a natural extension to the 2D convolutions you are already familiar with. Just like in 2D conv, you move the filter in two directions (x and y), in 3D conv, you move the filter in three directions (x, y and z). In this case, the input to a 3D conv is a video (which is a sequence of 30 RGB images). If we assume that 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

As an example, let's calculate the output shape and the number of parameters in a Conv3D with an example of a video having 7 frames. Each image is an RGB image of dimension 100x100x3. Here, the number of channels is 3.

The input (video) is then 7 images stacked on top of each other, so the shape of the input is (100x100x7)x3, i.e (length x width x number of images) x number of channels. Now, let's use a 3-D filter of size 2x2x2. This is represented as (2x2x2)x3 since the filter has the same number of channels as the input (exactly like in 2D convs).

Now let's perform a 3D convolution using a (2x2x2) filter on a (100x100x7) matrix (without any padding and stride = 1). You know that the output size after convolution is given by:

A picture containing text

Description automatically generated

Diagram

Description automatically generated with medium confidence

Above figure is black and white with 1 channel. In our case we are using a colored image thus using 3 channels for the analysis

## **Convolutions + RNN Architecture**

The conv2D network will extract a feature vector for each image, and a sequence of these feature vectors is then fed to an RNN-based network. The output of the RNN is a regular softmax (for a classification problem such as this one)

A picture containing timeline

Description automatically generated

## **Data Preprocessing**

We will be using a custom generator that will help us in pre-processing the data and create the batch data. We will be using the yield command so that the he generator yields a batch of data and 'pauses' until the fit\_generator calls next(). This will be a part of the data ingestion pipeline which will be fed into our model

Our function will complete the following steps for the batch of images:

* number of images to be taken per video/sequence – This will help us in managing the parameter count as well as will help in accuracy
* cropping and resizing the images – This will help us understand the gesture better. Model can focus on the actual gesture rather than looking at the noise
* normalizing the images – This is for easy computation and will help us manage the distortions (outliers)

# **Model Development and Experiments**

* We have run the model with different architecture like image dimensions, batch size, normalization techniques, filter sizes, optimizer, activation function and number of layers.
* We used **Conv3D with RNN** too but there was not much activity we saw on validation accuracy
* **Early stopping** was used to put a halt at the training process when the val\_loss would start to saturate / model’s performance would stop improving. Due to this **the model after 5-6 epochs stopped working as the validation parameters did not improve**. Hence the Early Stopping was removed.
* We tried with different image size of 120x120 and 100x100. The model with image size 120x120 gave better performance.
* We tried with different normalization techniques in generator function – one was **dividing by 255 and the other was the percentile technique**. The models where normalization was done by dividing by 255 gave better results.
* Model development was tried on **various platforms** like local machine / NimbleBox/ Colab Pro / Kaggle. Kaggle gave the best performance while running the models
* We tried different **Optimizers** like Adam/ RMSprop / SGD. RMSprop gave the best results.
* We did our hands dirty with **Transfer Learning** too. We used different architectures like EfficientNetB0 and MobileNet. While EfficientNetB0 gave around 75% accuracy on validation data but MobileNet was far better with above 84% accuracy.
* We tried with **different architectures** with different number of layers for Conv3d along with different number of neurons in dense layers.
* We tried different **activation functions** like RELU / LeakyRelu / ELU. LeakyRelu gave the best results
* We also played around with **different learning rates** and ReduceLROnPlateau was used to decrease the learning rate if the monitored metrics (val\_loss) remains unchanged in between epochs.
* We used **dropouts** extensively throughout the model to tackle overfitting and have used the batch normalization after different layers.
* Batch size the image sequence length impacts the GPU/CPU memory. A **large batch** size of 64 threw out of memory error while running the code.
* Similarly in some cases, 30 **frame sequence** may threw out of memory error. Hence, we used 15 frames per sequence. Although this reduced some accuracy
* We also experimented with the **order** of the activation function, batch normalization, dropouts and pooling. Order also gave different results for the same architecture.
* **Conv3D model has performed better** than the CNN + LSTM as per our architecture. Initial epochs suggested that the model was not learning however as the count increased the val accuracy jumped steeply
* In total we have tried a **minimum of 50 models** (which is a **very conservative figure**). We are representing only the best 5 models’ architectures. The other architectures are shared on GitHub. Link is: <https://github.com/Ashit-Aggarwal/Data-Science/tree/main/Deep%20Learning/Gensture%20Recognition/Other%20Models>

## **Observations**

We have compiled the list of models and the experiments along with the errors we received in the table below:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Model** | **Optimizer** | **Activation Function** | **Result** | **Decision + Explanation** |
| 1 | Conv3D | Adam | Relu | Out of Memory Error | Reduce the batch size and reduce the frame sequence length |
| 2 | Conv3D | Adam | Relu | Out of Memory Error | Change the architecture |
| 3 | Conv3D | Adam | Relu | Accuracy: 0.21 | Too much time for epoch. Change the architecture |
| 4 | Conv3D | Adam | LeakyRelu | Accuracy: 0.83 | Model is a good fit |
| 5 | Conv3D  (Augmentation) | Adam | LeakyRelu | Accuracy: 0.85 | Model is a good fit |
| 6 | ConvLSTM | RMSprop | LeakyRelu | Accuracy: 0.86 | Model is good |
| 7 | ConvLSTM | Adam | LeakyRelu | Accuracy: 0.82 | Slight Overfit |
| 8 | ConvGRU | RMSprop | LeakyRelu | Accuracy: 0.85 | Model is good |
| 9 | ConvGRU (Augmentation) | RMSprop | LeakyRelu | Accuracy: 0.86 | Model is good |
| 10 | EfficientNet | RMSprop | Elu | Accuracy: 0.35 | Underfit |
| 11 | MobileNet | RMSprop | Elu | Accuracy: 0.89 | Model is good |

Based on the above experiments we have choose the following model for the final notebook.

1. Conv3D model
2. CNN + LSTM
3. CNN + GRU
4. Transfer learning model
5. Conv3D with Data Augmentation

We have run the above 3 models for 40 epochs each to get the results. We are saving the h5 file for the following model:

**Conv3D with Data Augmentation** as it gave a very **good performance** and had the **least parameters** of all the models.

(GRU with Data Augmentation and MobileNet were not chosen due to model size.)