**Gesture Recognition Case Study**

Following two learning algorithms were implemented and validated upon.

1. Conv2D + GRU
2. Conv3D

The intent is to pick up the better learning algorithm and improve it further (for better categorical accuracy)

A function was built to create multiple models based on various hyperparameters . The sole purpose of this was to get a sense of direction in implementation . Once we are able to figure out a base model , we can improve it further for better accuracy.

**Conv3D**

Best Models (best hyperparameters) will be selected from each combination experiment and improved .

A single Conv3d layer contains 1 Conv3d network + 1 batch normalizer + 1 max pooling .

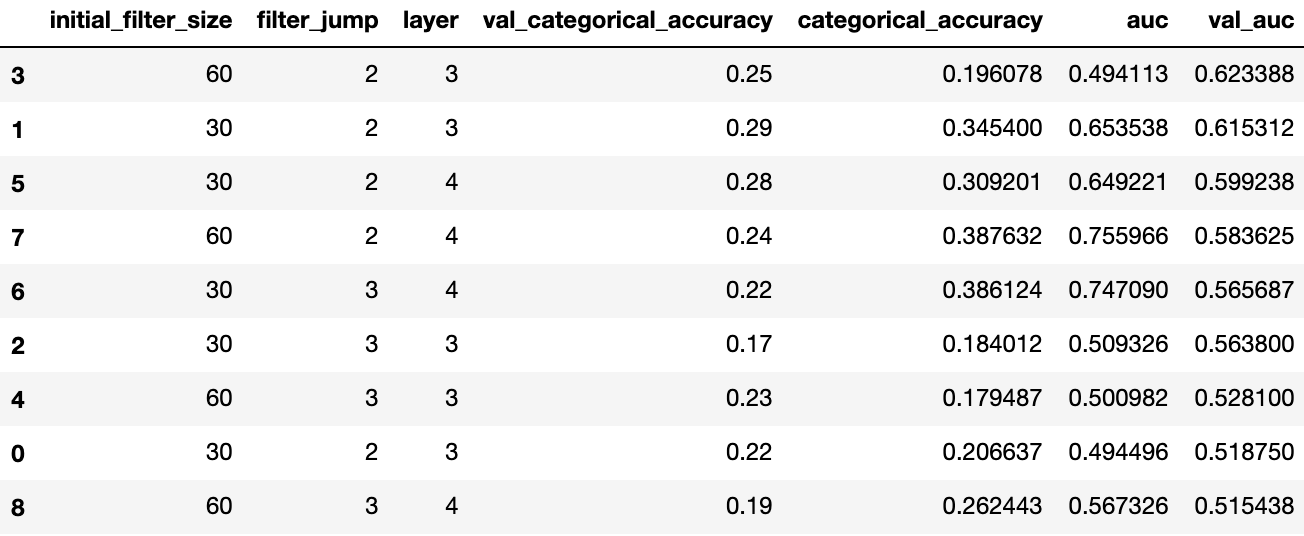
|  |  |
| --- | --- |
| **HyperParameters Experimented Upon** | **Result** |
| 1. 3 and 4 layers of (Conv3d + batch normalization + max pooling ) . 2. Initial Filter of 30 and 60 . Consecutive conv3d layers will have double and triple the filters as compared to previous layer 3. 3 and 4 dense layers (after flatten) with half and 1/3 decrease in output units . 4. Image frames to retain -> [16] 5. Image shape -> (40, 40) and (100, 100) 6. Epochs 3 to 4 epochs of each model from above combination | Model with 3 Con3d layers ( Conv3d + normalization + max pooling) , 30 initial filters , (40, 40 ) image size and 16 video frames outperforms other models based on category accuracy and AUC score .   1. Validation    1. 0.29 accuracy    2. 0.61 auc 2. Training    1. 0.34 accuracy    2. 0.65 auc |
| 1. 3 layers 2. 30 filters initially 3. Doubling of filter size for each consecutive Conv3d layer 4. 2 and 3 dense layers after Flatten with half and ⅓ decrease in output units 5. Image frame to retain 16 and 20 6. Image shapes -> (40, 40) and (60, 60) | After flatten dense layer with 3 dense networks and ⅓ decrease in output units , 20 frames and (40, 40) image shape outperforms other model   1. Validation    1. 0.53 accuracy    2. 0.86 auc 2. Training    1. 0.51 accuracy    2. 0.84 auc |
| Retrying with 4 Conv3d layers instead of 3 . | Both the models perform equally well . We’ll stick with 3 layers since that would mean lesser parameters . |

Further experiments were done with decreasing filter size for every consecutive Conv3d layer and decreasing the number of image frames, but the results were suboptimal. Same suboptimal results were achieved by increasing the kernel size.

