## Gesture Recognition

In this group project, you are going to build a 3D Conv model that will be able to predict the 5 gestures correctly. Please import the following libraries to get started.

```
import numpy as np
import os
import datetime
import cv2
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings('ignore')
```

We set the random seed so that the results don't vary drastically.

```
np.random.seed(30)
import random as rn
rn.seed(30)
from keras import backend as K
import tensorflow as tf
tf.random.set_seed(30)
```

In this block, you read the folder names for training and validation. You also set the `batch_size` here. Note that you set the batch size in such a way that you are able to use the GPU in full capacity. You keep increasing the batch size until the machine throws an error.

```
train_doc = np.random.permutation(open('/notebooks/storage/Final_data/Collated_training/train.csv').readlines())
val_doc = np.random.permutation(open('/notebooks/storage/Final_data/Collated_training/val.csv').readlines())
batch_size = 15 #experiment with the batch size
```

### Checking Data Imbalance

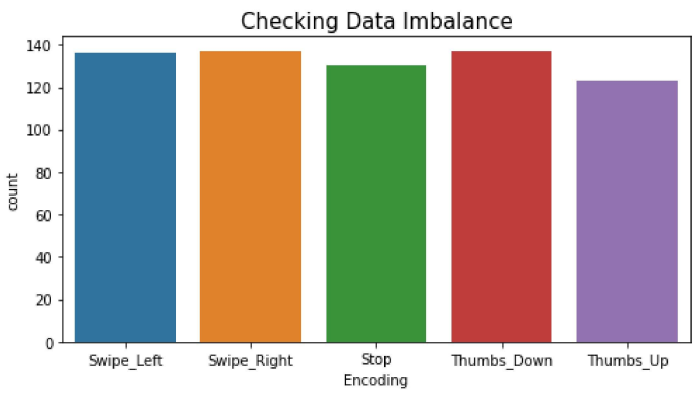
```
import pandas as pd
df = pd.read_csv('/content/train.csv',header=None)
df.columns = ['Action']
df['Encoding'] = df['Action'].apply(lambda x: x[-1])
data_count = df['Encoding'].value_counts()
data_count
```

```
1    137
3    137
0    136
2    130
4    123
Name: Encoding, dtype: int64
```

```
{x.split(';')[1]:x.split(';')[2][0] for x in train_doc}
```

```
{'Left Swipe_new_Left Swipe_new': '0',
 'Left_Swipe_new': '0',
 'Right Swipe_new': '1',
 'Right_Swipe_new': '1',
 'Stop Gesture_new': '2',
 'Stop_new': '2',
 'Thumbs Down_new': '3',
 'Thumbs Up_new': '4',
 'Thumbs_Down_new': '3',
 'Thumbs_Up_new': '4'}
```

```
import seaborn as sns
plt.figure(figsize=(8,4))
sns.countplot(df['Encoding'])
plt.title('Checking Data Imbalance',size=15)
plt.xticks(ticks=[0,1,2,3,4],labels=['Swipe_Left','Swipe_Right','Stop','Thumbs_Down','Thumbs_Up'])
plt.show()
```




## Generator

This is one of the most important part of the code. The overall structure of the generator has been given. In the generator, you are going to preprocess the images as you have images of 2 different dimensions as well as create a batch of video frames. You have to experiment with `img_idx, y,z` and normalization such that you get high accuracy.

### Using OpenCv instead of imread for the generator function due to following reasons.

- OpenCv is better in performance/speed of image processing as compared to skimage.
- OpenCv is more convenient to use than skimage.

```
def get_batch(source_path, folder_list, batch_size,batch,t,img_idx):
  batch_data = np.zeros((batch_size,len(img_idx),120,120,3)) # x is the number of images you use for each video, (y,z) is the final size of the input images and 3 is the number of channels RGB
  batch_labels = np.zeros((batch_size,5)) # batch_labels is the one hot representation of the output
  for folder in range(batch_size): # iterate over the batch_size
      imgs = os.listdir(source_path+'/'+ t[folder + (batch*batch_size)].split(';')[0]) # read all the images in the folder
      for idx,item in enumerate(img_idx): #  Iterate iver the frames/images of a folder to read them in
          image = cv2.imread(source_path+'/'+ t[folder + (batch*batch_size)].strip().split(';')[0]+'/'+imgs[item]).astype(np.float32)
          h,w = image.shape[0:2]
          #crop the images and resize them. Note that the images are of 2 different shape
          #and the conv3D will throw error if the inputs in a batch have different shapes
          if (h>120) and (w>160):
            image = cv2.resize(image,(160,120),interpolation = cv2.INTER_NEAREST)

          #Crop the Image
          image = image[:,20:140,:]

          batch_data[folder,idx,:,:,0] = cv2.normalize(image[:,:,0],None,0.00,1.00,cv2.NORM_MINMAX)#normalise and feed in the image
          batch_data[folder,idx,:,:,1] = cv2.normalize(image[:,:,1],None,0.00,1.00,cv2.NORM_MINMAX)#normalise and feed in the image
          batch_data[folder,idx,:,:,2] = cv2.normalize(image[:,:,2],None,0.00,1.00,cv2.NORM_MINMAX)#normalise and feed in the image

      batch_labels[folder, int(t[folder + (batch*batch_size)].strip().split(';')[2])] = 1
  return batch_data, batch_labels
```

```
def generator(source_path, folder_list, batch_size):
    print( 'Source path = ', source_path, '; batch size =', batch_size)
    img_idx = range(0,30)
    while True:
        t = np.random.permutation(folder_list)
        num_batches = len(folder_list) // batch_size # calculate the number of batches
        for batch in range(num_batches):  # we iterate over the number of batches
          yield get_batch(source_path, folder_list, batch_size,batch,t,img_idx)

        # write the code for the remaining data points which are left after full batches
        remaining = len(folder_list) % batch_size
        if remaining != 0:
          batch_size = remaining
          yield get_batch(source_path, folder_list, batch_size,batch,t,img_idx)
```

```
# def generator(source_path, folder_list, batch_size):
#     print( 'Source path = ', source_path, '; batch size =', batch_size)
#     img_idx = #create a list of image numbers you want to use for a particular video
#     while True:
#         t = np.random.permutation(folder_list)
#         num_batches = # calculate the number of batches
#         for batch in range(num_batches): # we iterate over the number of batches
#             batch_data = np.zeros((batch_size,x,y,z,3)) # x is the number of images you use for each video, (y,z) is the final size of the input images and 3 is the number of channels RGB
#             batch_labels = np.zeros((batch_size,5)) # batch_labels is the one hot representation of the output
#             for folder in range(batch_size): # iterate over the batch_size
#                 imgs = os.listdir(source_path+'/'+ t[folder + (batch*batch_size)].split(';')[0]) # read all the images in the folder
#                 for idx,item in enumerate(img_idx): #  Iterate iver the frames/images of a folder to read them in
#                     image = imread(source_path+'/'+ t[folder + (batch*batch_size)].strip().split(';')[0]+'/'+imgs[item]).astype(np.float32)

#                     #crop the images and resize them. Note that the images are of 2 different shape
#                     #and the conv3D will throw error if the inputs in a batch have different shapes

#                     batch_data[folder,idx,:,:,0] = #normalise and feed in the image
#                     batch_data[folder,idx,:,:,1] = #normalise and feed in the image
#                     batch_data[folder,idx,:,:,2] = #normalise and feed in the image

#                 batch_labels[folder, int(t[folder + (batch*batch_size)].strip().split(';')[2])] = 1
#             yield batch_data, batch_labels #you yield the batch_data and the batch_labels, remember what does yield do


#         # write the code for the remaining data points which are left after full batches
```

Note here that a video is represented above in the generator as (number of images, height, width, number of channels). Take this into consideration while creating the model architecture.

```
curr_dt_time = datetime.datetime.now()
# train_path = '/notebooks/storage/Final_data/Collated_training/train'
# val_path = '/notebooks/storage/Final_data/Collated_training/val'

# train_path = '/content/Project_data/train'
# val_path = '/content/Project_data/val'

train_path = '/content/train'
val_path = '/content/val'
num_train_sequences = len(train_doc)
print('# training sequences =', num_train_sequences)
num_val_sequences = len(val_doc)
print('# validation sequences =', num_val_sequences)
num_epochs = 20 # choose the number of epochs
print ('# epochs =', num_epochs)
```

```
# training sequences = 663
# validation sequences = 100
# epochs = 20
```

## Function for the Batch Size

```
def gen_train_val (batch_size):
  train_generator = generator(train_path, train_doc, batch_size)
  val_generator = generator(val_path, val_doc, batch_size)

  if (num_train_sequences%batch_size) == 0:
    steps_per_epoch = int(num_train_sequences/batch_size)
  else:
    steps_per_epoch = (num_train_sequences//batch_size) + 1

  if (num_val_sequences%batch_size) == 0:
    validation_steps = int(num_val_sequences/batch_size)
  else:
    validation_steps = (num_val_sequences//batch_size) + 1

  return train_generator,val_generator,steps_per_epoch,validation_steps
```

## Function for Checkpoint

```
def set_checkpoint(model_name,fact=0.1,pati=8):

  logdir = os.path.join("logs",model_name)
  tensorboard_callback = tf.keras.callbacks.TensorBoard(logdir, histogram_freq=1,write_images=True)

  filepath = os.path.join("model",model_name+'.h5',)
  checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=True, mode='auto')

  LR = ReduceLROnPlateau(monitor='val_loss', factor=fact,patience=pati,mode='min')

  return [tensorboard_callback, checkpoint, LR]
```

## Function for train and validation generator

```
def gen_train_val (batch_size):
  train_generator = generator(train_path, train_doc, batch_size)
  val_generator = generator(val_path, val_doc, batch_size)

  if (num_train_sequences%batch_size) == 0:
    steps_per_epoch = int(num_train_sequences/batch_size)
  else:
    steps_per_epoch = (num_train_sequences//batch_size) + 1

  if (num_val_sequences%batch_size) == 0:
    validation_steps = int(num_val_sequences/batch_size)
  else:
    validation_steps = (num_val_sequences//batch_size) + 1

  return train_generator,val_generator,steps_per_epoch,validation_steps
```

# Model

Here you make the model using different functionalities that Keras provides. Remember to use `Conv3D` and `MaxPooling3D` and not `Conv2D` and `Maxpooling2D` for a 3D convolution model. You would want to use `TimeDistributed` while building a Conv2D + RNN model. Also remember that the last layer is the softmax. Design the network in such a way that the model is able to give good accuracy on the least number of parameters so that it can fit in the memory of the webcam.

```
from keras.models import Sequential, Model
from keras.layers import Dense, GRU, Flatten, TimeDistributed, Flatten, BatchNormalization, Activation
from keras.layers.convolutional import Conv3D, MaxPooling3D
from keras.layers import Conv2D,MaxPooling2D,Dropout
from keras.callbacks import ModelCheckpoint, ReduceLROnPlateau,TensorBoard
from keras import optimizers
%load_ext tensorboard
```

## Convolution 3D : Initial Model

- Creating a base model where we just want to check if data fits the model.
- According to our expectation the model should overfit.

```
#write your model here
model = Sequential([
                    Conv3D(64, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(30,120,120,3)),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Conv3D(128, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Flatten(),
                    Dense(5, activation='softmax')
])
```

Now that you have written the model, the next step is to `compile` the model. When you print the `summary` of the model, you'll see the total number of parameters you have to train.

```
#optimiser =  tf.keras.optimizers.Adam(learning_rate=0.001) #write your optimizer
optimiser =  tf.keras.optimizers.SGD(learning_rate=0.001)
model.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print (model.summary())
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d (Conv3D)             (None, 28, 118, 118, 64)  5248

 max_pooling3d (MaxPooling3D  (None, 14, 59, 59, 64)   0
 )

 conv3d_1 (Conv3D)           (None, 12, 57, 57, 128)   221312

 max_pooling3d_1 (MaxPooling  (None, 6, 28, 28, 128)   0
 3D)

 flatten (Flatten)           (None, 602112)            0

 dense (Dense)               (None, 5)                 3010565

=================================================================
Total params: 3,237,125
Trainable params: 3,237,125
Non-trainable params: 0
_________________________________________________________________
None
```

Let us create the `train_generator` and the `val_generator` which will be used in `.fit_generator`.

```
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=15)
```

The `steps_per_epoch` and `validation_steps` are used by `fit_generator` to decide the number of next() calls it need to make.

