**REPORT ON RESULTS OF DEEP LEARNING MODELS**

**– GESTURE PREDICTION**

**Problem Statement**

1. A home electronics company which manufactures state of the art smart televisions, wants to develop a cool feature for a smart TV which is to recognize 5 different hand gestures which helps users control the TV without remote control.
2. 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

**Details of the dataset**

1. There are 663 videos recorded for training, each video is divided into 30 frames. There are 100 videos of 30 frames each to validate the accuracy of the solution.

**Solution**

1. I carried out experiments with various Deep Learning Models for recognition of gestures from a series of images. The main types of models used were as follows :-
2. 3D Convolutional Model which used time as the third dimension.
3. 2D Convolutional model with an RNN(LSTM) layer for the time dimension.
4. The results of the experiments are tabulated below :-

|  |  |  |
| --- | --- | --- |
| **Experiment Number** | **Model + Parameters** | **Results / Explanations** |
| **1** | **Simple Conv3D**  **Batch Size – 10**  **Image Size 160x160**  **Frames - 30** | **These experiments were carried out with the aim of finding out time and memory requirement for various combinations of batch size, image size, frame count, etc. My observations for my specific system:-**  **1. With image size of 160x160, 30 frames and 50 epochs, the system crashed as it ran out of memory.**  **2. The ideal combination was found to be as follows :-**  **Batch Size – 40**  **Image size 120x120, use frame count 30** |
| **2** | **Simple Conv3D**  **Batch Size – 50**  **Image Size 160x160**  **Frames 20** |
| **3** | **Simple Conv3D**  **Batch Size – 40**  **Image Size 160x160**  **Frames 20** |
| **4** | **Simple Conv3D**  **Batch Size – 40**  **Image Size 120x120**  **Frames 30** |
| **5** | **Conv2D-LSTM Model** | **Total params: 1,001,317**  **Trainable params: 1,001,317**  **In 14 epochs, the model gives an accuracy of 45% on training data and 40% on Val data.  The results will certainly improve with more epochs.** |
| **6** | **Conv3D Model**  **Filter size (3,3,3)** | **Total params: 1,115,589**  **Trainable params: 1,115,589**  **This is a typical Conv3D model. We ran two with a filtersize of (3,3,3) and (2,2,2). Initially I had Batchnormalization layers. However the performance was very bad as the model overfitted the data.**  **Removal of Batch Normalization improved the models. However, there were large fluctuations of accuracy between epochs.** |
| **7** | **Conv3D Model**  **Filtersize (2,2,2)** |
| **8** | **Conv3D Model**  **Filtersize (3,3,3)**  **Removed Batchnormalization.** |
| **9** | **Large Conv3D Model**  **Filter size (3,3,3)** | **Total params: 2,556,533**  **Trainable params: 2,554,549**  **Increased the size of the model by adding more convolutional layers. This increases the total parameters and also increases the accuracy of the model. However, the models are larger and takes more time for training and inference.**  **This model had the better performance with Training Data Accuracy: 85%**  **Validation Data Accuracy: 68%**  **However, the model continued to overfit.** |
| **10** | **Large Conv3D Model with Dropout (in Conv Layer)** | **Total params: 6,749,493**  **Trainable params: 6,746,613**  **Since there is a tendency to overfit with a large model, I have added Dropout after the convolution layers to reduce overfitting and improve the accuracy on validation data and real world data. I have also increased the layers in each Conv3D layer for higher accuracy.**  **This model had the best performance with Training Data Accuracy: 93%**  **Validation Data Accuracy: 79%**  **There is still little overfitting.**  **However, it was the best model.** |
| **11** | **Conv3D Model with Reduced Parameters** | **Total params: 3,643,941**  **Trainable params: 3,643,525**  **Reduction in layers to reduce parameters leads to lower performance.**  **Accuracy achieved :-**  **Training Data Accuracy: 81%**  **Validation Data Accuracy: 75%**  **As can be seen, the parameters are nearly half of the large model. However, the performance has not reduced very much. The validation accuracy has only reduced by 4%.**  **Therefore, if model size is a consideration, this would be a good model to choose.** |

**Personal Observations**

1. I was down for two weeks with Covid. Luckily I recovered by mid of last week. Therefore I had 4-5 days for doing the project. Since I did the project in a compressed timeline, I faces a few issues. Based on this experience, I would like to give a few personal observations and suggestions regarding the project.
2. The project details says that “Use the correct container: 'Tensorflow 1.5.0 GPU Py3' “. This meant that I could not run the code on my laptop which has a GPU as the GPU and CUDA drivers in the laptop are the latest versions. However, Tensorflow 1.5 no longer works properly with the new drivers. I therefore wasted a lot of time trying to get the code to work. Finally, I converted the code to Tensorflow 2.4 and ran the code. The conversion by itself is not a problem. But the statement “use Tensorflow 1.5” made me waste a lot of time to get it to work unsuccessfully. So suggest that the starter code may be updated to Tensorflow 2.
3. This project needs a lot of experimentation. But the GPU credits available with Nimblebox is just not enough to do experimentation using GPU instance. And using CPU takes a lot of time, which became very evident when the overall time available was less. So, if possible, more credits may be provided on Nimblebox. If not possible, a discount for purchase of credits would also be welcome.