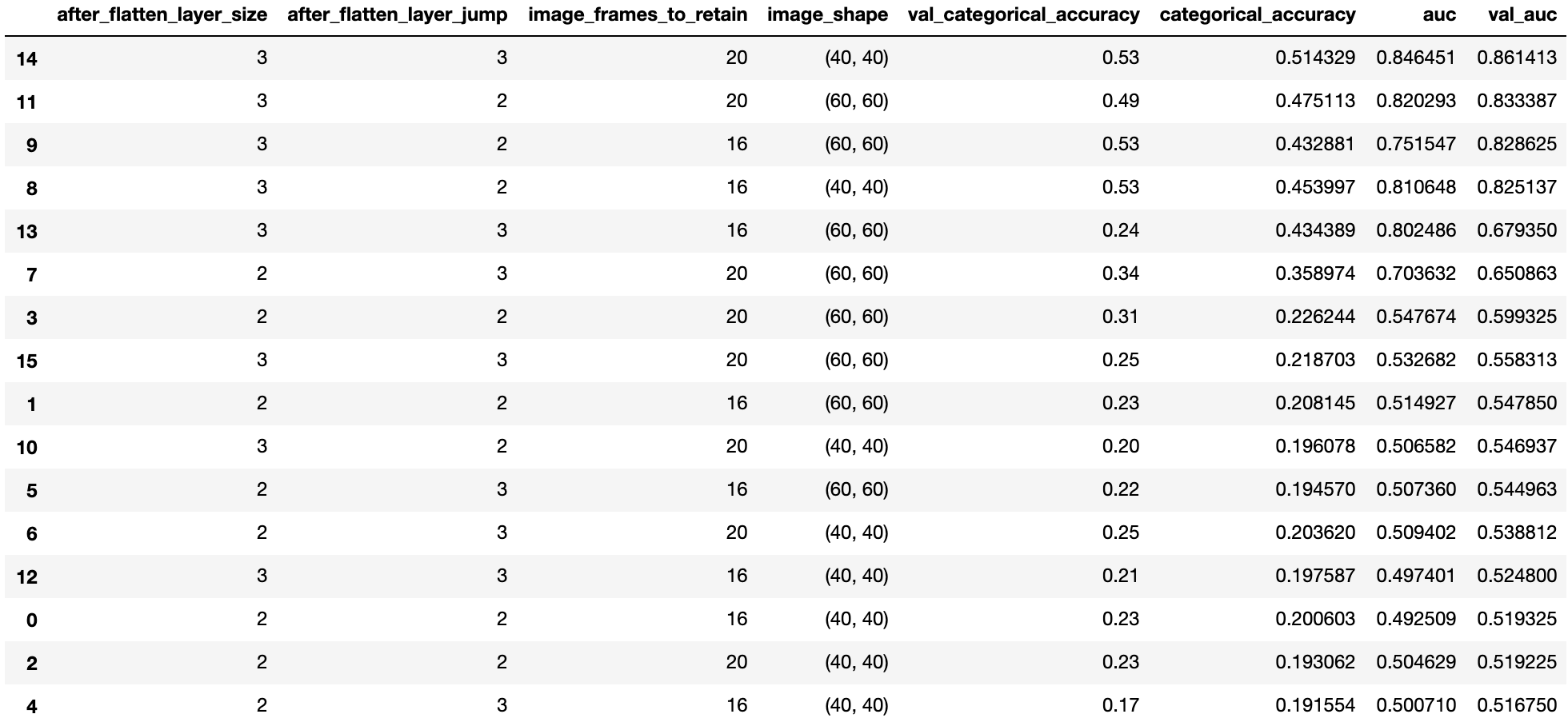
Number of trainable params - 348,485

Below tables showcase accuracy and auc for each of our experimentation.

