**Gesture Recognition – Deep learning**

**Problem Statement:**

We need 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.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Experiment No | Model | Hyper-parameters | Result | Params | Decision + Explanation |
| 1 | **Conv3D** | **Batch size: 200**  **Epoch: 10** | **OOM** | **-** | **Reduce batch size** |
| 2 | **Conv3D** | **Batch size: 32**  **Epoch: 10**  **LR: 0.01**  **Dim: 120\*120** | **Training accuracy** : 0.20  **Validation accuracy :** 0.2039 | **Total params:** 12,904,581  **Trainable params:** 12,904,581  **Non-trainable params:** 0 | **Model validation loss and training loss is decreasing trend.The Training accuracy and validation accuracy is approx same. The model architecture is good the model is learning but can learn faster if we add some extra conv3d layers and kernel size was made in 3 dimensional.** |
| 3 | **Conv3D** | **Batch size: 32**  **Epoch: 10**  **LR: 0.01**  **Dim: 120\*120** | **Training Accuracy:** 0.2009  **Validation accuracy:** 0.1562 | **Total params:** 28,562,437  **Trainable params:** 28,562,437  **Non-trainable params:** 0 | **Model is clearly not learning after adding layers of conv3d. But surprising part is the graph in notebook shows validation accuracy is greater than training accuracy.**  **Its usually mustn’t be there but people say this could be the case of underfitting few suggested to add dropout might solve this issue (Source:** [**Vali accuracy vs Traning accuracy**](https://qr.ae/pvEERe)**)hence adding drop outs ,batch normalization** |
| 4 | **Conv3D** | **Batch size: 32**  **Epoch: 10**  **LR: 0.01**  **Dim: 120\*120**  **Dropout: 0.5** | **Training accuracy :** 0.89  **Validation accuracy** : 0.3047 | **Total params:** 22,732,549  **Trainable params:** 22,730,629  **Non-trainable params:** 1,920 | **Model is overfitting .Reduce dropout** |
| 4 | **Conv3D** | **Batch size: 32**  **Epoch: 10**  **LR: 0.01**  **Dim: 120\*120**  **Dropout: 0.2** | **Training accuracy :**0.91  **Validation accuracy** : 0.35 | **Total params:** 22,732,549  **Trainable params:** 22,730,629  **Non-trainable params:** 1,920 | **Model is still overfitting .Replacing with global average pooling instead of flatten.**  **Why Global average pooling ?**  **Global average pooling was used in intension to reduce parameters after**  **experiments with dropouts.**  **No of params : around 24lakh .**  **This is very complex model.**  **Our goal is to get best possible accuracy with least params.**  **Also global average pooling is used when model is overfitting** |
| 5 | **Conv3D** | **Batch size: 32**  **Epoch: 50**  **LR: 0.01**  **Dim: 120\*120**  **Dropout: 0.2** | **Training accuracy :** 0.90923  **Validation accuracy :** 0.86 | **Total params:** 712,453  **Trainable params:** 710,533  **Non-trainable params:** 1,920 | **This is a very good model. Best one achieved after lot of try.Lets try different architecture** |
| 6 | **GRU+Time distributed+Conv2d** | **Batch size: 32**  **Epoch: 50**  **LR: 0.01**  **Dim: 120\*120** | **Training Accuracy :** 0.85714  **Validation accuracy :** 0.81250 | **Total params:** 99,845  **Trainable params:** 99,269  **Non-trainable params:** 576 | **This very good model .No overfitting.Trainable params are reduced as well with LSTM instead of GRU** |
| 7 | **LSTM+Time distributed** | **Batch size: 32**  **Epoch: 50**  **LR: 0.01**  **Dim: 120\*120** | **Training Accuracy :** 0.80357  **Validation accuracy :** 0.78125 | **Total params:** 13,781  **Trainable params:** 13,589  **Non-trainable params:** 192 | **Parameters reduced but validation accuracy dropped by 3% than GRU.But model is good its not overfitting and model is simple as well with just 13k params that a good number** |

**Learnings :**

Apart from listed models there were many models which failed with Negative dimension error when kernel size was tried to put as 3dimensional as it was explored 3d kernel size might fit well.