Let us now fit the model. This will start training the model and with the help of the checkpoints, you'll be able to save the model at the end of each epoch.

```
# Check Point for the base
callbacks_list = set_checkpoint('model')
```

```
model.fit(train_generator, steps_per_epoch=steps_per_epoch, epochs=num_epochs,
                    callbacks=callbacks_list, validation_data=val_generator,
                    validation_steps=validation_steps)
```

```
Epoch 00005: val_loss did not improve from 1.19525
45/45 [==============================] - 12s 275ms/step - loss: 1.0876 - categorical_accuracy: 0.5556 - val_loss: 1.3051 - val_categorical_accuracy: 0.4571 - lr: 0.0010
Epoch 6/20
45/45 [==============================] - ETA: 0s - loss: 0.9613 - categorical_accuracy: 0.6444
Epoch 00006: val_loss improved from 1.19525 to 1.03635, saving model to model/model.h5
45/45 [==============================] - 13s 296ms/step - loss: 0.9613 - categorical_accuracy: 0.6444 - val_loss: 1.0363 - val_categorical_accuracy: 0.6000 - lr: 0.0010
Epoch 7/20
45/45 [==============================] - ETA: 0s - loss: 0.4914 - categorical_accuracy: 0.8889
Epoch 00007: val_loss did not improve from 1.03635
45/45 [==============================] - 13s 286ms/step - loss: 0.4914 - categorical_accuracy: 0.8889 - val_loss: 1.2330 - val_categorical_accuracy: 0.5429 - lr: 0.0010
Epoch 8/20
45/45 [==============================] - ETA: 0s - loss: 0.4227 - categorical_accuracy: 0.8963
Epoch 00008: val_loss did not improve from 1.03635
45/45 [==============================] - 13s 303ms/step - loss: 0.4227 - categorical_accuracy: 0.8963 - val_loss: 1.0997 - val_categorical_accuracy: 0.6286 - lr: 0.0010
Epoch 9/20
45/45 [==============================] - ETA: 0s - loss: 0.5393 - categorical_accuracy: 0.8519
Epoch 00009: val_loss improved from 1.03635 to 1.01048, saving model to model/model.h5
45/45 [==============================] - 13s 295ms/step - loss: 0.5393 - categorical_accuracy: 0.8519 - val_loss: 1.0105 - val_categorical_accuracy: 0.6143 - lr: 0.0010
Epoch 10/20
45/45 [==============================] - ETA: 0s - loss: 0.4675 - categorical_accuracy: 0.8593
Epoch 00010: val_loss did not improve from 1.01048
45/45 [==============================] - 13s 283ms/step - loss: 0.4675 - categorical_accuracy: 0.8593 - val_loss: 1.3852 - val_categorical_accuracy: 0.3571 - lr: 0.0010
Epoch 11/20
45/45 [==============================] - ETA: 0s - loss: 0.4012 - categorical_accuracy: 0.8667
Epoch 00011: val_loss improved from 1.01048 to 0.94111, saving model to model/model.h5
45/45 [==============================] - 12s 268ms/step - loss: 0.4012 - categorical_accuracy: 0.8667 - val_loss: 0.9411 - val_categorical_accuracy: 0.6143 - lr: 0.0010
Epoch 12/20
45/45 [==============================] - ETA: 0s - loss: 0.2215 - categorical_accuracy: 0.9630
Epoch 00012: val_loss did not improve from 0.94111
45/45 [==============================] - 13s 296ms/step - loss: 0.2215 - categorical_accuracy: 0.9630 - val_loss: 1.1275 - val_categorical_accuracy: 0.6000 - lr: 0.0010
Epoch 13/20
45/45 [==============================] - ETA: 0s - loss: 0.2367 - categorical_accuracy: 0.9407
Epoch 00013: val_loss did not improve from 0.94111
45/45 [==============================] - 12s 280ms/step - loss: 0.2367 - categorical_accuracy: 0.9407 - val_loss: 1.1249 - val_categorical_accuracy: 0.6714 - lr: 0.0010
Epoch 14/20
45/45 [==============================] - ETA: 0s - loss: 0.1941 - categorical_accuracy: 0.9630
Epoch 00014: val_loss did not improve from 0.94111
45/45 [==============================] - 13s 302ms/step - loss: 0.1941 - categorical_accuracy: 0.9630 - val_loss: 1.1787 - val_categorical_accuracy: 0.5571 - lr: 0.0010
Epoch 15/20
45/45 [==============================] - ETA: 0s - loss: 0.2096 - categorical_accuracy: 0.9556
Epoch 00015: val_loss did not improve from 0.94111
45/45 [==============================] - 12s 276ms/step - loss: 0.2096 - categorical_accuracy: 0.9556 - val_loss: 1.0027 - val_categorical_accuracy: 0.5857 - lr: 0.0010
Epoch 16/20
45/45 [==============================] - ETA: 0s - loss: 0.1641 - categorical_accuracy: 0.9926
Epoch 00016: val_loss did not improve from 0.94111
45/45 [==============================] - 12s 272ms/step - loss: 0.1641 - categorical_accuracy: 0.9926 - val_loss: 0.9444 - val_categorical_accuracy: 0.6571 - lr: 0.0010
Epoch 17/20
45/45 [==============================] - ETA: 0s - loss: 0.0901 - categorical_accuracy: 0.9926
Epoch 00017: val_loss improved from 0.94111 to 0.94098, saving model to model/model.h5
45/45 [==============================] - 13s 286ms/step - loss: 0.0901 - categorical_accuracy: 0.9926 - val_loss: 0.9410 - val_categorical_accuracy: 0.7000 - lr: 0.0010
Epoch 18/20
45/45 [==============================] - ETA: 0s - loss: 0.0877 - categorical_accuracy: 0.9852
Epoch 00018: val_loss did not improve from 0.94098
45/45 [==============================] - 13s 287ms/step - loss: 0.0877 - categorical_accuracy: 0.9852 - val_loss: 1.0434 - val_categorical_accuracy: 0.6286 - lr: 0.0010
Epoch 19/20
45/45 [==============================] - ETA: 0s - loss: 0.0926 - categorical_accuracy: 1.0000
Epoch 00019: val_loss did not improve from 0.94098
45/45 [==============================] - 13s 293ms/step - loss: 0.0926 - categorical_accuracy: 1.0000 - val_loss: 1.1048 - val_categorical_accuracy: 0.5857 - lr: 0.0010
Epoch 20/20
```

```
# Visualizing the Model Results using TenserBoard

%tensorboard --logdir logs
```


Best Epoch (Epoch 17) Result from the above odel **Train Accuracy : 99.26 Validation Accuracy : 70.00**

- We can clearly see that the above Model is overfitting, it was the purpose of the initial model to verify the fit

## Convolution 3D Models

- Starting with the CONV 3D models
- We we see the accuracy by playing around with the Batch size, Optimizer, model architecture
- We will also expirment with the no. of layers and Drop outs in the Model Architecture

## Conv 3D Model1 : (Batch Size: 15 , No. of Frames = 30 , Optimizer=SGD)

```
#Changing the batch Size
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=15)

#Setting the Callback Parameters
callbacks_list1 = set_checkpoint('model1',pati=6)
```

```
model1 = Sequential([
                    Conv3D(16, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(30,120,120,3)),
                    BatchNormalization(),
                    Conv3D(32, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    Conv3D(64, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Conv3D(128, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Flatten(),
                    Dense(64, activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    Dense(5, activation='softmax')
])
```

```
optimiser = tf.keras.optimizers.SGD(learning_rate=0.001) # write your optimizer
model1.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print (model1.summary())
```

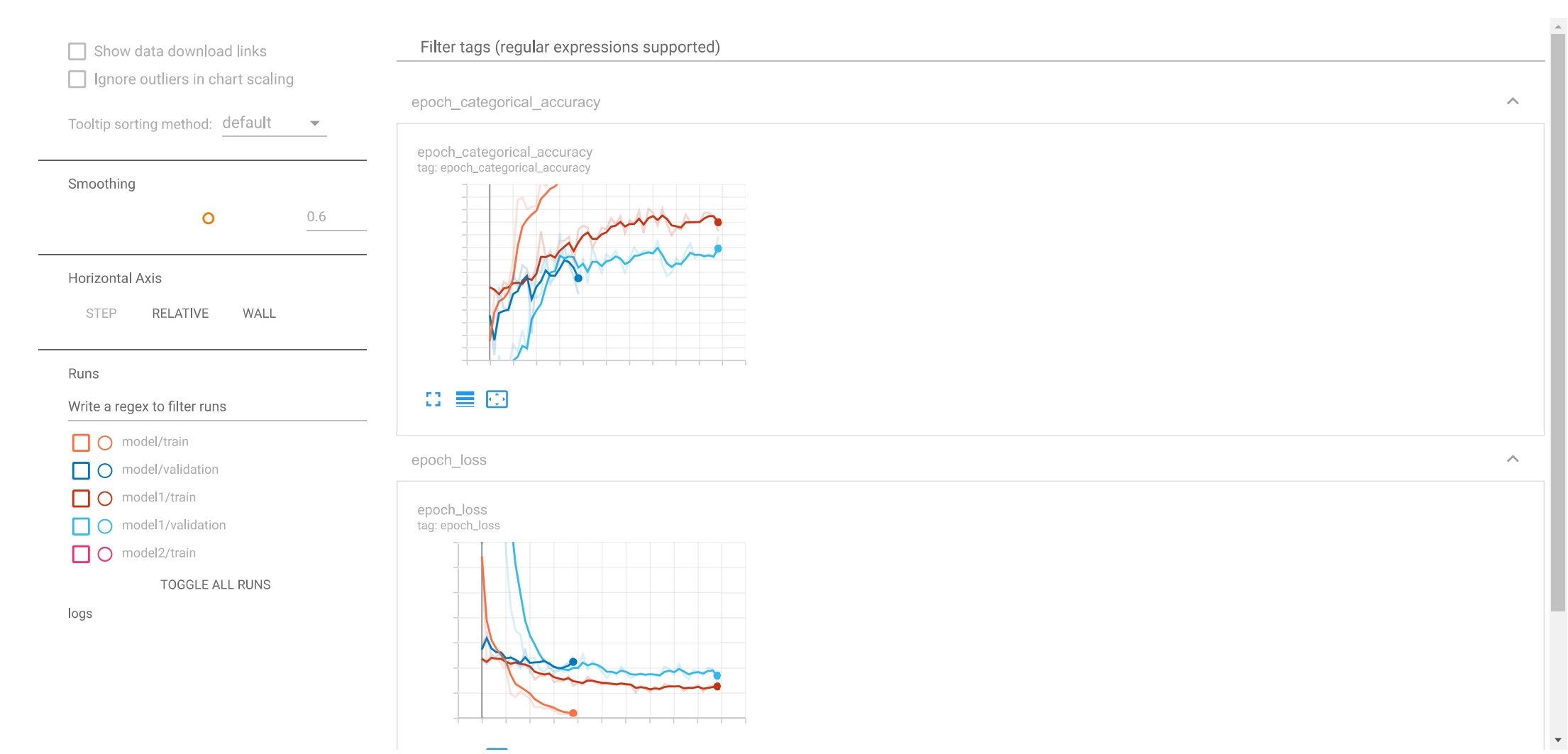
```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d_2 (Conv3D)           (None, 28, 118, 118, 16)  1312

 batch_normalization (BatchN  (None, 28, 118, 118, 16)  64
 ormalization)

 conv3d_3 (Conv3D)           (None, 26, 116, 116, 32)  13856

 batch_normalization_1 (Batc  (None, 26, 116, 116, 32)  128
 hNormalization)

 conv3d_4 (Conv3D)           (None, 24, 114, 114, 64)  55360

 batch_normalization_2 (Batc  (None, 24, 114, 114, 64)  256
 hNormalization)

 max_pooling3d_2 (MaxPooling  (None, 12, 57, 57, 64)   0
 3D)

 conv3d_5 (Conv3D)           (None, 10, 55, 55, 128)   221312

 batch_normalization_3 (Batc  (None, 10, 55, 55, 128)  512
 hNormalization)

 max_pooling3d_3 (MaxPooling  (None, 5, 27, 27, 128)   0
 3D)

 flatten_1 (Flatten)         (None, 466560)            0

 dense_1 (Dense)             (None, 64)                29859904

 batch_normalization_4 (Batc  (None, 64)               256
 hNormalization)

 dense_2 (Dense)             (None, 5)                 325

=================================================================
Total params: 30,153,285
Trainable params: 30,152,677
Non-trainable params: 608
_________________________________________________________________
None
```

```
model1.fit(train_generator,
          steps_per_epoch=steps_per_epoch,
          epochs=50,
          verbose=1,
          validation_data=val_generator,
          validation_steps=validation_steps,
          callbacks=callbacks_list1)
```

```
Epoch 00035: val_loss did not improve from 0.80483
45/45 [==============================] - 22s 487ms/step - loss: 0.5936 - categorical_accuracy: 0.8519 - val_loss: 0.8523 - val_categorical_accuracy: 0.6857 - lr: 1.0000e-04
Epoch 36/50
45/45 [==============================] - ETA: 0s - loss: 0.5338 - categorical_accuracy: 0.8370
Epoch 00036: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 467ms/step - loss: 0.5338 - categorical_accuracy: 0.8370 - val_loss: 0.8935 - val_categorical_accuracy: 0.6714 - lr: 1.0000e-04
Epoch 37/50
45/45 [==============================] - ETA: 0s - loss: 0.6265 - categorical_accuracy: 0.7852
Epoch 00037: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 474ms/step - loss: 0.6265 - categorical_accuracy: 0.7852 - val_loss: 0.8610 - val_categorical_accuracy: 0.7286 - lr: 1.0000e-04
Epoch 38/50
45/45 [==============================] - ETA: 0s - loss: 0.5749 - categorical_accuracy: 0.8519
Epoch 00038: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 471ms/step - loss: 0.5749 - categorical_accuracy: 0.8519 - val_loss: 0.8346 - val_categorical_accuracy: 0.6286 - lr: 1.0000e-04
Epoch 39/50
45/45 [==============================] - ETA: 0s - loss: 0.6854 - categorical_accuracy: 0.7926
Epoch 00039: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 472ms/step - loss: 0.6854 - categorical_accuracy: 0.7926 - val_loss: 1.0307 - val_categorical_accuracy: 0.5857 - lr: 1.0000e-05
Epoch 40/50
45/45 [==============================] - ETA: 0s - loss: 0.6554 - categorical_accuracy: 0.7481
Epoch 00040: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 472ms/step - loss: 0.6554 - categorical_accuracy: 0.7481 - val_loss: 0.9844 - val_categorical_accuracy: 0.6000 - lr: 1.0000e-05
Epoch 41/50
45/45 [==============================] - ETA: 0s - loss: 0.6330 - categorical_accuracy: 0.7778
Epoch 00041: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 473ms/step - loss: 0.6330 - categorical_accuracy: 0.7778 - val_loss: 0.8816 - val_categorical_accuracy: 0.6571 - lr: 1.0000e-05
Epoch 42/50
45/45 [==============================] - ETA: 0s - loss: 0.6525 - categorical_accuracy: 0.7704
Epoch 00042: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 478ms/step - loss: 0.6525 - categorical_accuracy: 0.7704 - val_loss: 1.0740 - val_categorical_accuracy: 0.6286 - lr: 1.0000e-05
Epoch 43/50
45/45 [==============================] - ETA: 0s - loss: 0.5451 - categorical_accuracy: 0.8296
Epoch 00043: val_loss did not improve from 0.80483
45/45 [==============================] - 21s 476ms/step - loss: 0.5451 - categorical_accuracy: 0.8296 - val_loss: 0.8655 - val_categorical_accuracy: 0.6857 - lr: 1.0000e-05
```

```
# Visualizing the Model Results using TenserBoard
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 646), started 0:18:12 ago. (Use '!kill 646' to kill it.)
```




