GESTURE RECOGNITION PROJECT

- Our aim is to build a 3D Convolutional Network and a CNN+RNN architecture that will be able to predict the 5 gestures correctly.

PROBLEM STATEMENT :

A home electronics company manufactures state of the art smart televisions. We have to develop a cool feature in the smart-TV that can recognise **five different gestures** performed by the user which will help users control the TV without using a remote. Each video is a sequence of 30 frames (or images).

The gestures are continuously monitored by the webcam mounted on the TV. Each gesture corresponds to a specific command:

- **Thumbs Up** - Increase the volume
- **Thumbs Down** - Decrease the volume
- **Left Swipe** - 'Jump' backwards 10 seconds
- **Right Swipe** - 'Jump' forward 10 seconds
- **Stop** - Pause the movie

For analysing videos using neural networks, **two types of architectures** are used commonly.

- One is the **standard CNN + RNN architecture** in which you pass the images of a video through a CNN which extracts a feature vector for each image, and then pass the sequence of these feature vectors through an RNN.
- The other popular architecture used to process videos is a natural extension of CNNs - **a 3D convolutional network**.

OBJECTIVES :

Generator: To define a generator that should be able to take a batch of videos as input. Steps like cropping, resizing and normalization are to be performed.

Model: To develop a model that is able to train the data without any errors.

- **We have tried various models with 3D convolutional networks and 2D CNN+RNN networks. Among those models, we propose the final model here with 2D convolutional + GRU network. Also, we implemented the transfer learning model to get better accuracy.**
- **Number of parameters in this proposed model is very less compared to other models that are tried previously.**

FINAL MODEL

In [1]:

```
# Import required libraries

import numpy as np
import pandas as pd
import os
from imageio import imread
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
from skimage.transform import resize
import datetime
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
pd.set_option('display.max_rows', 500)
```

In [2]:

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

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

In [3]:

```
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 [10]:

```
# Function to plot the training/validation accuracies/losses.

def plot_model(history):
    fig, axes = plt.subplots(nrows=1, ncols=2, figsize=(15,4))
    axes[0].plot(history.history['loss'])
    axes[0].plot(history.history['val_loss'])
    axes[0].grid()
    axes[0].legend(['loss','val_loss'])
    axes[1].plot(history.history['categorical_accuracy'])
    axes[1].plot(history.history['val_categorical_accuracy'])
    axes[1].grid()
    axes[1].legend(['categorical_accuracy','val_categorical_accuracy'])
```

In [5]:

```
class BuildModel:

    # Function to initialize the paths
    def path_initialization(self,source_path):
        self.train_doc = np.random.permutation(open(source_path + '/' + 'train.csv').readlines())
        self.val_doc = np.random.permutation(open(source_path + '/' + 'val.csv').readlines())
        self.train_path = source_path + '/' + 'train'
        self.val_path =  source_path + '/' + 'val'
        self.num_train_sequences = len(self.train_doc)
        self.num_val_sequences = len(self.val_doc)

                                             lues
    def values_initialization(self,image_height,image_width,sampled_frames,batch_size,num_of_epochs):
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
        self.image_height=image_height
        self.image_width=image_width
        self.sampled_frames=sampled_frames
        self.batch_size=batch_size
        self.num_epochs=num_of_epochs
        self.channels=3
        self.num_classes=5

    # Data generator function
    def generator(self, source_path, folder_list, batch_size, aug=False):
        print( 'Source path = ', source_path, '; batch size =', batch_size)
        num_batches = len(folder_list)//batch_size
        mod_batch = len(folder_list)%batch_size
        while True:
            t = np.random.permutation(folder_list)
            for batch in range(num_batches):
                batch_data = np.zeros((batch_size, self.sampled_frames, self.image_height, self.image_width, self.channels))
                batch_labels = np.zeros((batch_size, 5))
                if(aug):
                    data_aug = np.zeros((batch_size, self.sampled_frames, self.image_height, self.image_width, self.channels))
                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 over 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)
                        if(aug):

                            # Cropping the images
                            # If the image is not of square shape,lets crop it into square so that the gesture is more focussed

                            if image.shape[1] > image.shape[0]:
                                difference_value = image.shape[1] - image.shape[0]
                                crop_start = difference_value//2
                                crop_end = crop_start + image.shape[0]
                                image = image[:, crop_start:crop_end]
                            elif image.shape[0] > image.shape[1]:
                                difference_value = image.shape[0] - image.shape[1]
                                crop_start = difference_value//2
                                crop_end = crop_start + image.shape[1]
                                image = image[crop_start:crop_end,:]

                            # Resizing the images
                            image_resized=resize(image, (self.image_height, self.image_width, self.channels))

                            # Normalization
                            data_aug[folder,idx,:,:,0] = (image_resized[:,:,0])/255
                            data_aug[folder,idx,:,:,1] = (image_resized[:,:,1])/255
                            data_aug[folder,idx,:,:,2] = (image_resized[:,:,2])/255

