**Project Title: Deep Learning Course Project - Gesture Recognition**

**Project Description:**

To develop a gesture recognition system that can recognize five different gestures.

**Experiments**

**Experiment :1**

**Model**: **Conv3D**

**Objective**: Passing the folder names for training and validation and passing the batch size for training

**Result/Decision/Explanation**: The path to Project\_data/train.csv and Project\_data/train.csv is passed to train\_doc and val\_doc parameter. The initial batch\_size is taken as 50 to begin with but in corresponding experiments the batch size is increased upto the limit of GPU.

**Experiment: 2**

**Model**: **Conv3D**

**Objective**: Preprocessing image by defining the dimension of the converted image and the frames as well as the RGB channel variable.

**Result/Decision/Explanation**: Since the video contains image frames of different dimensions hence it is imperative to normalize the data before passing it to the generator function. The following dimension variable has been added:

nb\_rows = 120 # X dimension of the image

nb\_cols = 120 # Y dimesnion of the image

The number of frames and channels are taken to be: 30 and 3 respectively

nb\_frames = 30 # length of the video frames

nb\_channel = 3 # number rof channels in images 3 for color(RGB) and 1 for Gray

**Experiment: 3**

**Model**: **Conv3D**

**Objective**: Writing the generator function by modifying the skeleton code given

**Errors:** The generator function was showing an exception related to imread

**Result/Decision/Explanation**: Used openCV implementation of imread by importing the package directly: import cv2

Used cv2.imread() method instead of imread method.

**Experiment: 4**

**Model**: **Conv3D**

**Objective**: Resizing and normalizing the image data within the generator function.

**Result/Decision/Explanation**: To resize the data cv2.resize() is used. This method has an interpolation variable in the argument field. Interpolation of image tries to achieve a best approximation of a pixel's intensity based on the values at surrounding pixels. The advantage of this process is that in resizing the image information gets lost, thus by using image interpolation we can preserve the information in pixel level.

Method used: cv2.resize(image, (nb\_rows,nb\_cols), interpolation = cv2.INTER\_AREA)

Cv2.INTER\_AREA: Resampling using pixel area relation. It may be a preferred method for image decimation, as it gives more free results.

**Experiment: 5**

**Model**: **Conv3D**

**Objective**: Handling data points which are left after full batches

**Result/Decision/Explanation**: The skeleton generator function deals with all the data that can be accommodated within the fixed batches, however there are some data points that might get left out. To deal with it an addition is added to the existing generator function. A conditional statement is written to check for a condition when the datapoints exceed the fixed\_batch\_size, the new batch size is also stored in batch\_size varibale. A method gen\_def\_extra is written which is basically a copy of the generator function but instead of yielding the result it returns the data to the generator function.

**Experiment: 6**

**Model**: **Conv3D**

**Objective**: Creating the Conv3D

**Result/Decision/Explanation**:

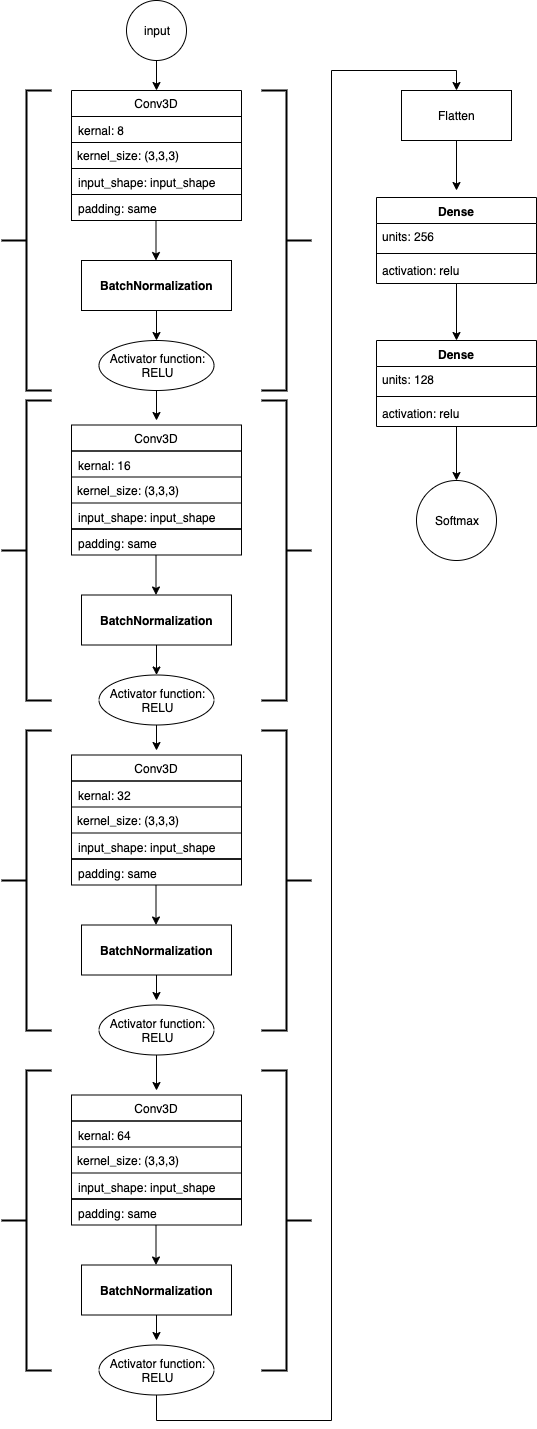
The Conv3D model created has the following structure:

Input neuron size = [8,16,32,64]

Dense layers= [256, 128, 5]

Optimizer: Adam, it was chosen as an optimizer as as Adaptive learning algorithm it shows fast convergence in contrast to other optimizers such as SGD,NAG and momentum

Model diagram:



Model summary:

|  |
| --- |
| Layer (type) Output Shape Param # |
| ================================================================= |
| conv3d\_1 (Conv3D) (None, 30, 120, 120, 8) 656 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| batch\_normalization\_1 (Batch (None, 30, 120, 120, 8) 32 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| activation\_1 (Activation) (None, 30, 120, 120, 8) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| max\_pooling3d\_1 (MaxPooling3 (None, 15, 60, 60, 8) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| conv3d\_2 (Conv3D) (None, 15, 60, 60, 16) 3472 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| batch\_normalization\_2 (Batch (None, 15, 60, 60, 16) 64 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| activation\_2 (Activation) (None, 15, 60, 60, 16) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| max\_pooling3d\_2 (MaxPooling3 (None, 7, 30, 30, 16) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| conv3d\_3 (Conv3D) (None, 7, 30, 30, 32) 4640 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| batch\_normalization\_3 (Batch (None, 7, 30, 30, 32) 128 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| activation\_3 (Activation) (None, 7, 30, 30, 32) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| max\_pooling3d\_3 (MaxPooling3 (None, 3, 15, 15, 32) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| conv3d\_4 (Conv3D) (None, 3, 15, 15, 64) 18496 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| activation\_4 (Activation) (None, 3, 15, 15, 64) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| dropout\_1 (Dropout) (None, 3, 15, 15, 64) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| max\_pooling3d\_4 (MaxPooling3 (None, 1, 7, 7, 64) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| flatten\_1 (Flatten) (None, 3136) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| dense\_1 (Dense) (None, 256) 803072 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| dropout\_2 (Dropout) (None, 256) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| dense\_2 (Dense) (None, 128) 32896 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| dropout\_3 (Dropout) (None, 128) 0 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |
| dense\_3 (Dense) (None, 5) 645 |
| ================================================================= |

|  |
| --- |
| ================================================================= |
| Total params: 864,101 |
| Trainable params: 863,989 |
| Non-trainable params: 112 |
| \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ |

**Experiment: 7**

**Model**: **Conv3D**

**Objective**: Defining the file-path where the model is stored.

**Error:** The file-path mentioned in the skeleton code does-not store the model in .h5 format.

**Result/Decision/Explanation**: The filepath code was changed so that the models can be stored in .h5 format.

**Experiment: 8**

**Model**: **Conv3D**

**Objective:** Compiling model with batch\_size = 50, num\_epochs = 20

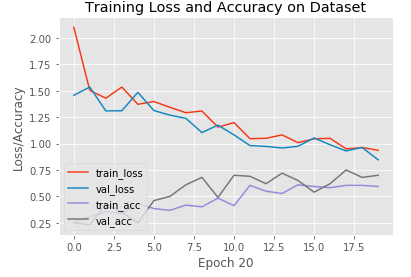
**Metrices**:

loss: 0.9363

categorical\_accuracy: 0.5934

val\_loss: 0.8462

val\_categorical\_accuracy: 0.7000



**Remarks**: Will increase the no of epocs to check if there is any increase in the accuracy

**Experiment: 9**

**Model**: **Conv3D**

**Objective:** Compiling model with batch\_size = 50, num\_epochs = 30

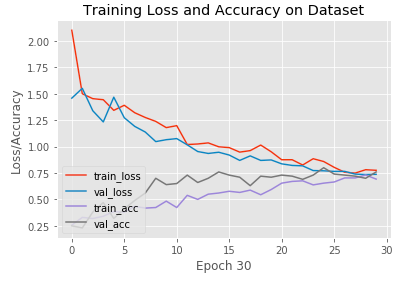
**Metrices**:

loss: 0.7757

categorical\_accuracy: 0.6923

val\_loss: 0.7375

val\_categorical\_accuracy: 0.7600



**Remarks**: Just by increasing the epocs to 30 from 20 has resulted in increase of categorical\_accuracy to 69.23% from 59.34% and of validation\_ categorical\_accuracy to 76% from 70%. In next experiment will drastically increase the epoch size keeping constant the batch size.

**Experiment: 10**

**Objective:** Compiling model with batch\_size = 50, num\_epochs = 90

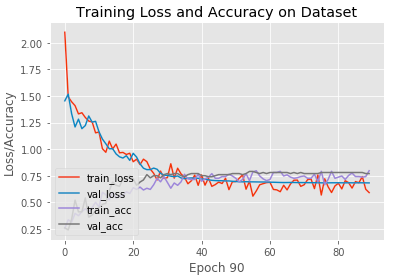
**Metrices**:

loss: 0.5908

categorical\_accuracy: 0.7967

val\_loss: 0.6821

val\_categorical\_accuracy: 0.7700



**Remarks**: Just by increasing the epocs to 90 from 30 has resulted in increase of categorical\_accuracy to 79.67% from 69.23% and of validation\_ categorical\_accuracy to 77% from 76%. In next experiment will drastically increase the batch size keeping constant the epoch size and see if there is any improvement in accuracy.

**Experiment: 11**

**Objective:** Compiling model with batch\_size = 90, num\_epochs = 90

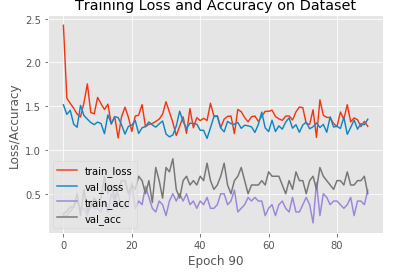
**Metrices**:

loss: 1.2703

categorical\_accuracy: 0.5417

val\_loss: 1.3518

val\_categorical\_accuracy: 0.5000



**Remarks**: It seems that increasing the batch size decreases the accuracy of the model. Categorical\_accuracy has come down to 54.17% from 79.67% and validation\_ categorical\_accuracy to 50% from 77%. In the next experiment the batch size will be drastically reduced keeping constant the no of epoch.

**Experiment: 12**

**Objective:** Compiling model with batch\_size = 10, num\_epochs = 90

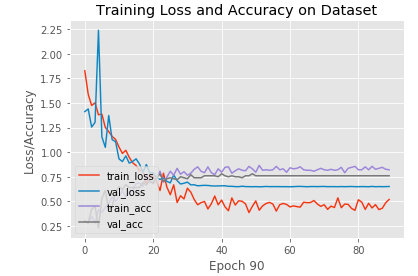
**Metrices**:

loss: 0.5195

categorical\_accuracy: 0.8209

val\_loss: 0.6511

val\_categorical\_accuracy: 0.7600



**Remarks**: It seems that the accuracy improves significantly on lower batch size. For a batch size = 10 and epoch = 90 the categorical\_accuracy and val\_categorical\_accuracy is highest in all the experiments and are: 82.09% and 76% respectively.