Best Epoch(Epoch 50) Result from the above Model1 **Train Accuracy : 76.30 Validation Accuracy : 74.29**

- We can clearly see that Model1 is performing better in terms of generalizing the model.
- But clearly the Accuracy is less on both the Train and Validation data.

## CONV3D : Model 2

### Conv 3D Model1 : (Batch Size: 10 , No. of Frames = 30 , Optimizer=Adam , Dense Layer= 64)

```
#Changing the batch Size
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=20)

#Setting the Callback Parameters
callbacks_list2 = set_checkpoint('model2',pati=6)
```

```
# Model Architecture
model2 = Sequential([
                    Conv3D(16, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(30,120,120,3)),
                    BatchNormalization(),
                    Conv3D(32, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    Conv3D(64, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Conv3D(128, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Flatten(),
                    Dense(64, activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    Dense(5, activation='softmax')
])
```

```
#Compiling the Model
optimiser =  tf.keras.optimizers.Adam(learning_rate=0.001) #write your optimizer
model2.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print (model2.summary())
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d_10 (Conv3D)          (None, 28, 118, 118, 16)  1312

 batch_normalization_10 (Bat  (None, 28, 118, 118, 16)  64
 chNormalization)

 conv3d_11 (Conv3D)          (None, 26, 116, 116, 32)  13856

 batch_normalization_11 (Bat  (None, 26, 116, 116, 32)  128
 chNormalization)

 conv3d_12 (Conv3D)          (None, 24, 114, 114, 64)  55360

 batch_normalization_12 (Bat  (None, 24, 114, 114, 64)  256
 chNormalization)

 max_pooling3d_6 (MaxPooling  (None, 12, 57, 57, 64)   0
 3D)

 conv3d_13 (Conv3D)          (None, 10, 55, 55, 128)   221312

 batch_normalization_13 (Bat  (None, 10, 55, 55, 128)  512
 chNormalization)

 max_pooling3d_7 (MaxPooling  (None, 5, 27, 27, 128)   0
 3D)

 flatten_3 (Flatten)         (None, 466560)            0

 dense_5 (Dense)             (None, 64)                29859904

 batch_normalization_14 (Bat  (None, 64)               256
 chNormalization)

 dense_6 (Dense)             (None, 5)                 325

=================================================================
Total params: 30,153,285
Trainable params: 30,152,677
Non-trainable params: 608
_________________________________________________________________
None
```

```
# Model Fit
model2.fit(train_generator,
          steps_per_epoch=steps_per_epoch,
          epochs=50,
          verbose=1,
          validation_data=val_generator,
          validation_steps=validation_steps,
          callbacks=callbacks_list2)
```

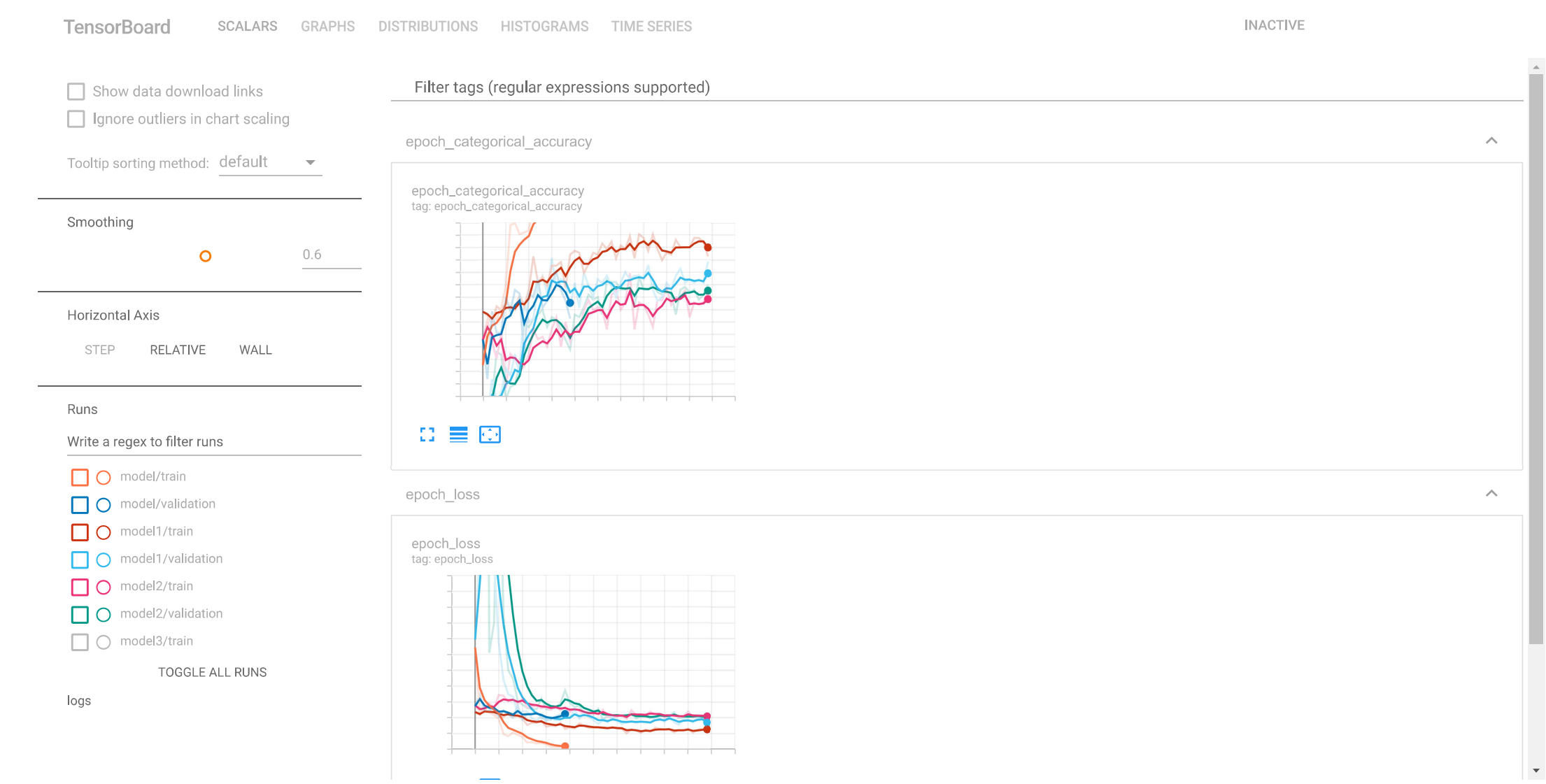
```
Source path =  /content/train ; batch size = 20
Epoch 1/50
34/34 [==============================] - ETA: 0s - loss: 1.3981 - categorical_accuracy: 0.4284Source path =  /content/val ; batch size = 20

Epoch 00001: val_loss improved from inf to 19.28531, saving model to model/model2.h5
34/34 [==============================] - 61s 2s/step - loss: 1.3981 - categorical_accuracy: 0.4284 - val_loss: 19.2853 - val_categorical_accuracy: 0.2000 - lr: 0.0010
Epoch 2/50
34/34 [==============================] - ETA: 0s - loss: 1.2044 - categorical_accuracy: 0.5098
Epoch 00002: val_loss improved from 19.28531 to 18.41357, saving model to model/model2.h5
34/34 [==============================] - 27s 820ms/step - loss: 1.2044 - categorical_accuracy: 0.5098 - val_loss: 18.4136 - val_categorical_accuracy: 0.1800 - lr: 0.0010
Epoch 3/50
34/34 [==============================] - ETA: 0s - loss: 1.3247 - categorical_accuracy: 0.4216
Epoch 00003: val_loss did not improve from 18.41357
34/34 [==============================] - 25s 757ms/step - loss: 1.3247 - categorical_accuracy: 0.4216 - val_loss: 31.1670 - val_categorical_accuracy: 0.1900 - lr: 0.0010
Epoch 4/50
34/34 [==============================] - ETA: 0s - loss: 1.4603 - categorical_accuracy: 0.3529
Epoch 00004: val_loss improved from 18.41357 to 3.05344, saving model to model/model2.h5
34/34 [==============================] - 27s 800ms/step - loss: 1.4603 - categorical_accuracy: 0.3529 - val_loss: 3.0534 - val_categorical_accuracy: 0.3800 - lr: 0.0010
Epoch 5/50
34/34 [==============================] - ETA: 0s - loss: 1.3089 - categorical_accuracy: 0.4608
Epoch 00005: val_loss did not improve from 3.05344
34/34 [==============================] - 26s 771ms/step - loss: 1.3089 - categorical_accuracy: 0.4608 - val_loss: 3.9947 - val_categorical_accuracy: 0.3700 - lr: 0.0010
Epoch 6/50
34/34 [==============================] - ETA: 0s - loss: 1.7235 - categorical_accuracy: 0.2353
Epoch 00006: val_loss did not improve from 3.05344
34/34 [==============================] - 26s 775ms/step - loss: 1.7235 - categorical_accuracy: 0.2353 - val_loss: 10.2852 - val_categorical_accuracy: 0.1900 - lr: 0.0010
Epoch 7/50
34/34 [==============================] - ETA: 0s - loss: 1.6958 - categorical_accuracy: 0.3725
Epoch 00007: val_loss did not improve from 3.05344
34/34 [==============================] - 26s 766ms/step - loss: 1.6958 - categorical_accuracy: 0.3725 - val_loss: 4.2388 - val_categorical_accuracy: 0.2400 - lr: 0.0010
Epoch 8/50
34/34 [==============================] - ETA: 0s - loss: 1.5186 - categorical_accuracy: 0.3529
Epoch 00008: val_loss improved from 3.05344 to 2.49613, saving model to model/model2.h5
34/34 [==============================] - 26s 780ms/step - loss: 1.5186 - categorical_accuracy: 0.3529 - val_loss: 2.4961 - val_categorical_accuracy: 0.2500 - lr: 0.0010
Epoch 9/50
34/34 [==============================] - ETA: 0s - loss: 1.5864 - categorical_accuracy: 0.3039
Epoch 00009: val_loss improved from 2.49613 to 2.23198, saving model to model/model2.h5
34/34 [==============================] - 26s 790ms/step - loss: 1.5864 - categorical_accuracy: 0.3039 - val_loss: 2.2320 - val_categorical_accuracy: 0.3300 - lr: 0.0010
Epoch 10/50
34/34 [==============================] - ETA: 0s - loss: 1.4191 - categorical_accuracy: 0.3235
Epoch 00010: val_loss improved from 2.23198 to 1.54431, saving model to model/model2.h5
34/34 [==============================] - 26s 793ms/step - loss: 1.4191 - categorical_accuracy: 0.3235 - val_loss: 1.5443 - val_categorical_accuracy: 0.4900 - lr: 0.0010
Epoch 11/50
34/34 [==============================] - ETA: 0s - loss: 1.6019 - categorical_accuracy: 0.3725
Epoch 00011: val_loss improved from 1.54431 to 1.41477, saving model to model/model2.h5
34/34 [==============================] - 26s 790ms/step - loss: 1.6019 - categorical_accuracy: 0.3725 - val_loss: 1.4148 - val_categorical_accuracy: 0.4800 - lr: 0.0010
Epoch 12/50
34/34 [==============================] - ETA: 0s - loss: 1.3308 - categorical_accuracy: 0.4706
Epoch 00012: val_loss improved from 1.41477 to 1.30888, saving model to model/model2.h5
34/34 [==============================] - 26s 773ms/step - loss: 1.3308 - categorical_accuracy: 0.4706 - val_loss: 1.3089 - val_categorical_accuracy: 0.5100 - lr: 0.0010
Epoch 13/50
34/34 [==============================] - ETA: 0s - loss: 1.3525 - categorical_accuracy: 0.4314
Epoch 00013: val_loss improved from 1.30888 to 1.28501, saving model to model/model2.h5
34/34 [==============================] - 26s 791ms/step - loss: 1.3525 - categorical_accuracy: 0.4314 - val_loss: 1.2850 - val_categorical_accuracy: 0.5300 - lr: 0.0010
Epoch 14/50
34/34 [==============================] - ETA: 0s - loss: 1.3801 - categorical_accuracy: 0.4412
Epoch 00014: val_loss improved from 1.28501 to 1.27724, saving model to model/model2.h5
34/34 [==============================] - 26s 785ms/step - loss: 1.3801 - categorical_accuracy: 0.4412 - val_loss: 1.2772 - val_categorical_accuracy: 0.5500 - lr: 0.0010
```