                        else:
                            # Resizing the image
                            image_resized=resize(image, (self.image_height, self.image_width, self.channels))

                            # Normalization
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
                            batch_data[folder,idx,:,:,0] = (image_resized[:,:,0])/255
                            batch_data[folder,idx,:,:,1] = (image_resized[:,:,1])/255
                            batch_data[folder,idx,:,:,2] = (image_resized[:,:,2])/255

                    batch_labels[folder, int(t[folder+(batch*batch_size)].strip().split(';')[2])] = 1
                    if(aug):
                        batch_data = data_aug
                yield batch_data, batch_labels

            # Code for the remaining data points which are left after full batches
            if(mod_batch!=0):
                batch_data = np.zeros((batch_size, self.sampled_frames, self.image_height, self.image_width, self.channels))
                batch_labels = np.zeros((batch_size, 5))
                if(aug):
                    data_aug = np.zeros((batch_size, self.sampled_frames, self.image_height, self.image_width, self.channels))
                for folder in range(mod_batch): # iterate over the mod_batch
                    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 over 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)
                        if(aug):

                            # Cropping the images
                            # If the image is not of square shape,lets crop it into square so that the gesture is more focussed

                            if image.shape[1] > image.shape[0]:
                                difference_value = image.shape[1] - image.shape[0]
                                crop_start = difference_value//2
                                crop_end = crop_start + image.shape[0]
                                image = image[:, crop_start:crop_end]
                            elif image.shape[0] > image.shape[1]:
                                difference_value = image.shape[0] - image.shape[1]
                                crop_start = difference_value//2
                                crop_end = crop_start + image.shape[1]
                                image = image[crop_start:crop_end,:]

                            # Resizing the images
                            image_resized=resize(image, (self.image_height, self.image_width, self.channels))

                            # Normalization
                            data_aug[folder,idx,:,:,0] = (image_resized[:,:,0])/255
                            data_aug[folder,idx,:,:,1] = (image_resized[:,:,1])/255
                            data_aug[folder,idx,:,:,2] = (image_resized[:,:,2])/255

                        else:
                            # Resizing the image
                            image_resized=resize(image, (self.image_height, self.image_width, self.channels))

                            # Normalization
                            batch_data[folder,idx,:,:,0] = (image_resized[:,:,0])/255
                            batch_data[folder,idx,:,:,1] = (image_resized[:,:,1])/255
                            batch_data[folder,idx,:,:,2] = (image_resized[:,:,2])/255

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

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
                    if(aug):
                        batch_data = data_aug
                yield batch_data, batch_labels

    def CNN_GRU_Model_definition(self):

        # 2D CONVOLUTIONAL NETWORK + GRU

        # Number of TimeDistributed 2D convolutional Layers = 4
        # With GRU layer
        # 2 Dense TimeDistributed layers
        # 1 Dense Softmax output layer
        # Kernel size = (3,3)
        # Pool size = (2,2)
        # With dropout layers
        # Learning rate = 0.0002
        # Activation function - 'relu'

        model = Sequential()
        model.add(TimeDistributed(Conv2D(8, (3,3), activation='relu'), input_shape=(self.sampled_frames, self.image_height, self.image_width, self.channels)))
        model.add(TimeDistributed(MaxPooling2D((2,2))))

        model.add(TimeDistributed(Conv2D(16, (3,3), activation='relu')))
        model.add(TimeDistributed(MaxPooling2D((2,2))))

        model.add(TimeDistributed(Conv2D(16, (3,3), activation='relu')))
        model.add(TimeDistributed(MaxPooling2D((2,2))))

        model.add(TimeDistributed(Conv2D(32, (3,3), activation='relu')))
        model.add(TimeDistributed(MaxPooling2D((2,2))))

        model.add(TimeDistributed(Flatten()))

        model.add(TimeDistributed(Dense(64, activation='relu')))
        model.add(TimeDistributed(Dense(128, activation='relu')))
        model.add(Dropout(0.30))

        model.add(GRU(32))
        model.add(Dropout(0.50))

        model.add(Dense(self.num_classes,activation='softmax'))

        # Compile the model

        optimiser = optimizers.Adam(learning_rate=0.0002)
        model.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['categorical_accuracy'])

        return model


    def TransferLearning_GRU_Model_definition(self, mobilenet_transfer):

        # Transfer Learning + GRU
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
        # With dropout layers
        # Activation function - 'relu'

        model = Sequential()
        model.add(TimeDistributed(mobilenet_transfer,input_shape=(self.sampled_frames, self.image_height, self.image_width, self.channels)))
        model.add(TimeDistributed(BatchNormalization()))
        model.add(TimeDistributed(MaxPooling2D((2, 2))))
        model.add(Dropout(0.30))

        model.add(TimeDistributed(Flatten()))

        model.add(GRU(16))
        model.add(Dropout(0.35))

        model.add(Dense(64,activation='relu'))
        model.add(Dropout(0.40))

        model.add(Dense(self.num_classes, activation='softmax'))