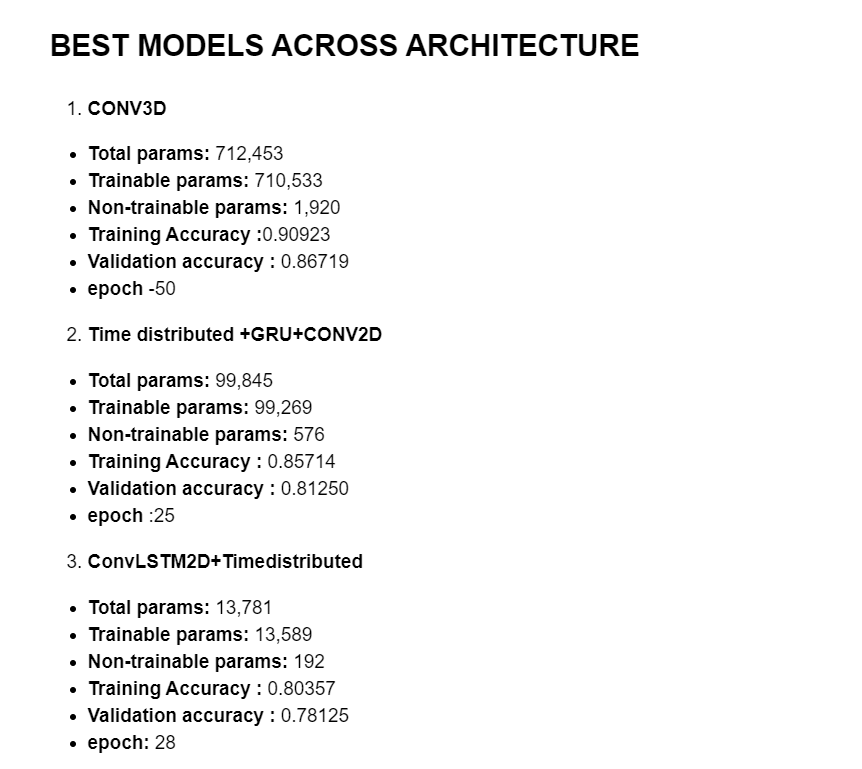
This was solved by adding padding =’same’

Initially the image size was chosen as 160\*160 but the models with this size accuracy and training accuracy in 10 epochs and architecture as 1st model simple ConV3d gave almost same accuracy and saturated on 17-18% just like 120\*120 Experiment 1 model . Hence it was decided that image size reduction to 120 \*120 didn’t cause depletion of model accuracy. Hence we kept all models with same dimension 120\*120.

Interesting point was on Conv3d was when drop out 0.5 (50% removal) was introduced the training accuracy increased within 10 epoch but validation accuracy remained same.This gave clarity that first 0.5 dropout was too much that made model overfit.

Another interesting point found was with flatten layer same architecture without drop out yet trainable parameters were 28lakh after adding dropout reduced to 22lakh but when global average pooling was introduced with drop out reduced to 0.2 the number of params reduced to 7lakh..this shows the beauty of Global average pooling to be noted previous layer of global average pooling had 256 tensor layer of conv3d.

**CONCLUSION :**



The best model chosen is **Time distributed +GRU+CONV2D**

The process of evaluation: Among the best listed Convlstm2d +timedistributed and Time distributed +GRU+CONV2D GRU gives best accuracy with considerable trainable params(ie; not too much )almost at same epoch as well. Infact LSTM model has very less params as less as 13k which makes it very simple model and takes less training time. But yet the best validation score was seen in GRU model.  
**Reason**

* GRU model has 7 times less params than Conv3d (ie approx 1lakh params)
* Validation accuracy is maximum for simple model 81%
* Training accuracy is 85% which is good.
* Difference between train and validation is 4.4% less than 5% which is tolerable. This confirms there is tolerable overfitting of model or in other words model is not overfitted.
* Size of model approx – 1mb (1259kb)

**BEST MODEL:** Time distributed +GRU+CONV2D