Best Epoch (Epoch 34) Result from the above Model2: **Train Accuracy : 69 Validation Accuracy : 64**

- We can clearly see that Model1 is performing better in terms of generalizing the model.
- Accuracy is also less than Model 1

```
# Tenser Board
%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 646), started 0:48:56 ago. (Use '!kill 646' to kill it.)
```




## CONV3D : Model 3

(Batch Size: 15 , No. of Frames = 30 , Optimizer=Adam , LR =0.01,Dense_Layer=32)

```
#Changing the batch Size
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=15)

#Setting the Callback Parameters
callbacks_list3 = set_checkpoint('model3',pati=6)
```

```
# Model Architecture
model3 = Sequential([
                    Conv3D(16, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform', input_shape=(30,120,120,3)),
                    BatchNormalization(),
                    Conv3D(32, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    Conv3D(64, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Conv3D(128, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Flatten(),
                    Dense(32, activation='relu', kernel_initializer='he_uniform'),
                    #Dropout(0.1),
                    BatchNormalization(),
                    Dense(5, activation='softmax')
])
```

```
## Compiling the Model
optimiser = tf.keras.optimizers.Adam(learning_rate=0.01) # write your optimizer
model3.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print (model3.summary())
```

```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d_14 (Conv3D)          (None, 28, 118, 118, 16)  1312

 batch_normalization_15 (Bat  (None, 28, 118, 118, 16)  64
 chNormalization)

 conv3d_15 (Conv3D)          (None, 26, 116, 116, 32)  13856

 batch_normalization_16 (Bat  (None, 26, 116, 116, 32)  128
 chNormalization)

 conv3d_16 (Conv3D)          (None, 24, 114, 114, 64)  55360

 batch_normalization_17 (Bat  (None, 24, 114, 114, 64)  256
 chNormalization)

 max_pooling3d_8 (MaxPooling  (None, 12, 57, 57, 64)   0
 3D)

 conv3d_17 (Conv3D)          (None, 10, 55, 55, 128)   221312

 batch_normalization_18 (Bat  (None, 10, 55, 55, 128)  512
 chNormalization)

 max_pooling3d_9 (MaxPooling  (None, 5, 27, 27, 128)   0
 3D)

 flatten_4 (Flatten)         (None, 466560)            0

 dense_7 (Dense)             (None, 32)                14929952
```

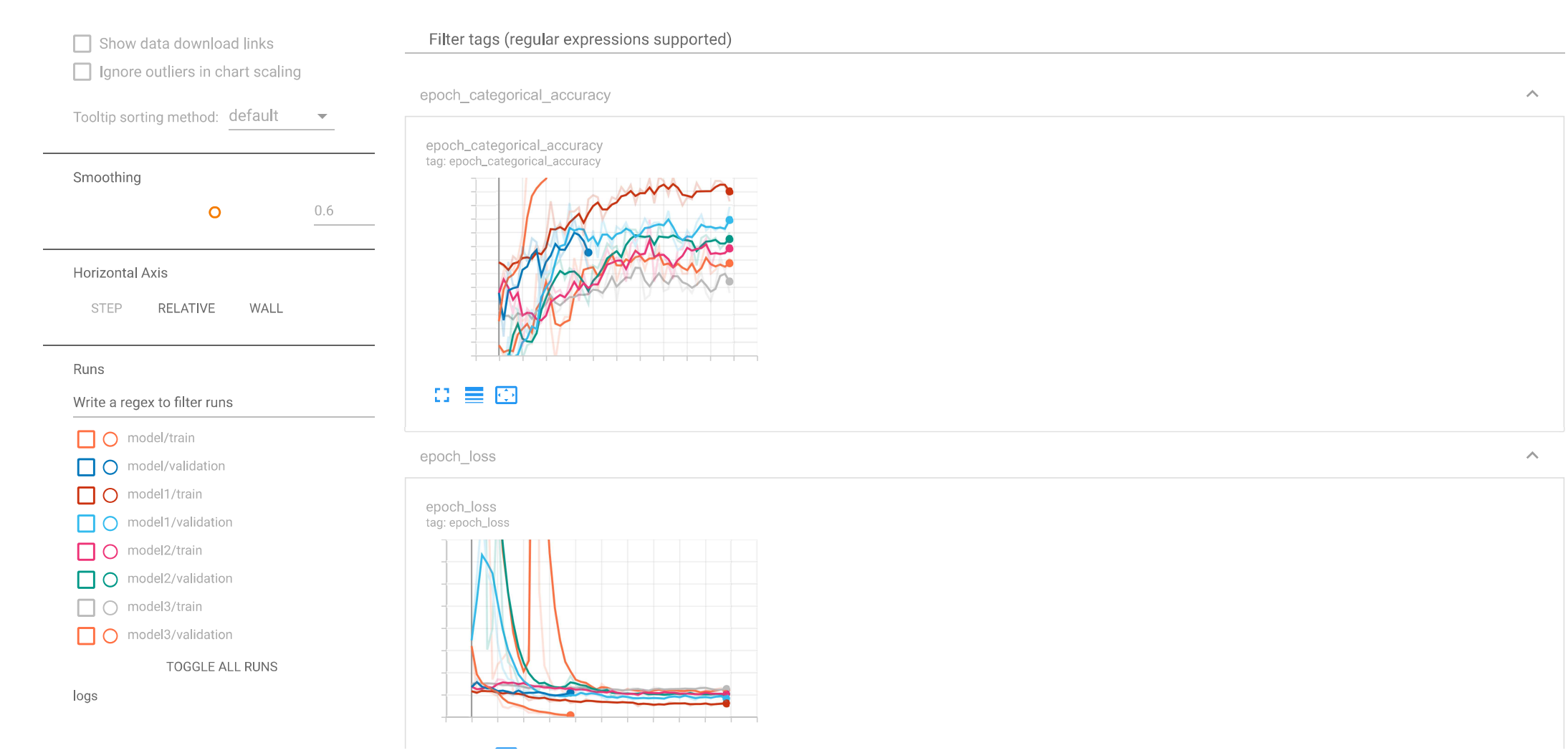
```
 batch_normalization_19 (Bat  (None, 32)              128
 chNormalization)

 dense_8 (Dense)             (None, 5)                 165

=================================================================
Total params: 15,223,045
Trainable params: 15,222,501
Non-trainable params: 544
_________________________________________________________________
None
```

```
## Model Fit
model3.fit(train_generator,
          steps_per_epoch=steps_per_epoch,
          epochs=50,
          verbose=1,
          validation_data=val_generator,
          validation_steps=validation_steps,
           callbacks=callbacks_list3)
```

```
Source path =  /content/train ; batch size = 15
Epoch 1/50
45/45 [==============================] - ETA: 0s - loss: 1.5155 - categorical_accuracy: 0.3680Source path =  /content/val ; batch size = 15

Epoch 00001: val_loss improved from inf to 119.81718, saving model to model/model3.h5
45/45 [==============================] - 53s 1s/step - loss: 1.5155 - categorical_accuracy: 0.3680 - val_loss: 119.8172 - val_categorical_accuracy: 0.2400 - lr: 0.0100
Epoch 2/50
45/45 [==============================] - ETA: 0s - loss: 1.5756 - categorical_accuracy: 0.3333
Epoch 00002: val_loss improved from 119.81718 to 47.96495, saving model to model/model3.h5
45/45 [==============================] - 22s 501ms/step - loss: 1.5756 - categorical_accuracy: 0.3333 - val_loss: 47.9649 - val_categorical_accuracy: 0.2000 - lr: 0.0100
Epoch 3/50
45/45 [==============================] - ETA: 0s - loss: 1.5092 - categorical_accuracy: 0.3481
Epoch 00003: val_loss improved from 47.96495 to 10.47097, saving model to model/model3.h5
45/45 [==============================] - 22s 499ms/step - loss: 1.5092 - categorical_accuracy: 0.3481 - val_loss: 10.4710 - val_categorical_accuracy: 0.2286 - lr: 0.0100
Epoch 4/50
45/45 [==============================] - ETA: 0s - loss: 1.5566 - categorical_accuracy: 0.3185
Epoch 00004: val_loss did not improve from 10.47097
45/45 [==============================] - 22s 504ms/step - loss: 1.5566 - categorical_accuracy: 0.3185 - val_loss: 12.9208 - val_categorical_accuracy: 0.2143 - lr: 0.0100
Epoch 5/50
45/45 [==============================] - ETA: 0s - loss: 1.5159 - categorical_accuracy: 0.3852
Epoch 00005: val_loss improved from 10.47097 to 1.71418, saving model to model/model3.h5
45/45 [==============================] - 22s 499ms/step - loss: 1.5159 - categorical_accuracy: 0.3852 - val_loss: 1.7142 - val_categorical_accuracy: 0.3571 - lr: 0.0100
Epoch 6/50
45/45 [==============================] - ETA: 0s - loss: 1.4723 - categorical_accuracy: 0.3407
Epoch 00006: val_loss did not improve from 1.71418
45/45 [==============================] - 22s 495ms/step - loss: 1.4723 - categorical_accuracy: 0.3407 - val_loss: 2.3845 - val_categorical_accuracy: 0.3286 - lr: 0.0100
Epoch 7/50
45/45 [==============================] - ETA: 0s - loss: 1.4102 - categorical_accuracy: 0.3704
Epoch 00007: val_loss did not improve from 1.71418
45/45 [==============================] - 22s 499ms/step - loss: 1.4102 - categorical_accuracy: 0.3704 - val_loss: 2.6589 - val_categorical_accuracy: 0.3429 - lr: 0.0100
Epoch 8/50
45/45 [==============================] - ETA: 0s - loss: 1.4636 - categorical_accuracy: 0.3037
Epoch 00008: val_loss did not improve from 1.71418
45/45 [==============================] - 22s 494ms/step - loss: 1.4636 - categorical_accuracy: 0.3037 - val_loss: 2.9771 - val_categorical_accuracy: 0.2429 - lr: 0.0100
Epoch 9/50
45/45 [==============================] - ETA: 0s - loss: 1.4042 - categorical_accuracy: 0.4074
Epoch 00009: val_loss improved from 1.71418 to 1.13578, saving model to model/model3.h5
45/45 [==============================] - 22s 494ms/step - loss: 1.4042 - categorical_accuracy: 0.4074 - val_loss: 1.1358 - val_categorical_accuracy: 0.5000 - lr: 0.0100
Epoch 10/50
45/45 [==============================] - ETA: 0s - loss: 1.4263 - categorical_accuracy: 0.3778
Epoch 00010: val_loss did not improve from 1.13578
45/45 [==============================] - 22s 494ms/step - loss: 1.4263 - categorical_accuracy: 0.3778 - val_loss: 1.1855 - val_categorical_accuracy: 0.4429 - lr: 0.0100
Epoch 11/50
45/45 [==============================] - ETA: 0s - loss: 1.2764 - categorical_accuracy: 0.4222
Epoch 00011: val_loss improved from 1.13578 to 1.04099, saving model to model/model3.h5
45/45 [==============================] - 22s 503ms/step - loss: 1.2764 - categorical_accuracy: 0.4222 - val_loss: 1.0410 - val_categorical_accuracy: 0.5714 - lr: 0.0100
Epoch 12/50
45/45 [==============================] - ETA: 0s - loss: 1.2662 - categorical_accuracy: 0.4667
Epoch 00012: val_loss did not improve from 1.04099
45/45 [==============================] - 22s 490ms/step - loss: 1.2662 - categorical_accuracy: 0.4667 - val_loss: 3.2940 - val_categorical_accuracy: 0.3143 - lr: 0.0100
Epoch 13/50
45/45 [==============================] - ETA: 0s - loss: 1.4883 - categorical_accuracy: 0.3556
Epoch 00013: val_loss did not improve from 1.04099
45/45 [==============================] - 22s 483ms/step - loss: 1.4883 - categorical_accuracy: 0.3556 - val_loss: 57.5957 - val_categorical_accuracy: 0.1857 - lr: 0.0100
Epoch 14/50
45/45 [==============================] - ETA: 0s - loss: 1.4666 - categorical_accuracy: 0.3704
Epoch 00014: val_loss did not improve from 1.04099
45/45 [==============================] - 22s 487ms/step - loss: 1.4666 - categorical_accuracy: 0.3704 - val_loss: 5.7077 - val_categorical_accuracy: 0.3000 - lr: 0.0100
```

```
# Visualizing the Model Results using TenserBoard

%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 646), started 1:07:45 ago. (Use '!kill 646' to kill it.)
```