        # Compile the model

        optimiser = optimizers.Adam(learning_rate=0.0003)
        model.compile(optimizer=optimiser, loss='categorical_crossentropy', metrics=['categorical_accuracy'])

        return model


    def Model_Training(self, model, factor_value, patience_lr, patience_early_stop, augment=False):

        curr_dt_time = datetime.datetime.now()

        # Train generator and Validation generator

        train_generator = self.generator(self.train_path, self.train_doc, self.batch_size,aug=augment)
        val_generator = self.generator(self.val_path, self.val_doc, self.batch_size)

        model_name = 'model_init' + '_' + str(curr_dt_time).replace(' ','').replace(':','_') + '/'
        if not os.path.exists(model_name):
            os.mkdir(model_name)

        filepath = model_name + 'model-{epoch:05d}-{loss:.5f}-{categorical_accuracy:.5f}-{val_loss:.5f}-{val_categorical_accuracy:.5f}.h5'

        # Model checkpoint to save the best model

        checkpoint = ModelCheckpoint(filepath, monitor='val_loss', verbose=1, save_best_only=False,
                                     save_weights_only=False, mode='auto')

        # Reduce learning rate when a metric has stopped improving.This callback monitors a quantity
        # and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.

        LR = ReduceLROnPlateau(monitor="val_loss",
            factor=factor_value,
            patience=patience_lr,
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
            verbose=0,
            mode="auto",
            cooldown=0,
            min_lr=0)

        # Early stopping is a method that allows to specify an arbitrary large number of training epochs.
        # And stops training once the model performance stops improving on a hold out validation dataset.

        EarlyStop = EarlyStopping(monitor='val_loss',
            min_delta=0,
            patience=patience_early_stop,
            verbose=0,
            mode='auto',
            baseline=None,
            restore_best_weights=False)

        # List of callbacks

        callbacks_list = [checkpoint, LR, EarlyStop]

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

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

        # Fit the model on the data

        fitted_model = model.fit_generator(train_generator, steps_per_epoch = steps_per_epoch, epochs = self.num_epochs,
                       verbose=1,callbacks=callbacks_list, validation_data=val_generator,validation_steps=validation_steps,
                       class_weight=None, workers=1, initial_epoch=0)
        return fitted_model
```

FINAL MODEL - 2D CNN + GRU Network Model

- **2D convolutional network + GRU model**
- **With augmentation**
- **With dropout layers**
- **4 2D-convolutional layers**
- **2 Dense TimeDistributed layers**
- **1 GRU layer**
- **1 softmax Dense output layer**
- **Activation function** : 'relu'

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

- **Learning rate** : 0.0002
- **Image height** = 100
- **Image width** = 100
- **Batch size** = 20
- **Number of epochs** = 70
- **Metrics** : Accuracy

In [6]:

```
# Passing source path for initialising all the required paths

source_path = 'C:/Users/gayth/Project_data'

# Frame indices to be selected among the available 30 frames per sequence

img_idx = [3,6,9,11,13,15,17,19,21,24,27]
num_of_selected_frames = len(img_idx)

# Instantiating the object for BuildModel() class

Model_Conv2D_GRU = BuildModel()

# Image height = 100
# Image width = 100
# Batch_size = 20
# Number of epochs = 65

Model_Conv2D_GRU.path_initialization(source_path)
Model_Conv2D_GRU.values_initialization(100,100,num_of_selected_frames,20,65)

# Define the 2D Conv + GRU network model using function 'CNN_GRU_Model_definition'

Model_1 = Model_Conv2D_GRU.CNN_GRU_Model_definition()
Model_1.summary()

# Pass the model to train the data and obtain the results using Model_Training() function

Model_ConvGRU = Model_Conv2D_GRU.Model_Training(Model_1,0.5,10,30,augment=True)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 time_distributed (TimeDistr  (None, 11, 98, 98, 8)    224       
 ibuted)                                                         
                                                                 
 time_distributed_1 (TimeDis  (None, 11, 49, 49, 8)    0         
 tributed)                                                       
                                                                 
 time_distributed_2 (TimeDis  (None, 11, 47, 47, 16)   1168      
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
 time_distributed_3 (TimeDis  (None, 11, 23, 23, 16)   0
 tributed)

 time_distributed_4 (TimeDis  (None, 11, 21, 21, 16)   2320
 tributed)

 time_distributed_5 (TimeDis  (None, 11, 10, 10, 16)   0
 tributed)

 time_distributed_6 (TimeDis  (None, 11, 8, 8, 32)     4640
 tributed)

 time_distributed_7 (TimeDis  (None, 11, 4, 4, 32)     0
 tributed)

 time_distributed_8 (TimeDis  (None, 11, 512)          0
 tributed)

 time_distributed_9 (TimeDis  (None, 11, 64)           32832
 tributed)

 time_distributed_10 (TimeDi  (None, 11, 128)          8320
 stributed)

 dropout (Dropout)           (None, 11, 128)           0

 gru (GRU)                   (None, 32)                15552

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 5)                 165

=================================================================
Total params: 65,221
Trainable params: 65,221
Non-trainable params: 0
_________________________________________________________________
Source path =  C:/Users/gayth/Project_data/train ; batch size = 20
Epoch 1/65
34/34 [==============================] - ETA: 0s - loss: 1.5721 - categorical_accuracy: 0.2353Source path =  C:/Users/gayth/Project_data/val ; batch size = 20

