**Gesture Recognition – Case Study**

* Rajesh Sikdar (Group Facilitator)
* Prashant Bande

**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.

**Dataset**

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.

**Suggested architecture**

* Conv3D - 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 100 x 100 x 3, for example, the video becomes a 4D tensor of shape 100 x 100 x 3 x 30 which can be written as (100 x 100 x 30) x 3 where 3 is the number of channels. Hence, deriving the analogy from 2D convolutions where a 2D kernel/filter (a square filter) is represented as (f x f) x c where f is filter size and c is the number of channels, a 3D kernel/filter (a 'cubic' filter) is represented as (f x f x f) x c (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 (100 x 100 x 30) tensor.
* CNN+RNN - 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.

**Data Generator**

* Resizing/Cropping image – it is done to recognizes the gestures effectively rather than focusing on the other background noise.
* Normalization of Image – Normalized images to get rid of distortions caused by lights and shadows in an image.
* Additionally – Data augmentation – Images are slightly rotated and pre-processed in order to bring in more data for the model to train on and to make it more generalizable.

**Observations and Understanding**

* Number of parameter increase, add more complexity in learning process
* When tried with the large batch size faced memory issue.
* Increasing the batch size greatly reduces the training time but this also has a negative impact on the model accuracy.
* Data Augmentation and Early stopping greatly helped in overcoming the problem of overfitting which our initial version of model was facing.
* Conv3D model with data augmentation and reduce parameters perform well and solve the problem of overfitting. Accuracy and validation accuracy converges after certain set of epochs and received the good result. Training the model for more epochs will help to achieve the great result.

**Analysis Table**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Experiment No.** | **Model** | **Parameters** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D1** | **1,117,061** | **Acc:0.98**  **Val\_Acc:0.27** | **Network overfitted.** |
| **2** | **Conv3D1 with dropout** | **3,638,981** | **Acc:0.77**  **Val\_Acc:0.26** | **Increase the amount of trainable data, reduce batch size.** |
| **3** | **Conv3D3** | **1,762,613** | **Acc:0.71**  **Val\_Acc:0.22** | **Reduced filter size, Imaged size, Increase batch size. Still overfitted** |
| **4** | **Conv3D4** | **2,556,533** | **Acc:0.77**  **Val\_Acc:0.26** | **Added dropout at dense layer, No improvement. Still overfitted.** |
| **5** | **Conv3D5** | **2,556,533** | **Acc:0.83**  **Val\_Acc:0.21** | **Added dropout after conv. No great improvement. Still overfitted.** |
| **6** | **Conv3D6** | **696,645** | **Acc:0.78**  **Val\_Acc:0.20** | **Reduce Parameters. Still overfitted.** |
| **7** | **Conv3D7** | **504,709** | **Acc:0.81**  **Val\_Acc:0.27** | **Reduced parameters again. Still overfitted.** |
| **8** | **RNNCNN1** | **1,657,445** | **Acc:0.94**  **Val\_Acc:0.35** | **Little improvement in RNN\_LSTM cells. Still overfitted.** |
| **9** | **Conv3D9** | **504,709** | **Acc:0.75**  **Val\_Acc:0.73** | **With data augmentation, and reduced set of parameters, network improved after certain stage. Got decent accuracy and finally overfitting controlled.** |
| **10** | **RNNCNN2** | **2,573,925** | **Acc:0.94**  **Val\_Acc:0.83** | **With augmented data, CNN+LSTM Cells, got improved accuracy and val\_accuracy, It is the 2nd model which performs better.** |
|  |  |  |  |  |
| **Final Model** | **Conv3D9** | **504,709** | **Acc:0.75**  **Val\_Acc:0.73** | **With data augmentation, and reduced set of parameters, network improved after certain stage. Got decent accuracy and finally overfitting controlled.** |