Best Epoch (Epoch 48) Result from the above Model3: **Train Accuracy : 56 Validation Accuracy : 54**

## Conv3D: Model 4

- (Batch Size: 7 , No. of Frames = 30 , Optimizer=Adam , LR =0.0001,Dense_Layer=32)

```
#Changing the batch Size
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=7)

#Setting the Callback Parameters
callbacks_list3 = set_checkpoint('model4',pati=6)
```

```
model4 = Sequential([
                    Conv3D(16, kernel_size=(3, 3, 3), activation='relu', kernel_initializer='he_uniform',padding='same', input_shape=(30,120,120,3)),
                    BatchNormalization(),
                    #MaxPooling3D(pool_size=(2, 2, 2)),
                    Conv3D(32, kernel_size=(3, 3, 3), activation='relu',padding='same', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Conv3D(64, kernel_size=(3, 3, 3), activation='relu',padding='same', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Conv3D(128, kernel_size=(3, 3, 3), activation='relu',padding='same', kernel_initializer='he_uniform'),
                    BatchNormalization(),
                    MaxPooling3D(pool_size=(2, 2, 2)),
                    Flatten(),
                    Dense(32, activation='relu', kernel_initializer='he_uniform'),
                    #Dropout(0.25),
                    BatchNormalization(),
                    Dense(5, activation='softmax')
])
```

```
optimiser = tf.keras.optimizers.Adam(learning_rate=0.0001) # write your optimizer
model4.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['categorical_accuracy'])
print (model4.summary())
```

```
Model: "sequential_5"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv3d_18 (Conv3D)          (None, 30, 120, 120, 16)  1312

 batch_normalization_20 (Bat  (None, 30, 120, 120, 16)  64
 chNormalization)

 conv3d_19 (Conv3D)          (None, 30, 120, 120, 32)  13856

 batch_normalization_21 (Bat  (None, 30, 120, 120, 32)  128
 chNormalization)

 max_pooling3d_10 (MaxPoolin  (None, 15, 60, 60, 32)   0
 g3D)

 conv3d_20 (Conv3D)          (None, 15, 60, 60, 64)    55360

 batch_normalization_22 (Bat  (None, 15, 60, 60, 64)   256
 chNormalization)

 max_pooling3d_11 (MaxPoolin  (None, 7, 30, 30, 64)    0
 g3D)

 conv3d_21 (Conv3D)          (None, 7, 30, 30, 128)    221312

 batch_normalization_23 (Bat  (None, 7, 30, 30, 128)   512
 chNormalization)

 max_pooling3d_12 (MaxPoolin  (None, 3, 15, 15, 128)   0
 g3D)

 flatten_5 (Flatten)         (None, 86400)             0
```

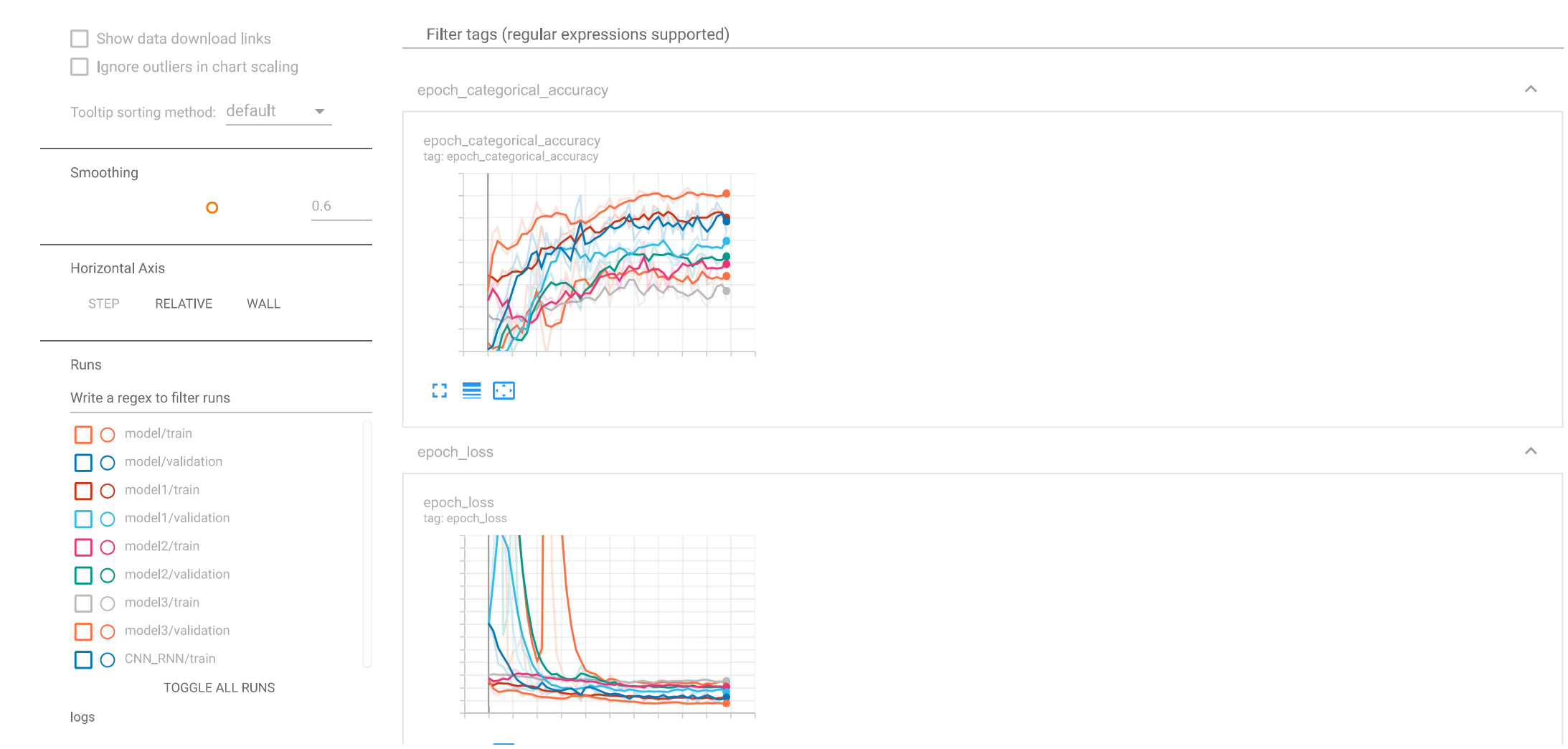
```
 dense_9 (Dense)             (None, 32)                2764832

 batch_normalization_24 (Bat  (None, 32)               128
 chNormalization)

 dense_10 (Dense)            (None, 5)                 165

=================================================================
Total params: 3,057,925
Trainable params: 3,057,381
Non-trainable params: 544
_________________________________________________________________
None
```

```
model4.fit(train_generator,
          steps_per_epoch=steps_per_epoch,
          epochs=50,
          verbose=1,
          validation_data=val_generator,
          validation_steps=validation_steps,
          callbacks=callbacks_list)
```

```
Source path =  /content/train ; batch size = 7
Epoch 1/50
95/95 [==============================] - ETA: 0s - loss: 1.3216 - categorical_accuracy: 0.4691Source path =  /content/val ; batch size = 7

Epoch 00001: val_loss did not improve from 0.94098
95/95 [==============================] - 49s 498ms/step - loss: 1.3216 - categorical_accuracy: 0.4691 - val_loss: 3.5595 - val_categorical_accuracy: 0.2100 - lr: 1.0000e-04
Epoch 2/50
95/95 [==============================] - ETA: 0s - loss: 0.7929 - categorical_accuracy: 0.7305
Epoch 00002: val_loss did not improve from 0.94098
95/95 [==============================] - 32s 337ms/step - loss: 0.7929 - categorical_accuracy: 0.7305 - val_loss: 3.0417 - val_categorical_accuracy: 0.2333 - lr: 1.0000e-04
Epoch 3/50
95/95 [==============================] - ETA: 0s - loss: 0.7289 - categorical_accuracy: 0.7549
Epoch 00003: val_loss did not improve from 0.94098
95/95 [==============================] - 26s 273ms/step - loss: 0.7289 - categorical_accuracy: 0.7549 - val_loss: 1.9516 - val_categorical_accuracy: 0.3667 - lr: 1.0000e-04
Epoch 4/50
95/95 [==============================] - ETA: 0s - loss: 0.9127 - categorical_accuracy: 0.6561
Epoch 00004: val_loss did not improve from 0.94098
95/95 [==============================] - 22s 234ms/step - loss: 0.9127 - categorical_accuracy: 0.6561 - val_loss: 1.8493 - val_categorical_accuracy: 0.4000 - lr: 1.0000e-04
Epoch 5/50
95/95 [==============================] - ETA: 0s - loss: 0.9673 - categorical_accuracy: 0.6351
Epoch 00005: val_loss did not improve from 0.94098
95/95 [==============================] - 21s 225ms/step - loss: 0.9673 - categorical_accuracy: 0.6351 - val_loss: 1.5497 - val_categorical_accuracy: 0.4667 - lr: 1.0000e-04
Epoch 6/50
95/95 [==============================] - ETA: 0s - loss: 0.9127 - categorical_accuracy: 0.6842
Epoch 00006: val_loss did not improve from 0.94098
95/95 [==============================] - 21s 225ms/step - loss: 0.9127 - categorical_accuracy: 0.6842 - val_loss: 1.0356 - val_categorical_accuracy: 0.5667 - lr: 1.0000e-04
Epoch 7/50
95/95 [==============================] - ETA: 0s - loss: 0.8693 - categorical_accuracy: 0.6982
Epoch 00007: val_loss did not improve from 0.94098
95/95 [==============================] - 21s 226ms/step - loss: 0.8693 - categorical_accuracy: 0.6982 - val_loss: 0.9585 - val_categorical_accuracy: 0.6333 - lr: 1.0000e-04
Epoch 8/50
95/95 [==============================] - ETA: 0s - loss: 0.6846 - categorical_accuracy: 0.7895
Epoch 00008: val_loss did not improve from 0.94098
95/95 [==============================] - 22s 230ms/step - loss: 0.6846 - categorical_accuracy: 0.7895 - val_loss: 0.9784 - val_categorical_accuracy: 0.5667 - lr: 1.0000e-04
Epoch 9/50
95/95 [==============================] - ETA: 0s - loss: 0.7579 - categorical_accuracy: 0.7298
Epoch 00009: val_loss did not improve from 0.94098
95/95 [==============================] - 22s 230ms/step - loss: 0.7579 - categorical_accuracy: 0.7298 - val_loss: 0.9447 - val_categorical_accuracy: 0.6000 - lr: 1.0000e-04
Epoch 10/50
95/95 [==============================] - ETA: 0s - loss: 0.7182 - categorical_accuracy: 0.7754
Epoch 00010: val_loss improved from 0.94098 to 0.87866, saving model to model/model.h5
95/95 [==============================] - 22s 231ms/step - loss: 0.7182 - categorical_accuracy: 0.7754 - val_loss: 0.8787 - val_categorical_accuracy: 0.7333 - lr: 1.0000e-04
Epoch 11/50
95/95 [==============================] - ETA: 0s - loss: 0.5791 - categorical_accuracy: 0.8632
Epoch 00011: val_loss did not improve from 0.87866
95/95 [==============================] - 21s 225ms/step - loss: 0.5791 - categorical_accuracy: 0.8632 - val_loss: 0.9839 - val_categorical_accuracy: 0.6667 - lr: 1.0000e-04
Epoch 12/50
95/95 [==============================] - ETA: 0s - loss: 0.6253 - categorical_accuracy: 0.8211
Epoch 00012: val_loss did not improve from 0.87866
95/95 [==============================] - 23s 240ms/step - loss: 0.6253 - categorical_accuracy: 0.8211 - val_loss: 1.5694 - val_categorical_accuracy: 0.4667 - lr: 1.0000e-04
Epoch 13/50
95/95 [==============================] - ETA: 0s - loss: 0.6426 - categorical_accuracy: 0.8105
Epoch 00013: val_loss improved from 0.87866 to 0.76935, saving model to model/model.h5
95/95 [==============================] - 22s 230ms/step - loss: 0.6426 - categorical_accuracy: 0.8105 - val_loss: 0.7694 - val_categorical_accuracy: 0.7333 - lr: 1.0000e-04
Epoch 14/50
95/95 [==============================] - ETA: 0s - loss: 0.6050 - categorical_accuracy: 0.8000
Epoch 00014: val_loss did not improve from 0.76935
95/95 [==============================] - 21s 223ms/step - loss: 0.6050 - categorical_accuracy: 0.8000 - val_loss: 0.8425 - val_categorical_accuracy: 0.6333 - lr: 1.0000e-04
```

```
# Visualizing the Model Results using TenserBoard

%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 646), started 1:27:08 ago. (Use '!kill 646' to kill it.)
```