Epoch 1: saving model to model_init_2022-09-1518_15_00.104107\model-00001-1.57206-0.23529-1.59984-0.18000.h5
34/34 [==============================] - 65s 2s/step - loss: 1.5721 - categorical_accuracy: 0.2353 - val_loss: 1.5998 - val_categorical_accuracy: 0.1800 - lr:
2.0000e-04
Epoch 2/65
34/34 [==============================] - ETA: 0s - loss: 1.5612 - categorical_accuracy: 0.2074
Epoch 2: saving model to model_init_2022-09-1518_15_00.104107\model-00002-1.56121-0.20735-1.58612-0.32000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.5612 - categorical_accuracy: 0.2074 - val_loss: 1.5861 - val_categorical_accuracy: 0.3200 - lr:
2.0000e-04
Epoch 3/65
34/34 [==============================] - ETA: 0s - loss: 1.5530 - categorical_accuracy: 0.2397
Epoch 3: saving model to model_init_2022-09-1518_15_00.104107\model-00003-1.55295-0.23971-1.59036-0.25000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.5530 - categorical_accuracy: 0.2397 - val_loss: 1.5904 - val_categorical_accuracy: 0.2500 - lr:
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
2.0000e-04
Epoch 4/65
34/34 [==============================] - ETA: 0s - loss: 1.5459 - categorical_accuracy: 0.2485
Epoch 4: saving model to model_init_2022-09-1518_15_00.104107\model-00004-1.54587-0.24853-1.57549-0.32000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.5459 - categorical_accuracy: 0.2485 - val_loss: 1.5755 - val_categorical_accuracy: 0.3200 - lr:
2.0000e-04
Epoch 5/65
34/34 [==============================] - ETA: 0s - loss: 1.5302 - categorical_accuracy: 0.2926
Epoch 5: saving model to model_init_2022-09-1518_15_00.104107\model-00005-1.53023-0.29265-1.54569-0.38000.h5
34/34 [==============================] - 60s 2s/step - loss: 1.5302 - categorical_accuracy: 0.2926 - val_loss: 1.5457 - val_categorical_accuracy: 0.3800 - lr:
2.0000e-04
Epoch 6/65
34/34 [==============================] - ETA: 0s - loss: 1.4924 - categorical_accuracy: 0.3147
Epoch 6: saving model to model_init_2022-09-1518_15_00.104107\model-00006-1.49239-0.31471-1.49485-0.54000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.4924 - categorical_accuracy: 0.3147 - val_loss: 1.4949 - val_categorical_accuracy: 0.5400 - lr:
2.0000e-04
Epoch 7/65
34/34 [==============================] - ETA: 0s - loss: 1.4461 - categorical_accuracy: 0.3544
Epoch 7: saving model to model_init_2022-09-1518_15_00.104107\model-00007-1.44612-0.35441-1.42674-0.50000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.4461 - categorical_accuracy: 0.3544 - val_loss: 1.4267 - val_categorical_accuracy: 0.5000 - lr:
2.0000e-04
Epoch 8/65
34/34 [==============================] - ETA: 0s - loss: 1.3728 - categorical_accuracy: 0.4176
Epoch 8: saving model to model_init_2022-09-1518_15_00.104107\model-00008-1.37278-0.41765-1.44841-0.38000.h5
34/34 [==============================] - 60s 2s/step - loss: 1.3728 - categorical_accuracy: 0.4176 - val_loss: 1.4484 - val_categorical_accuracy: 0.3800 - lr:
2.0000e-04
Epoch 9/65
34/34 [==============================] - ETA: 0s - loss: 1.3050 - categorical_accuracy: 0.4485
Epoch 9: saving model to model_init_2022-09-1518_15_00.104107\model-00009-1.30497-0.44853-1.28570-0.53000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.3050 - categorical_accuracy: 0.4485 - val_loss: 1.2857 - val_categorical_accuracy: 0.5300 - lr:
2.0000e-04
Epoch 10/65
34/34 [==============================] - ETA: 0s - loss: 1.2187 - categorical_accuracy: 0.4971
Epoch 10: saving model to model_init_2022-09-1518_15_00.104107\model-00010-1.21875-0.49706-1.21798-0.48000.h5
34/34 [==============================] - 61s 2s/step - loss: 1.2187 - categorical_accuracy: 0.4971 - val_loss: 1.2180 - val_categorical_accuracy: 0.4800 - lr:
2.0000e-04
Epoch 11/65
34/34 [==============================] - ETA: 0s - loss: 1.1786 - categorical_accuracy: 0.5176
Epoch 11: saving model to model_init_2022-09-1518_15_00.104107\model-00011-1.17864-0.51765-1.12861-0.58000.h5
34/34 [==============================] - 60s 2s/step - loss: 1.1786 - categorical_accuracy: 0.5176 - val_loss: 1.1286 - val_categorical_accuracy: 0.5800 - lr:
2.0000e-04
Epoch 12/65
34/34 [==============================] - ETA: 0s - loss: 1.1171 - categorical_accuracy: 0.5441
