**The Gesture Recognition Assignment using Neural Networks**

We have used the below two different types of Neural Network Models as a solution for this.

1. Conv3D – Convolution 3D using Time Distributed and Max Pooling 3D. – Referred as Model 1
2. Conv2D with RNN and GRU – Convolution 2D in RNN and GRU with Time Distributed and Max Pooling 2D using Transfer Learning VGG16 – Referred as Model 2

Below is the Table which depicts the conclusion we made after each run of the Models and moved onto the next run with different varied parameters just to better the Model output and minimize the Loss.

|  |  |  |  |
| --- | --- | --- | --- |
| **Exp #** | **Model** | **Result** | **Decision + Explanation** |
| **1** | **Conv3D** | 1. **Error/warning:scipy.misc .imread deprecated** 2. **Error/warning:scipy.misc .** **imresize deprecated** | 1. **We started with scipy imread but encountered issues therefore changed to cv2.imread and fixed the issue.** 2. **Used cv.resize for image transformation** |
| **2** | **Conv3D** | **ValueError: could not broadcast input array from shape (360,360,3) into shape (360,360)** | **There was an issue in normalizing RGB images and fixed the code.** |
| **3** | **Conv3D** | **Encountered:**  ValueError: Error when checking input: expected conv3d\_1\_input to have shape (360, 360, 10, 3) but got array with shape (10, 360, 360, 3) | **We Changed Shape of input to the model as inputshape= (10,360,3,360)** |
| **4** | **Conv3D** | **Encountered:**  ValueError: could not broadcast input array from shape (120,120) into shape (360,360) | **We Changed the logic :**  **batch\_data = np.zeros((batch\_size,len(img\_idx),120,120,3)) as we resized the image to 120\*120 because of memory issues and improve the processing speed.** |
| **5** | **Conv3D** | **Encountered:**  Space issue as lot of models were saved. | **Deleted some of the Saved Models and also added Dropouts to Improve model performance and also to reduce space.** |
| **6** | **Conv3D** | **We ran the Model with initial Batch size 10 and Filter Size (2,2,2) and total Parameters around 3 Million. Training time was around 17 Seconds per Epoch.**  **Best Validation accuracy noted was around 21%.**  **Total Epoch size was 30.** | **We Increased the LR to 0.03 and went for another run** |
| **7** | **Conv3D** | **The losses went down but no significant improvement in validation accuracy.** | **We Reduced the learning rate LR to 0.005 and Ran the Model again** |
| **8** | **Conv3D** | **Some improvement in Accuracy and Loss were observed, but we felt that it can improve further.** | **We changed the First Conv3d,maxpool as (3,3,3) and next Conv3d and maxpool as (4,4,4)** |
| **9** | **Conv3D** | **Improvements in Accuracy and Loss were observed, to this we decided to go further with our tuning.** | **We changed the all the Conv3d layer and the maxpool as (4,4,4)** |
| **10** | **Conv3D** | **Some Improvements in Accuracy and Loss were observed but not much compared to the previous one. Accuracy was in the range of 55-60% and Loss was near to 2. The total parameters was 1.6 Million** | **We changed the all the Conv3d layer and the maxpool as (6,6,6)**  **We introduced Dropout and made it to 0.5** |
| **11** | **Conv3D** | **Some Improvements in Accuracy and Loss were observed but not much compared to the previous one. Best Accuracy was 66% and Loss was near to 1. The Total parameters was around 700K** | **We reverted the all the Conv3d layer and the maxpool back to (4,4,4)**  **With Dropout as 0.5** |
| **12** | **Conv3D** | **Some Improvements in Accuracy and Loss were observed but not much compared to the previous one. Best Accuracy was touching 70% and Loss was around .8 and .9.** | **We removed the LR as introduced earlier. Also increased the EPOCH to 50 from 30.** |
| **13** | **Conv3D** | **Accuracy was in the range of 70-75% Loss was less than 0.7** | **This was the best Accuracy and Loss that we observed. The Total Parameters count was 901,701.** |
| **14** | **Conv2D RNN with LSTM (Time Distributed)** | **Initial Run with Batch size 10** | **We implemented VGG16 for Transfer learning and Trained on last 4 layers of VGG16 with EPOCH 30** |
| **15** | **Conv2D RNN with LSTM (Time Distributed)** | **Accuracy with 21% and was not growing , the total parameters were more than 14 million and all were trainable, Loss was more than 8** | **We configured the VGG16 for Transfer Learning NON-Trainable and added our own Conv2D layers.** |
| **16** | **Conv2D RNN with GRU (Time Distributed)** | **Accuracy with 95% Plus but the Validation Accuracy was not above 68%** | **Changed from RNN LSTM to RNN GRU With Dense Layer as 256. Increased batch size to 16.** |
| **17** | **Conv2D RNN with GRU (Time Distributed)** | **Accuracy with 95% plus but the Validation Accuracy was not going above 75% Loss was around 2** | **Changed the Dense Layer to 128 and then to 64, increased EPOCH to 50** |
| **18** | **Conv2D RNN with GRU (Time Distributed)** | **Train Accuracy around 80% plus but the Validation Accuracy was around 70% with total Parameters were 15,545,285**  **Loss was less than 0.7** | **Used VGG16 for Transfer Learning with all non-trainable. With Dense Layer as 64 with our own conv 2D and maxpool layer. Size of H5 file was around 67MB** |