Best Epoch (Epoch 88) Result from the above Model3: **Train Accuracy : 89 Validation Accuracy : 87**

- We can clearly see that Model3 is performing better in terms of generalizing the model.
- Validation Accuracy is also also better than Model 1 and Model 2

## Models : CNN + RNN

- Now we will be working on CNN + RNN Models and Transfer Learning Models

```
#Changing the batch Size
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=7)

#Setting the Callback Parameters
callbacks_list5 = set_checkpoint('CNN_RNN',pati=6)
```

```
CNN_RNN = Sequential([
                      TimeDistributed(Conv2D(16,kernel_size=(3,3),activation='relu',padding='same',kernel_regularizer='l2',bias_initializer='he_uniform',bias_regularizer='l2'),input_shape=(30,120,120,3
                      TimeDistributed(BatchNormalization()),
                      TimeDistributed(MaxPooling2D()),

                      TimeDistributed(Conv2D(32,kernel_size=(3,3),activation='relu',padding='same',kernel_regularizer='l2',bias_initializer='he_uniform',bias_regularizer='l2')),
                      TimeDistributed(BatchNormalization()),
                      # TimeDistributed(MaxPooling2D()),

                      TimeDistributed(Conv2D(64,kernel_size=(3,3),activation='relu',padding='same',kernel_regularizer='l2',bias_initializer='he_uniform',bias_regularizer='l2')),
                      TimeDistributed(BatchNormalization()),
                      # TimeDistributed(MaxPooling2D()),

                      TimeDistributed(Conv2D(128,kernel_size=(3,3),activation='relu',padding='same',kernel_regularizer='l2',bias_initializer='he_uniform',bias_regularizer='l2')),
                      TimeDistributed(BatchNormalization()),
                      TimeDistributed(MaxPooling2D()),

                      TimeDistributed(Conv2D(256,kernel_size=(3,3),activation='relu',padding='same',kernel_regularizer='l2',bias_initializer='he_uniform',bias_regularizer='l2')),
                      TimeDistributed(BatchNormalization()),
                      TimeDistributed(MaxPooling2D()),
                      TimeDistributed(Flatten()),

                      # TimeDistributed(Dropout(0.25)),
                      # TimeDistributed(Dense(64,activation='relu',kernel_regularizer='l2')),

                      GRU(16,return_sequences=False),

                      Dense(16,activation='relu',kernel_regularizer='l2'),

                      Dense(5, activation='softmax')
])
```

```
optimiser = tf.keras.optimizers.SGD(learning_rate=0.01) # write your optimizer
CNN_RNN.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['CategoricalAccuracy'])
CNN_RNN.summary()
```

```
Model: "sequential_6"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed (TimeDistr  (None, 30, 120, 120, 16)  448
 ibuted)

 time_distributed_1 (TimeDis  (None, 30, 120, 120, 16)  64
 tributed)

 time_distributed_2 (TimeDis  (None, 30, 60, 60, 16)   0
 tributed)

 time_distributed_3 (TimeDis  (None, 30, 60, 60, 32)   4640
 tributed)

 time_distributed_4 (TimeDis  (None, 30, 60, 60, 32)   128
```

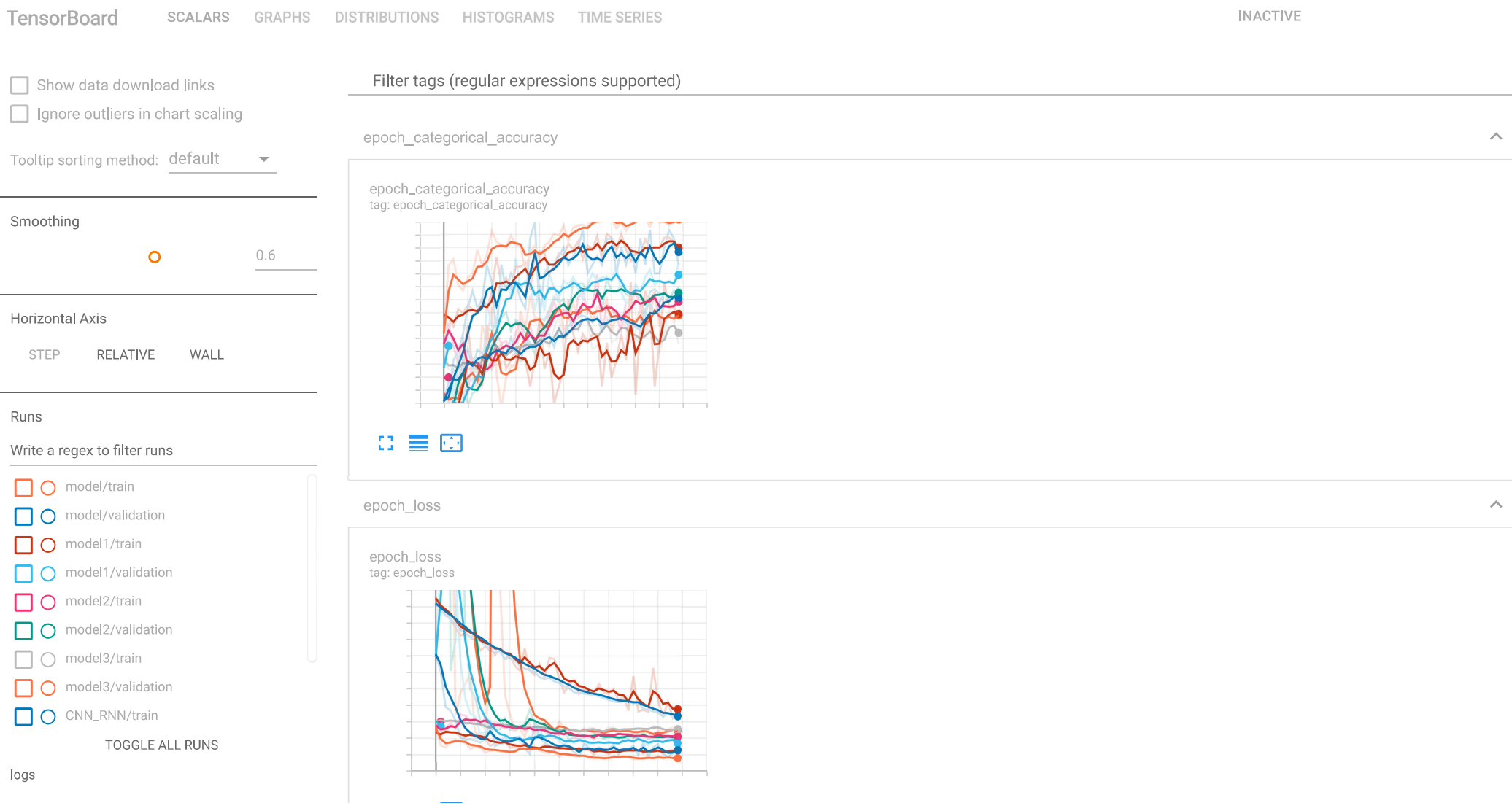
```
 tributed)

 time_distributed_5 (TimeDis  (None, 30, 60, 60, 64)   18496
 tributed)

 time_distributed_6 (TimeDis  (None, 30, 60, 60, 64)   256
 tributed)

 time_distributed_7 (TimeDis  (None, 30, 60, 60, 128)  73856
 tributed)

 time_distributed_8 (TimeDis  (None, 30, 60, 60, 128)  512
 tributed)

 time_distributed_9 (TimeDis  (None, 30, 30, 30, 128)  0
 tributed)

 time_distributed_10 (TimeDi  (None, 30, 30, 30, 256)  295168
 stributed)

 time_distributed_11 (TimeDi  (None, 30, 30, 30, 256)  1024
 stributed)

 time_distributed_12 (TimeDi  (None, 30, 15, 15, 256)  0
 stributed)

 time_distributed_13 (TimeDi  (None, 30, 57600)        0
 stributed)

 gru (GRU)                   (None, 16)                2765664

 dense_11 (Dense)            (None, 16)                272

 dense_12 (Dense)            (None, 5)                 85

=================================================================
Total params: 3,160,613
Trainable params: 3,159,621
Non-trainable params: 992
_________________________________________________________________
```

```
CNN_RNN.fit(train_generator,
          steps_per_epoch=steps_per_epoch,
          epochs=50,
          verbose=1,
          validation_data=val_generator,
          validation_steps=validation_steps,
          callbacks=callbacks_list5)