Epoch 12: saving model to model_init_2022-09-1518_15_00.104107\model-00012-1.11706-0.54412-1.05643-0.66000.h5
34/34 [==============================] - 60s 2s/step - loss: 1.1171 - categorical_accuracy: 0.5441 - val_loss: 1.0564 - val_categorical_accuracy: 0.6600 - lr:
2.0000e-04
Epoch 13/65
34/34 [==============================] - ETA: 0s - loss: 1.0766 - categorical_accuracy: 0.5676
Epoch 13: saving model to model_init_2022-09-1518_15_00.104107\model-00013-1.07655-0.56765-1.10330-0.63000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.0766 - categorical_accuracy: 0.5676 - val_loss: 1.1033 - val_categorical_accuracy: 0.6300 - lr:
2.0000e-04
Epoch 14/65
34/34 [==============================] - ETA: 0s - loss: 1.0201 - categorical_accuracy: 0.5882
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Epoch 14: saving model to model_init_2022-09-1518_15_00.104107\model-00014-1.02013-0.58824-1.08143-0.58000.h5
34/34 [==============================] - 59s 2s/step - loss: 1.0201 - categorical_accuracy: 0.5882 - val_loss: 1.0814 - val_categorical_accuracy: 0.5800 - lr:
2.0000e-04
Epoch 15/65
34/34 [==============================] - ETA: 0s - loss: 0.9713 - categorical_accuracy: 0.6235
Epoch 15: saving model to model_init_2022-09-1518_15_00.104107\model-00015-0.97131-0.62353-0.89293-0.73000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.9713 - categorical_accuracy: 0.6235 - val_loss: 0.8929 - val_categorical_accuracy: 0.7300 - lr:
2.0000e-04
Epoch 16/65
34/34 [==============================] - ETA: 0s - loss: 0.9567 - categorical_accuracy: 0.6162
Epoch 16: saving model to model_init_2022-09-1518_15_00.104107\model-00016-0.95672-0.61618-0.92126-0.69000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.9567 - categorical_accuracy: 0.6162 - val_loss: 0.9213 - val_categorical_accuracy: 0.6900 - lr:
2.0000e-04
Epoch 17/65
34/34 [==============================] - ETA: 0s - loss: 0.8863 - categorical_accuracy: 0.6382
Epoch 17: saving model to model_init_2022-09-1518_15_00.104107\model-00017-0.88632-0.63824-0.88426-0.69000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.8863 - categorical_accuracy: 0.6382 - val_loss: 0.8843 - val_categorical_accuracy: 0.6900 - lr:
2.0000e-04
Epoch 18/65
34/34 [==============================] - ETA: 0s - loss: 0.8366 - categorical_accuracy: 0.6926
Epoch 18: saving model to model_init_2022-09-1518_15_00.104107\model-00018-0.83656-0.69265-0.86907-0.69000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.8366 - categorical_accuracy: 0.6926 - val_loss: 0.8691 - val_categorical_accuracy: 0.6900 - lr:
2.0000e-04
Epoch 19/65
34/34 [==============================] - ETA: 0s - loss: 0.8007 - categorical_accuracy: 0.6838
Epoch 19: saving model to model_init_2022-09-1518_15_00.104107\model-00019-0.80069-0.68382-0.94206-0.63000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.8007 - categorical_accuracy: 0.6838 - val_loss: 0.9421 - val_categorical_accuracy: 0.6300 - lr:
2.0000e-04
Epoch 20/65
34/34 [==============================] - ETA: 0s - loss: 0.7903 - categorical_accuracy: 0.6985
Epoch 20: saving model to model_init_2022-09-1518_15_00.104107\model-00020-0.79032-0.69853-0.72702-0.77000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.7903 - categorical_accuracy: 0.6985 - val_loss: 0.7270 - val_categorical_accuracy: 0.7700 - lr:
2.0000e-04
Epoch 21/65
34/34 [==============================] - ETA: 0s - loss: 0.7823 - categorical_accuracy: 0.7221
Epoch 21: saving model to model_init_2022-09-1518_15_00.104107\model-00021-0.78228-0.72206-0.89267-0.66000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.7823 - categorical_accuracy: 0.7221 - val_loss: 0.8927 - val_categorical_accuracy: 0.6600 - lr:
2.0000e-04
Epoch 22/65
34/34 [==============================] - ETA: 0s - loss: 0.7667 - categorical_accuracy: 0.7074
Epoch 22: saving model to model_init_2022-09-1518_15_00.104107\model-00022-0.76667-0.70735-0.88463-0.73000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.7667 - categorical_accuracy: 0.7074 - val_loss: 0.8846 - val_categorical_accuracy: 0.7300 - lr:
2.0000e-04
Epoch 23/65
34/34 [==============================] - ETA: 0s - loss: 0.7356 - categorical_accuracy: 0.7147