We observed that the Conv3D suited the solution more hence the final Model that we chose was **Conv3D**, below are the supporting points for our decision:

1. The Conv3D model had a Loss of around 0.67 and Validation and Categorical Accuracy both around 74%
2. The Model Size is around 10 MB as compared to the Conv2D RNN+GRU where it is around 67 MB. The Less Model size will easily fit into the Webcam for implementation.
3. The Total Parameters in case of Conv3D is less than 1 Million and is less than the Conv2D RNN+GRU model which is more than 15 million.

As we have built the code for **both** Conv3D and Conv2D RNN+GRU models in a single .ipnyb file we are submitting the below files along with this write-up:

1. One Python Code consisting of the Model Generator , both the Models, Model Summary and Plots - GestureReco\_NN\_Assig\_Conv3D\_2D+RNNvFinal.ipynb
2. H5 file for the selected model under Conv3d - model-CONV3D-00042-0.67197-0.73958-0.67851-0.74107.h5

|  |  |  |  |
| --- | --- | --- | --- |
| **Experiment Number** | **Model – 2D RNN with LSTM and GRU** | **Result** | **Decision + Explanation** |
| **01** | **RNN with LSTM (Time Distributed)** | **Accuracy with 21% and was not growing** | **Used VGG16 for Transfer learning and Trained on last 4 layers of VGG16** |
| **02** | **RNN with LSTM (Time Distributed)** | **Accuracy with 95% Plus but the Validation Accuracy was not above 68%** | **Used VGG16 for Transfer Learning NON-Trainable and added our own Conv2D layers.** |
| **03** | **RNN with GRU (Time Distributed)** | **Accuracy with 95% plus but the Validation Accuracy was not going above 75%** | **Used VGG16 for Transfer Learning NON-Trainable and added our own Conv2D layers. With Dense Layer as 256.** |
| **04** | **RNN with GRU (Time Distributed)** | **Accuracy with 95% plus but the Validation Accuracy was not going above 75%** | **Used VGG16 for Transfer Learning NON-Trainable and added our own Conv2D layers. With Dense Layer as 64.** |
| **05** | **RNN with GRU (Time Distributed)** | **Accuracy with no more than 22% and Validation Accuracy less than 20%** | **Used VGG16 for Transfer Learning last 8 layers Trainable. With Dense Layer as 64.** |
|  |  |  |  |