```

```
Source path =  /content/train ; batch size = 7
Epoch 1/50
95/95 [==============================] - ETA: 0s - loss: 5.0923 - categorical_accuracy: 0.2081Source path =  /content/val ; batch size = 7

Epoch 00001: val_loss improved from inf to 5.25212, saving model to model/CNN_RNN.h5
95/95 [==============================] - 52s 503ms/step - loss: 5.0923 - categorical_accuracy: 0.2081 - val_loss: 5.2521 - val_categorical_accuracy: 0.1600 - lr: 0.0100
Epoch 2/50
95/95 [==============================] - ETA: 0s - loss: 4.8982 - categorical_accuracy: 0.2821
Epoch 00002: val_loss improved from 5.25212 to 4.90046, saving model to model/CNN_RNN.h5
95/95 [==============================] - 30s 315ms/step - loss: 4.8982 - categorical_accuracy: 0.2821 - val_loss: 4.9005 - val_categorical_accuracy: 0.2000 - lr: 0.0100
Epoch 3/50
95/95 [==============================] - ETA: 0s - loss: 4.7272 - categorical_accuracy: 0.3287
Epoch 00003: val_loss improved from 4.90046 to 4.79702, saving model to model/CNN_RNN.h5
95/95 [==============================] - 23s 239ms/step - loss: 4.7272 - categorical_accuracy: 0.3287 - val_loss: 4.7970 - val_categorical_accuracy: 0.1667 - lr: 0.0100
Epoch 4/50
95/95 [==============================] - ETA: 0s - loss: 4.6158 - categorical_accuracy: 0.2912
Epoch 00004: val_loss improved from 4.79702 to 4.62067, saving model to model/CNN_RNN.h5
95/95 [==============================] - 19s 199ms/step - loss: 4.6158 - categorical_accuracy: 0.2912 - val_loss: 4.6207 - val_categorical_accuracy: 0.2000 - lr: 0.0100
Epoch 5/50
95/95 [==============================] - ETA: 0s - loss: 4.4826 - categorical_accuracy: 0.3509
Epoch 00005: val_loss improved from 4.62067 to 4.35728, saving model to model/CNN_RNN.h5
95/95 [==============================] - 19s 206ms/step - loss: 4.4826 - categorical_accuracy: 0.3509 - val_loss: 4.3573 - val_categorical_accuracy: 0.3667 - lr: 0.0100
Epoch 6/50
95/95 [==============================] - ETA: 0s - loss: 4.2950 - categorical_accuracy: 0.4105
Epoch 00006: val_loss improved from 4.35728 to 4.11698, saving model to model/CNN_RNN.h5
95/95 [==============================] - 19s 203ms/step - loss: 4.2950 - categorical_accuracy: 0.4105 - val_loss: 4.1170 - val_categorical_accuracy: 0.5000 - lr: 0.0100
Epoch 7/50
95/95 [==============================] - ETA: 0s - loss: 4.2201 - categorical_accuracy: 0.3298
Epoch 00007: val_loss did not improve from 4.11698
95/95 [==============================] - 19s 198ms/step - loss: 4.2201 - categorical_accuracy: 0.3298 - val_loss: 4.1854 - val_categorical_accuracy: 0.3333 - lr: 0.0100
Epoch 8/50
95/95 [==============================] - ETA: 0s - loss: 4.0627 - categorical_accuracy: 0.4070
Epoch 00008: val_loss improved from 4.11698 to 4.06504, saving model to model/CNN_RNN.h5
95/95 [==============================] - 18s 195ms/step - loss: 4.0627 - categorical_accuracy: 0.4070 - val_loss: 4.0650 - val_categorical_accuracy: 0.3333 - lr: 0.0100
Epoch 9/50
95/95 [==============================] - ETA: 0s - loss: 3.9726 - categorical_accuracy: 0.3930
Epoch 00009: val_loss did not improve from 4.06504
95/95 [==============================] - 19s 202ms/step - loss: 3.9726 - categorical_accuracy: 0.3930 - val_loss: 4.1699 - val_categorical_accuracy: 0.2667 - lr: 0.0100
Epoch 10/50
95/95 [==============================] - ETA: 0s - loss: 3.9525 - categorical_accuracy: 0.3579
Epoch 00010: val_loss improved from 4.06504 to 3.66747, saving model to model/CNN_RNN.h5
95/95 [==============================] - 19s 206ms/step - loss: 3.9525 - categorical_accuracy: 0.3579 - val_loss: 3.6675 - val_categorical_accuracy: 0.5000 - lr: 0.0100
Epoch 11/50
95/95 [==============================] - ETA: 0s - loss: 3.8151 - categorical_accuracy: 0.4246
Epoch 00011: val_loss did not improve from 3.66747
95/95 [==============================] - 19s 196ms/step - loss: 3.8151 - categorical_accuracy: 0.4246 - val_loss: 3.7786 - val_categorical_accuracy: 0.3333 - lr: 0.0100
Epoch 12/50
95/95 [==============================] - ETA: 0s - loss: 3.6483 - categorical_accuracy: 0.4211
Epoch 00012: val_loss did not improve from 3.66747
95/95 [==============================] - 19s 199ms/step - loss: 3.6483 - categorical_accuracy: 0.4211 - val_loss: 3.7847 - val_categorical_accuracy: 0.3667 - lr: 0.0100
Epoch 13/50
95/95 [==============================] - ETA: 0s - loss: 3.6275 - categorical_accuracy: 0.3895
Epoch 00013: val_loss improved from 3.66747 to 3.53565, saving model to model/CNN_RNN.h5
95/95 [==============================] - 20s 211ms/step - loss: 3.6275 - categorical_accuracy: 0.3895 - val_loss: 3.5356 - val_categorical_accuracy: 0.4333 - lr: 0.0100
Epoch 14/50
95/95 [==============================] - ETA: 0s - loss: 3.6418 - categorical_accuracy: 0.3895
Epoch 00014: val_loss did not improve from 3.53565
95/95 [==============================] - 18s 190ms/step - loss: 3.6418 - categorical_accuracy: 0.3895 - val_loss: 3.7158 - val_categorical_accuracy: 0.3333 - lr: 0.0100
```

```
# Visualizing the Model Results using TenserBoard

%tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 646), started 1:44:33 ago. (Use '!kill 646' to kill it.)
```




Best Epoch Result from the above Model: **Train Accuracy : 63 /Validation Accuracy : 57**

## Transfer Learning Model (Mobile Net)

- Creating a Model using transfer Learning , we will be using MobileNet architecture for Transfer learning
- Transfer Learning has less parameters

```
#Changing the batch Size
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=7)

#Setting the Callback Parameters
callbacks_list6 = set_checkpoint('MobileNet',pati=6)
```

```
## Transfer Learning using Mobile Net Architecture
mobilenet = tf.keras.applications.MobileNetV2(include_top=False)
mobilenet.trainable = True
mobilnet = Sequential([
                    TimeDistributed(mobilenet,input_shape=(30,120,120,3)),
                    TimeDistributed(BatchNormalization()),
                    TimeDistributed(Flatten()),
                    GRU(6,return_sequences=False,dropout=0.25),
                    #BatchNormalization(),
                    Dense(5,activation='softmax')
])
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_224_no_top.h5
9412608/9406464 [==============================] - 0s 0us/step
9420800/9406464 [==============================] - 0s 0us/step
```

```
## Compile the Model
optimiser = tf.keras.optimizers.SGD(learning_rate=0.001) # write your optimizer
mobilnet.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['CategoricalAccuracy'])
mobilnet.summary()
```

