Project Based Learning:

Deep Learning Course Project- Gesture Recognition

# Project Overview:

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.

# Gestures & Actions

* 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.

# Objectives:

* To build different models on the 'train' folder sequence of images to predict the action performed in each sequence or video.
* The model should also perform well on the 'val' data.
* Implement generator function, for generating batch data for model for training.
* Identify suitable Batch size, image dimensions, and number of frames to train the model

# Requirements/Task(s):

Two types of architectures are suggested to use for analyzing videos using deep learning.

* **3D Convolutional Neural Networks (Conv3D).** It is a natural extension of CNNs - a **3D convolutional network**.
* **CNN + RNN architecture:** We 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.

# Data Understanding

# 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. Data set can be downloaded from below link.

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

# Sample data:

# 

# Data Preprocessing: Performed in batch generator code block.

# Resizing and cropping of the image:

# To ensure that we capture most of the target area in frame rather than the background noise. For this we experimented with various size dimensions for training the model, to get the best suitable values.

# Normalization of the images:

# To get rid of distortions caused by lights and shadows in an image.

# Data Generator:

# This is one of the most important parts of the code. In the generator, we pre-processed the images as we have images of 2 different dimensions (360 x 360 and 120 x 160) as well as create a batch of video frames. The generator should be able to take a batch of videos as input without any error.

# We ensured that the sequence of frames should be preserved during batch generation as, altering the sequence of actions may change the meaning of action.

* We experimented with below parameters

Batch size, Image dimensions, Frame sequences.

* To reduce the computation complexity, we tried to reduce the number of frames in sequence. We considered only those frames which describes the intended action accurately.

(Using Alternate frames/Trimming the frames from beginning and end).

# Observations:

1. Batch size ∝ GPU memory / available compute. A large batch size can throw GPU Out of memory error/ResourceExhausted error, and thus here we had to play around with the batch size till we were able to arrive at an optimal value of the batch size.
2. Then we realized that larger batch size impacted the accuracy of model negatively. Though having higher batch size reduced the training time, but model tend to learn less due to large batch size.
3. Larger image dimensions gave more data points for training but, it also increased number of trainable parameters. Hence, we had to trade off with the image dimensions also.
4. Splitting train data into test and train reduced the model’s accuracy as we had lesser data for training. So, we later decided to split data for testing from validation data.
5. Data Augmentation greatly helped in overcoming the problem of overfitting which our initial version of model was facing.
6. CNN+LSTM based model with GRU cells had better performance than Conv3D.
7. Number of time distributed layers in CNN+LSTM also had impacted the accuracy of model a lot.
8. Using dropouts helped us in controlling overfitting.

Record your notes/research here:

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D: model\_1st** | Train accuracy: 0.96  Val accuracy: 0.18 | **Highly overfitting model**  **Try some experiment with batch size** |
| **2** | **Conv3D: model\_2nd** | **Model not trainable due to** ResourceExhaustedError | **Reduce the Batch size from 128 to 64 and increase the number of epochs** |
| **3** | **Conv3D: model\_3rd** | **Overfitted model, even after 34 epochs hence interrupted learning further.** | **Reducing size of image and number of frames by using alternate Frames. Also reduce 1 conv3D block from model.** |
| **4.** | **Conv3D: model\_4th** | Train accuracy: 0.84  Val accuracy: 0.27 | **Model still overfitting. Try**  **Trimming frames from beginning and end.** **and reduce image dimensions to 64 by 64.**  **Increase 1 conv3D block in model.** |
| **5** | **Conv3D: model\_5th** | **Not improved** | **Reduce batch size to 32**  **Increase Number of Epochs** |
| **6** | **Conv3D: model\_6th** | Train accuracy: 0.82  Val accuracy: 0.41  Overfitting reduced a little | **Try CNN- LSTM architecture for Model** |
| **7** | **ConvLSTM: lstm\_model\_1st** | **Test model** | **Due to too many training parameters, it was taking too long to train. So, tried to reduce parameters.**   1. **Reduce image size** 2. **Reduce number of frames** 3. **Reduce number of Time distributed layers from model.** |
| **8** | **ConvLSTM: lstm\_model\_2nd** | **No improvement** | **Try ConvGRU as ConvLSTM is not giving desired accuracy** |
| **9** | **ConvGRU: GRU\_model\_1st** | Train accuracy: 0.53  Val accuracy: 0.41  Overfitting gone but model is taking too long to train due to lots of parameters. | **Try some data augmentation**  **Reduce batch size to 30** |
| **10th** | **ConvGRU:**  **GRU\_model\_2nd** | **Model seemed performing little better than earlier** | **As this model has only 1 Time distributed +2DConv block, we decided to increase the number of such blocks to 4.**  **Also to trim frames from end.** |
| **11th** | **ConvGRU:**  **GRU\_model\_3rd** | Train accuracy: 0.90  Val accuracy: 0.68 | **Reduce size of images to 100 \* 100**  **Use alternate frames from sequence** |
| **Final Model** | **ConvGRU:**  **GRU\_model\_4th** | Train accuracy: 0.82  Val accuracy: 0.83 | **Test the mode with test data**  **Test accuracy observed :83** |

# The details of the final Model

##### Total train data available in Train.csv have been used for training the model. The Validation data has been spitted into 2 halves, one for validation and other half is used for calculating test accuracy

##### Parameters for generator are:

* image\_height=100
* image\_width=100
* BATCH\_SIZE=30
* IMG\_IDX= [0,2,4,6,8,10,12,14,16,18,20,22,24,26,28] i.e., Alternate frame has been used for training. Hence number of frames in a sequence is 15.
* For the final model Generator with Augmentation has been used where we have performed some wrap affine transformation, cropping and restoring image to specified size

##### Parameters for Model training are:

* NUM\_EPOCHS=30
* CHANNELS=3
* KERNEL\_SIZE= (3,3)

##### Model Architecture is

* **CNN+ GRU** with Total params: 1,000,293 and **Trainable params: 999,813** and non-trainable params: 480

Table

Description automatically generated

# Conclusion:

# Further, more data availability and preprocessing can increase the performance of model

# Contributors

Santosh

Rudranil Gupta