Epoch 23: saving model to model_init_2022-09-1518_15_00.104107\model-00023-0.73559-0.71471-0.95279-0.63000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.7356 - categorical_accuracy: 0.7147 - val_loss: 0.9528 - val_categorical_accuracy: 0.6300 - lr:
2.0000e-04
Epoch 24/65
34/34 [==============================] - ETA: 0s - loss: 0.6934 - categorical_accuracy: 0.7397
Epoch 24: saving model to model_init_2022-09-1518_15_00.104107\model-00024-0.69339-0.73971-0.77393-0.73000.h5
] - 60s 2s/step - loss: 0.6934 - categorical_accuracy: 0.7397 - val_loss: 0.7739 - val_categorical_accuracy: 0.7300 - lr:
2.0000e-04
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
Epoch 25/65
34/34 [==============================] - ETA: 0s - loss: 0.6565 - categorical_accuracy: 0.7412
Epoch 25: saving model to model_init_2022-09-1518_15_00.104107\model-00025-0.65654-0.74118-0.73768-0.70000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.6565 - categorical_accuracy: 0.7412 - val_loss: 0.7377 - val_categorical_accuracy: 0.7000 - lr: 
2.0000e-04
Epoch 26/65
34/34 [==============================] - ETA: 0s - loss: 0.6134 - categorical_accuracy: 0.7721
Epoch 26: saving model to model_init_2022-09-1518_15_00.104107\model-00026-0.61338-0.77206-0.75010-0.74000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.6134 - categorical_accuracy: 0.7721 - val_loss: 0.7501 - val_categorical_accuracy: 0.7400 - lr: 
2.0000e-04
Epoch 27/65
34/34 [==============================] - ETA: 0s - loss: 0.6035 - categorical_accuracy: 0.7868
Epoch 27: saving model to model_init_2022-09-1518_15_00.104107\model-00027-0.60354-0.78676-0.68983-0.77000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.6035 - categorical_accuracy: 0.7868 - val_loss: 0.6898 - val_categorical_accuracy: 0.7700 - lr: 
2.0000e-04
Epoch 28/65
34/34 [==============================] - ETA: 0s - loss: 0.5463 - categorical_accuracy: 0.8015
Epoch 28: saving model to model_init_2022-09-1518_15_00.104107\model-00028-0.54625-0.80147-0.80661-0.75000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.5463 - categorical_accuracy: 0.8015 - val_loss: 0.8066 - val_categorical_accuracy: 0.7500 - lr: 
2.0000e-04
Epoch 29/65
34/34 [==============================] - ETA: 0s - loss: 0.5123 - categorical_accuracy: 0.7985
Epoch 29: saving model to model_init_2022-09-1518_15_00.104107\model-00029-0.51230-0.79853-0.80927-0.74000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.5123 - categorical_accuracy: 0.7985 - val_loss: 0.8093 - val_categorical_accuracy: 0.7400 - lr: 
2.0000e-04
Epoch 30/65
34/34 [==============================] - ETA: 0s - loss: 0.5288 - categorical_accuracy: 0.8029
Epoch 30: saving model to model_init_2022-09-1518_15_00.104107\model-00030-0.52880-0.80294-0.86395-0.73000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.5288 - categorical_accuracy: 0.8029 - val_loss: 0.8640 - val_categorical_accuracy: 0.7300 - lr: 
2.0000e-04
Epoch 31/65
34/34 [==============================] - ETA: 0s - loss: 0.5313 - categorical_accuracy: 0.8059
Epoch 31: saving model to model_init_2022-09-1518_15_00.104107\model-00031-0.53129-0.80588-0.76271-0.76000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.5313 - categorical_accuracy: 0.8059 - val_loss: 0.7627 - val_categorical_accuracy: 0.7600 - lr: 
2.0000e-04
Epoch 32/65
34/34 [==============================] - ETA: 0s - loss: 0.4802 - categorical_accuracy: 0.8441
Epoch 32: saving model to model_init_2022-09-1518_15_00.104107\model-00032-0.48023-0.84412-0.83953-0.70000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.4802 - categorical_accuracy: 0.8441 - val_loss: 0.8395 - val_categorical_accuracy: 0.7000 - lr: 
2.0000e-04
Epoch 33/65
34/34 [==============================] - ETA: 0s - loss: 0.4567 - categorical_accuracy: 0.8191
Epoch 33: saving model to model_init_2022-09-1518_15_00.104107\model-00033-0.45667-0.81912-0.75144-0.78000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.4567 - categorical_accuracy: 0.8191 - val_loss: 0.7514 - val_categorical_accuracy: 0.7800 - lr: 
2.0000e-04
Epoch 34/65
34/34 [==============================] - ETA: 0s - loss: 0.4268 - categorical_accuracy: 0.8426
Epoch 34: saving model to model_init_2022-09-1518_15_00.104107\model-00034-0.42676-0.84265-0.85273-0.73000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.4268 - categorical_accuracy: 0.8426 - val_loss: 0.8527 - val_categorical_accuracy: 0.7300 - lr: 