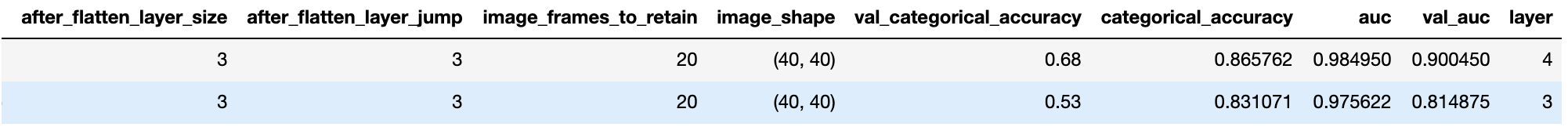
Experiment 1 -> image size fixed at (40,40) and 16 frames. Other factors were iterated upon(as discussed in the table above).



Experiment 2 -> iteration in image frames and shapes , number of layers were kept fixed at 3



Experiment on group layer size (3 and 4 , number of conv2d layer).



We’ll now experiment with the Conv2D + GRU model.

**-------------------------- More on next page---------------------------**

**Conv2D + GRU**

The Conv2D layer here is inspired from VGG16 network . Layers are extracted as per the requirement and plugged in with GRU and dense networks . A single layer group comprises

1. 2 or 3 time distributed Conv2d network
2. 1 time distributed max pooling
3. 1 time distributed batch normalization

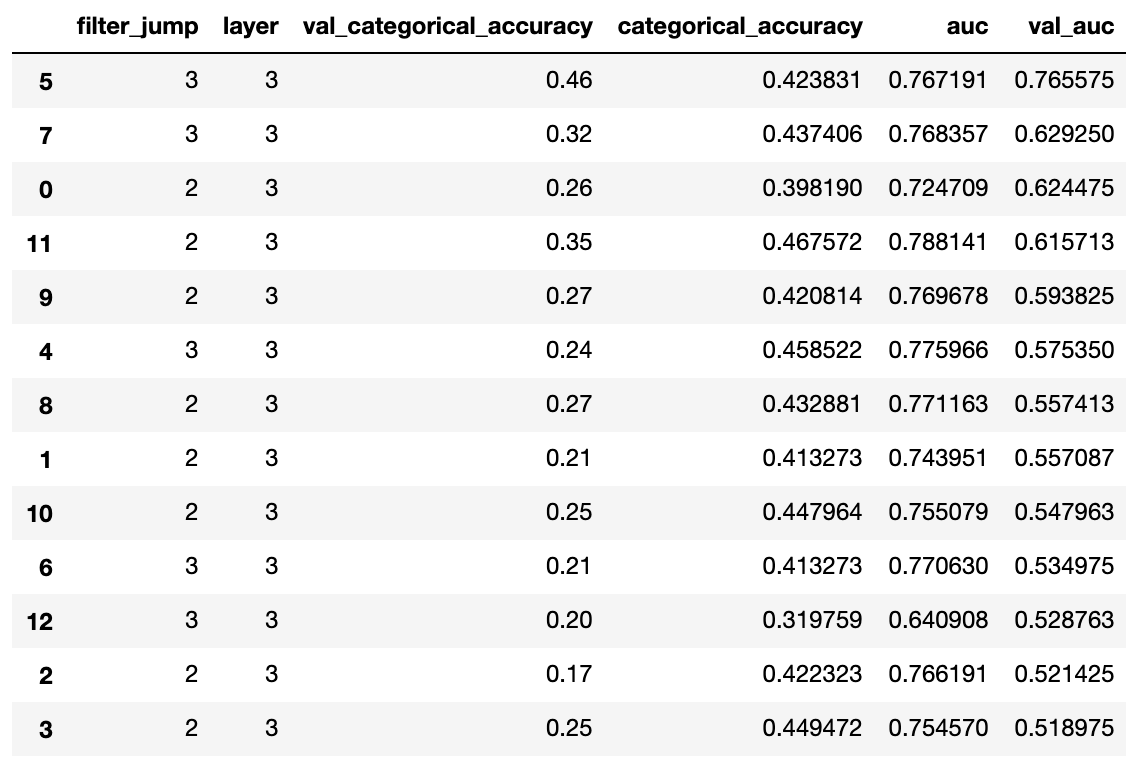
|  |  |
| --- | --- |
| **HyperParameters Experimented Upon** | **Result** |
| 1. Layers -> 3 and 4 2. 3 and 4 dense layers (after flatten) with half and 1/3 decrease in output units . 3. Frames to retain 16 4. Image shape (40, 40) | Model 5 with layer 3 , filter jump 3 outperformed other models .   1. Val    1. Accuracy - 0.46    2. Auc 0.76 2. Train    1. Accuracy - 0.42    2. Aux 0.76 |