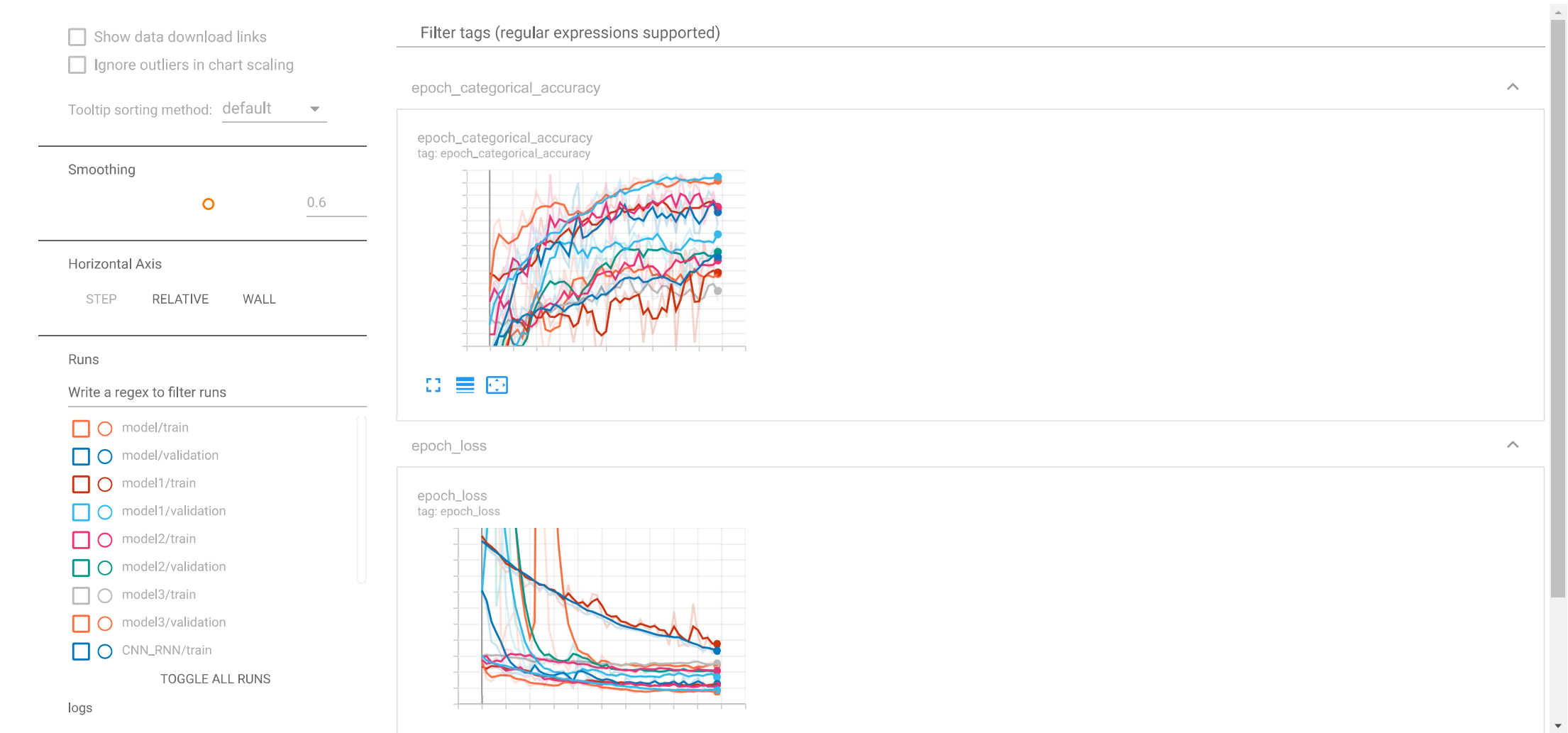
```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed_14 (TimeDi  (None, 30, 3, 3, 1280)   2257984
 stributed)

 time_distributed_15 (TimeDi  (None, 30, 3, 3, 1280)   5120
 stributed)

 time_distributed_16 (TimeDi  (None, 30, 11520)        0
 stributed)

 gru_1 (GRU)                 (None, 6)                 207504

 dense_13 (Dense)            (None, 5)                 35

=================================================================
Total params: 2,470,643
Trainable params: 2,433,971
Non-trainable params: 36,672
_________________________________________________________________
```

```
#Transfer Learning Model fit
mobilnet.fit(train_generator,
          steps_per_epoch=steps_per_epoch,
          epochs=50,
          verbose=1,
          callbacks=callbacks_list6,
          validation_data=val_generator,
          validation_steps=validation_steps)
```

```
Source path =  /content/train ; batch size = 7
Epoch 1/50
95/95 [==============================] - ETA: 0s - loss: 1.5286 - categorical_accuracy: 0.3379Source path =  /content/val ; batch size = 7

Epoch 00001: val_loss improved from inf to 1.48336, saving model to model/MobileNet.h5
95/95 [==============================] - 57s 537ms/step - loss: 1.5286 - categorical_accuracy: 0.3379 - val_loss: 1.4834 - val_categorical_accuracy: 0.3000 - lr: 0.0010
Epoch 2/50
95/95 [==============================] - ETA: 0s - loss: 1.2885 - categorical_accuracy: 0.4716
Epoch 00002: val_loss did not improve from 1.48336
95/95 [==============================] - 33s 354ms/step - loss: 1.2885 - categorical_accuracy: 0.4716 - val_loss: 1.5226 - val_categorical_accuracy: 0.3000 - lr: 0.0010
Epoch 3/50
95/95 [==============================] - ETA: 0s - loss: 1.2455 - categorical_accuracy: 0.5014
Epoch 00003: val_loss improved from 1.48336 to 1.13933, saving model to model/MobileNet.h5
95/95 [==============================] - 29s 302ms/step - loss: 1.2455 - categorical_accuracy: 0.5014 - val_loss: 1.1393 - val_categorical_accuracy: 0.5333 - lr: 0.0010
Epoch 4/50
95/95 [==============================] - ETA: 0s - loss: 1.2300 - categorical_accuracy: 0.5018
Epoch 00004: val_loss did not improve from 1.13933
95/95 [==============================] - 23s 244ms/step - loss: 1.2300 - categorical_accuracy: 0.5018 - val_loss: 1.1589 - val_categorical_accuracy: 0.5333 - lr: 0.0010
Epoch 5/50
95/95 [==============================] - ETA: 0s - loss: 1.1705 - categorical_accuracy: 0.5719
Epoch 00005: val_loss improved from 1.13933 to 1.09619, saving model to model/MobileNet.h5
95/95 [==============================] - 24s 255ms/step - loss: 1.1705 - categorical_accuracy: 0.5719 - val_loss: 1.0962 - val_categorical_accuracy: 0.6000 - lr: 0.0010
Epoch 6/50
95/95 [==============================] - ETA: 0s - loss: 1.1863 - categorical_accuracy: 0.5123
Epoch 00006: val_loss improved from 1.09619 to 0.89137, saving model to model/MobileNet.h5
95/95 [==============================] - 23s 245ms/step - loss: 1.1863 - categorical_accuracy: 0.5123 - val_loss: 0.8914 - val_categorical_accuracy: 0.7000 - lr: 0.0010
Epoch 7/50
95/95 [==============================] - ETA: 0s - loss: 1.0956 - categorical_accuracy: 0.6105
Epoch 00007: val_loss did not improve from 0.89137
95/95 [==============================] - 24s 249ms/step - loss: 1.0956 - categorical_accuracy: 0.6105 - val_loss: 1.1298 - val_categorical_accuracy: 0.5667 - lr: 0.0010
Epoch 8/50
95/95 [==============================] - ETA: 0s - loss: 1.0659 - categorical_accuracy: 0.6140
Epoch 00008: val_loss did not improve from 0.89137
95/95 [==============================] - 23s 239ms/step - loss: 1.0659 - categorical_accuracy: 0.6140 - val_loss: 1.1294 - val_categorical_accuracy: 0.5000 - lr: 0.0010
Epoch 9/50
95/95 [==============================] - ETA: 0s - loss: 1.0268 - categorical_accuracy: 0.6175
Epoch 00009: val_loss did not improve from 0.89137
95/95 [==============================] - 25s 261ms/step - loss: 1.0268 - categorical_accuracy: 0.6175 - val_loss: 1.2509 - val_categorical_accuracy: 0.5000 - lr: 0.0010
Epoch 10/50
95/95 [==============================] - ETA: 0s - loss: 0.9709 - categorical_accuracy: 0.6491
Epoch 00010: val_loss improved from 0.89137 to 0.85638, saving model to model/MobileNet.h5
95/95 [==============================] - 24s 251ms/step - loss: 0.9709 - categorical_accuracy: 0.6491 - val_loss: 0.8564 - val_categorical_accuracy: 0.7667 - lr: 0.0010
Epoch 11/50
95/95 [==============================] - ETA: 0s - loss: 0.9334 - categorical_accuracy: 0.7123
Epoch 00011: val_loss did not improve from 0.85638
95/95 [==============================] - 23s 241ms/step - loss: 0.9334 - categorical_accuracy: 0.7123 - val_loss: 0.8875 - val_categorical_accuracy: 0.6667 - lr: 0.0010
Epoch 12/50
95/95 [==============================] - ETA: 0s - loss: 0.9366 - categorical_accuracy: 0.6667
Epoch 00012: val_loss did not improve from 0.85638
95/95 [==============================] - 23s 242ms/step - loss: 0.9366 - categorical_accuracy: 0.6667 - val_loss: 0.9344 - val_categorical_accuracy: 0.7000 - lr: 0.0010
Epoch 13/50
95/95 [==============================] - ETA: 0s - loss: 0.8739 - categorical_accuracy: 0.7088
Epoch 00013: val_loss did not improve from 0.85638
95/95 [==============================] - 24s 255ms/step - loss: 0.8739 - categorical_accuracy: 0.7088 - val_loss: 0.8702 - val_categorical_accuracy: 0.6333 - lr: 0.0010
Epoch 14/50
95/95 [==============================] - ETA: 0s - loss: 0.8687 - categorical_accuracy: 0.7298
Epoch 00014: val_loss improved from 0.85638 to 0.58786, saving model to model/MobileNet.h5
95/95 [==============================] - 23s 243ms/step - loss: 0.8687 - categorical_accuracy: 0.7298 - val_loss: 0.5879 - val_categorical_accuracy: 0.9333 - lr: 0.0010
```

```
1 # Visualizing the Model Results using TenserBoard
2
3 %tensorboard --logdir logs
```

```
Reusing TensorBoard on port 6006 (pid 646), started 2:05:03 ago. (Use '!kill 646' to kill it.)
```




Best Epoch Result from the above Model: **Train Accuracy : 93 Validation Accuracy : 90**

We can clearly see that Transfer Learning Model using GRU is performing best.

## Transfer Learning Model(Mobile Net : Trainable = False)

```
#Changing the batch Size
train_generator,val_generator,steps_per_epoch,validation_steps = gen_train_val(batch_size=15)
```

```
## Transfer Learning using Mobile Net Architecture
mobilenetnt = tf.keras.applications.MobileNetV2(include_top=False)
mobilenetnt.trainable = False
mobile = Sequential([
                    TimeDistributed(mobilenetnt,input_shape=(30,120,120,3)),
                    TimeDistributed(BatchNormalization()),
                    TimeDistributed(Flatten()),
                    GRU(6,return_sequences=False,dropout=0.25),
                    Dense(5,activation='softmax')
])
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the default.
```

```
## Compile the Model
optimiser = tf.keras.optimizers.SGD(learning_rate=0.001) # write your optimizer
mobile.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['CategoricalAccuracy'])
mobile.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 time_distributed_17 (TimeDi  (None, 30, 3, 3, 1280)   2257984
 stributed)

 time_distributed_18 (TimeDi  (None, 30, 3, 3, 1280)   5120
 stributed)

 time_distributed_19 (TimeDi  (None, 30, 11520)        0
 stributed)

 gru_2 (GRU)                 (None, 6)                 207504

 dense_14 (Dense)            (None, 5)                 35

=================================================================
Total params: 2,470,643
Trainable params: 210,099
Non-trainable params: 2,260,544
_________________________________________________________________
```

```
#Transfer Learning Model fit
mobile.fit(train_generator,
          steps_per_epoch=steps_per_epoch,
          epochs=50,
          verbose=1,
          callbacks=callbacks_list7,
          validation_data=val_generator,
          validation_steps=validation_steps)
```

```
Source path =  /content/train ; batch size = 15
Epoch 1/50
45/45 [==============================] - ETA: 0s - loss: 1.5471 - categorical_accuracy: 0.3363Source path =  /content/val ; batch size = 15

Epoch 00001: val_loss improved from inf to 1.56592, saving model to model/MobileNetNonTrainable.h5
45/45 [==============================] - 55s 1s/step - loss: 1.5471 - categorical_accuracy: 0.3363 - val_loss: 1.5659 - val_categorical_accuracy: 0.3200 - lr: 0.0010
Epoch 2/50
45/45 [==============================] - ETA: 0s - loss: 1.4612 - categorical_accuracy: 0.4222
Epoch 00002: val_loss improved from 1.56592 to 1.32979, saving model to model/MobileNetNonTrainable.h5
45/45 [==============================] - 16s 374ms/step - loss: 1.4612 - categorical_accuracy: 0.4222 - val_loss: 1.3298 - val_categorical_accuracy: 0.4429 - lr: 0.0010
Epoch 3/50
45/45 [==============================] - ETA: 0s - loss: 1.3197 - categorical_accuracy: 0.4444
Epoch 00003: val_loss improved from 1.32979 to 1.22636, saving model to model/MobileNetNonTrainable.h5
45/45 [==============================] - 16s 365ms/step - loss: 1.3197 - categorical_accuracy: 0.4444 - val_loss: 1.2264 - val_categorical_accuracy: 0.4143 - lr: 0.0010
Epoch 4/50
45/45 [==============================] - ETA: 0s - loss: 1.2844 - categorical_accuracy: 0.4593
Epoch 00004: val_loss improved from 1.22636 to 1.22389, saving model to model/MobileNetNonTrainable.h5
45/45 [==============================] - 15s 350ms/step - loss: 1.2844 - categorical_accuracy: 0.4593 - val_loss: 1.2239 - val_categorical_accuracy: 0.4429 - lr: 0.0010
Epoch 5/50
45/45 [==============================] - ETA: 0s - loss: 1.2501 - categorical_accuracy: 0.4741
Epoch 00005: val_loss improved from 1.22389 to 1.18851, saving model to model/MobileNetNonTrainable.h5
45/45 [==============================] - 16s 368ms/step - loss: 1.2501 - categorical_accuracy: 0.4741 - val_loss: 1.1885 - val_categorical_accuracy: 0.4571 - lr: 0.0010
Epoch 6/50
45/45 [==============================] - ETA: 0s - loss: 1.2013 - categorical_accuracy: 0.5704
Epoch 00006: val_loss improved from 1.18851 to 1.16508, saving model to model/MobileNetNonTrainable.h5
45/45 [==============================] - 17s 375ms/step - loss: 1.2013 - categorical_accuracy: 0.5704 - val_loss: 1.1651 - val_categorical_accuracy: 0.4571 - lr: 0.0010
Epoch 7/50
45/45 [==============================] - ETA: 0s - loss: 1.0523 - categorical_accuracy: 0.5778
Epoch 00007: val_loss did not improve from 1.16508
45/45 [==============================] - 15s 350ms/step - loss: 1.0523 - categorical_accuracy: 0.5778 - val_loss: 1.1831 - val_categorical_accuracy: 0.4429 - lr: 0.0010
Epoch 8/50
45/45 [==============================] - ETA: 0s - loss: 1.1176 - categorical_accuracy: 0.6222
Epoch 00008: val_loss improved from 1.16508 to 1.15055, saving model to model/MobileNetNonTrainable.h5
```