The result is more or less the same (compared to Conv3d model). However , the number of trainable params in Conv2d + GRU model is almost 10 times the Conv3d model .

Hence , we will be doing further iterations/ tweaking on the Conv3d model.

Next page contains a summary of the Conv3d model . Further experiments were done on the below explained model (dropouts(0.2), kernel size (3,3) ) , but the results so far have been extremely suboptimal .

**-------------------------- More on next page---------------------------**



**-------------------------- More on next page---------------------------**

**Final Conv3d model**

Layer (type) Output Shape Param #

=================================================================

conv3d\_37 (Conv3D) (None, 20, 40, 40, 30) 750

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_37 (Batc (None, 20, 40, 40, 30) 120

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_37 (MaxPooling (None, 10, 20, 20, 30) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_38 (Conv3D) (None, 10, 20, 20, 60) 14460

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_38 (Batc (None, 10, 20, 20, 60) 240

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_38 (MaxPooling (None, 5, 10, 10, 60) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_39 (Conv3D) (None, 5, 10, 10, 120) 57720

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_39 (Batc (None, 5, 10, 10, 120) 480

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_39 (MaxPooling (None, 2, 5, 5, 120) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv3d\_40 (Conv3D) (None, 2, 5, 5, 240) 230640

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_40 (Batc (None, 2, 5, 5, 240) 960

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling3d\_40 (MaxPooling (None, 1, 2, 2, 240) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_10 (Flatten) (None, 960) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_28 (Dense) (None, 45) 43245

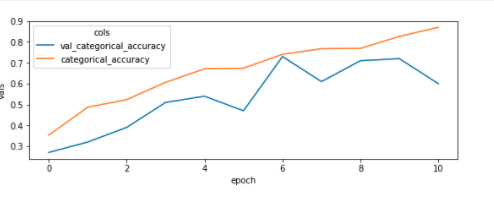
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

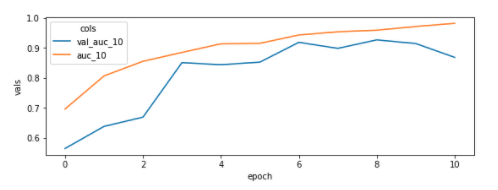
dense\_29 (Dense) (None, 15) 690

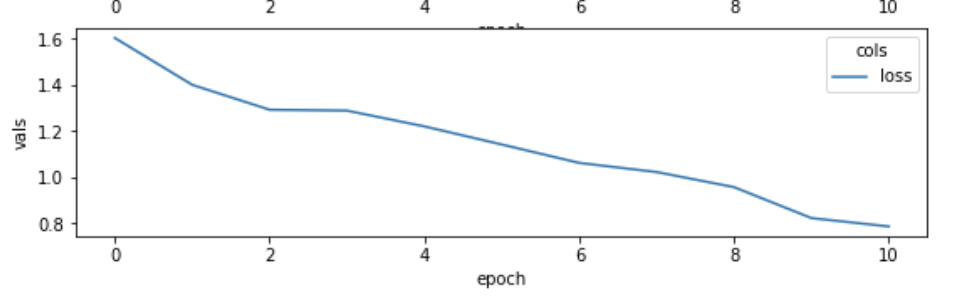
\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_30 (Dense) (None, 5) 80

**-------------------------- More on next page---------------------------**







Somewhere around epoch 6 , both auc and categorical accuracy peaked on validation set before the overfitting started . Following are the results of our final Conv3d model

1. Trainable params - 348,485
2. Validation accuracy - 73%
3. Validation auc - 92%