2.0000e-04
Epoch 35/65
34/34 [] - ETA: 0s - loss: 0.4281 - categorical_accuracy: 0.8441
Epoch 35: saving model to model_init_2022-09-1518_15_00.104107\model-00035-0.42811-0.84412-1.00103-0.69000.h5
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
34/34 [==============================] - 61s 2s/step - loss: 0.4281 - categorical_accuracy: 0.8441 - val_loss: 1.0010 - val_categorical_accuracy: 0.6900 - lr:
2.0000e-04
Epoch 36/65
34/34 [==============================] - ETA: 0s - loss: 0.3687 - categorical_accuracy: 0.8618
Epoch 36: saving model to model_init_2022-09-1518_15_00.104107\model-00036-0.36873-0.86176-0.77732-0.70000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.3687 - categorical_accuracy: 0.8618 - val_loss: 0.7773 - val_categorical_accuracy: 0.7000 - lr:
2.0000e-04
Epoch 37/65
34/34 [==============================] - ETA: 0s - loss: 0.4115 - categorical_accuracy: 0.8412
Epoch 37: saving model to model_init_2022-09-1518_15_00.104107\model-00037-0.41147-0.84118-0.93592-0.66000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.4115 - categorical_accuracy: 0.8412 - val_loss: 0.9359 - val_categorical_accuracy: 0.6600 - lr:
2.0000e-04
Epoch 38/65
34/34 [==============================] - ETA: 0s - loss: 0.3660 - categorical_accuracy: 0.8588
Epoch 38: saving model to model_init_2022-09-1518_15_00.104107\model-00038-0.36604-0.85882-0.76052-0.76000.h5
34/34 [==============================] - 65s 2s/step - loss: 0.3660 - categorical_accuracy: 0.8588 - val_loss: 0.7605 - val_categorical_accuracy: 0.7600 - lr:
1.0000e-04
Epoch 39/65
34/34 [==============================] - ETA: 0s - loss: 0.3431 - categorical_accuracy: 0.8750
Epoch 39: saving model to model_init_2022-09-1518_15_00.104107\model-00039-0.34306-0.87500-0.82742-0.69000.h5
34/34 [==============================] - 64s 2s/step - loss: 0.3431 - categorical_accuracy: 0.8750 - val_loss: 0.8274 - val_categorical_accuracy: 0.6900 - lr:
1.0000e-04
Epoch 40/65
34/34 [==============================] - ETA: 0s - loss: 0.3178 - categorical_accuracy: 0.8912
Epoch 40: saving model to model_init_2022-09-1518_15_00.104107\model-00040-0.31785-0.89118-0.71300-0.78000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.3178 - categorical_accuracy: 0.8912 - val_loss: 0.7130 - val_categorical_accuracy: 0.7800 - lr:
1.0000e-04
Epoch 41/65
34/34 [==============================] - ETA: 0s - loss: 0.3046 - categorical_accuracy: 0.8941
Epoch 41: saving model to model_init_2022-09-1518_15_00.104107\model-00041-0.30464-0.89412-0.76005-0.74000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.3046 - categorical_accuracy: 0.8941 - val_loss: 0.7601 - val_categorical_accuracy: 0.7400 - lr:
1.0000e-04
Epoch 42/65
34/34 [==============================] - ETA: 0s - loss: 0.2943 - categorical_accuracy: 0.8882
Epoch 42: saving model to model_init_2022-09-1518_15_00.104107\model-00042-0.29427-0.88824-0.62220-0.81000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2943 - categorical_accuracy: 0.8882 - val_loss: 0.6222 - val_categorical_accuracy: 0.8100 - lr:
1.0000e-04
Epoch 43/65
34/34 [==============================] - ETA: 0s - loss: 0.2957 - categorical_accuracy: 0.9000
Epoch 43: saving model to model_init_2022-09-1518_15_00.104107\model-00043-0.29574-0.90000-0.67629-0.83000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2957 - categorical_accuracy: 0.9000 - val_loss: 0.6763 - val_categorical_accuracy: 0.8300 - lr:
1.0000e-04
Epoch 44/65
34/34 [==============================] - ETA: 0s - loss: 0.2582 - categorical_accuracy: 0.9162
Epoch 44: saving model to model_init_2022-09-1518_15_00.104107\model-00044-0.25818-0.91618-0.90709-0.69000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2582 - categorical_accuracy: 0.9162 - val_loss: 0.9071 - val_categorical_accuracy: 0.6900 - lr:
1.0000e-04
Epoch 45/65
34/34 [==============================] - ETA: 0s - loss: 0.2656 - categorical_accuracy: 0.9088
Epoch 45: saving model to model_init_2022-09-1518_15_00.104107\model-00045-0.26558-0.90882-0.79542-0.76000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.2656 - categorical_accuracy: 0.9088 - val_loss: 0.7954 - val_categorical_accuracy: 0.7600 - lr:
1.0000e-04
Epoch 46/65
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
34/34 [==============================] - ETA: 0s - loss: 0.2769 - categorical_accuracy: 0.8971
Epoch 46: saving model to model_init_2022-09-1518_15_00.104107\model-00046-0.27694-0.89706-0.81495-0.72000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2769 - categorical_accuracy: 0.8971 - val_loss: 0.8149 - val_categorical_accuracy: 0.7200 - lr:
1.0000e-04
Epoch 47/65
34/34 [==============================] - ETA: 0s - loss: 0.2489 - categorical_accuracy: 0.9176
Epoch 47: saving model to model_init_2022-09-1518_15_00.104107\model-00047-0.24886-0.91765-0.70873-0.79000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.2489 - categorical_accuracy: 0.9176 - val_loss: 0.7087 - val_categorical_accuracy: 0.7900 - lr:
1.0000e-04
Epoch 48/65
34/34 [==============================] - ETA: 0s - loss: 0.2151 - categorical_accuracy: 0.9353
Epoch 48: saving model to model_init_2022-09-1518_15_00.104107\model-00048-0.21511-0.93529-0.77517-0.76000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2151 - categorical_accuracy: 0.9353 - val_loss: 0.7752 - val_categorical_accuracy: 0.7600 - lr:
1.0000e-04
Epoch 49/65
34/34 [==============================] - ETA: 0s - loss: 0.2340 - categorical_accuracy: 0.9176
Epoch 49: saving model to model_init_2022-09-1518_15_00.104107\model-00049-0.23402-0.91765-0.52271-0.86000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2340 - categorical_accuracy: 0.9176 - val_loss: 0.5227 - val_categorical_accuracy: 0.8600 - lr:
1.0000e-04
Epoch 50/65
34/34 [==============================] - ETA: 0s - loss: 0.2548 - categorical_accuracy: 0.9029
Epoch 50: saving model to model_init_2022-09-1518_15_00.104107\model-00050-0.25478-0.90294-0.89981-0.72000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2548 - categorical_accuracy: 0.9029 - val_loss: 0.8998 - val_categorical_accuracy: 0.7200 - lr:
1.0000e-04
Epoch 51/65
34/34 [==============================] - ETA: 0s - loss: 0.2351 - categorical_accuracy: 0.9132
Epoch 51: saving model to model_init_2022-09-1518_15_00.104107\model-00051-0.23512-0.91324-0.77630-0.75000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2351 - categorical_accuracy: 0.9132 - val_loss: 0.7763 - val_categorical_accuracy: 0.7500 - lr:
1.0000e-04
Epoch 52/65
34/34 [==============================] - ETA: 0s - loss: 0.2135 - categorical_accuracy: 0.9235
Epoch 52: saving model to model_init_2022-09-1518_15_00.104107\model-00052-0.21346-0.92353-0.69747-0.80000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.2135 - categorical_accuracy: 0.9235 - val_loss: 0.6975 - val_categorical_accuracy: 0.8000 - lr:
1.0000e-04
Epoch 53/65
34/34 [==============================] - ETA: 0s - loss: 0.2198 - categorical_accuracy: 0.9176
Epoch 53: saving model to model_init_2022-09-1518_15_00.104107\model-00053-0.21983-0.91765-0.66130-0.75000.h5
34/34 [==============================] - 61s 2s/step - loss: 0.2198 - categorical_accuracy: 0.9176 - val_loss: 0.6613 - val_categorical_accuracy: 0.7500 - lr:
1.0000e-04
Epoch 54/65
34/34 [==============================] - ETA: 0s - loss: 0.2005 - categorical_accuracy: 0.9324
Epoch 54: saving model to model_init_2022-09-1518_15_00.104107\model-00054-0.20046-0.93235-0.89407-0.75000.h5
34/34 [==============================] - 61s 2s/step - loss: 0.2005 - categorical_accuracy: 0.9324 - val_loss: 0.8941 - val_categorical_accuracy: 0.7500 - lr:
1.0000e-04
Epoch 55/65
34/34 [==============================] - ETA: 0s - loss: 0.1977 - categorical_accuracy: 0.9426
Epoch 55: saving model to model_init_2022-09-1518_15_00.104107\model-00055-0.19766-0.94265-0.76078-0.81000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.1977 - categorical_accuracy: 0.9426 - val_loss: 0.7608 - val_categorical_accuracy: 0.8100 - lr:
1.0000e-04
Epoch 56/65
34/34 [==============================] - ETA: 0s - loss: 0.2045 - categorical_accuracy: 0.9265
Epoch 56: saving model to model_init_2022-09-1518_15_00.104107\model-00056-0.20452-0.92647-0.77070-0.74000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.2045 - categorical_accuracy: 0.9265 - val_loss: 0.7707 - val_categorical_accuracy: 0.7400 - lr:
```

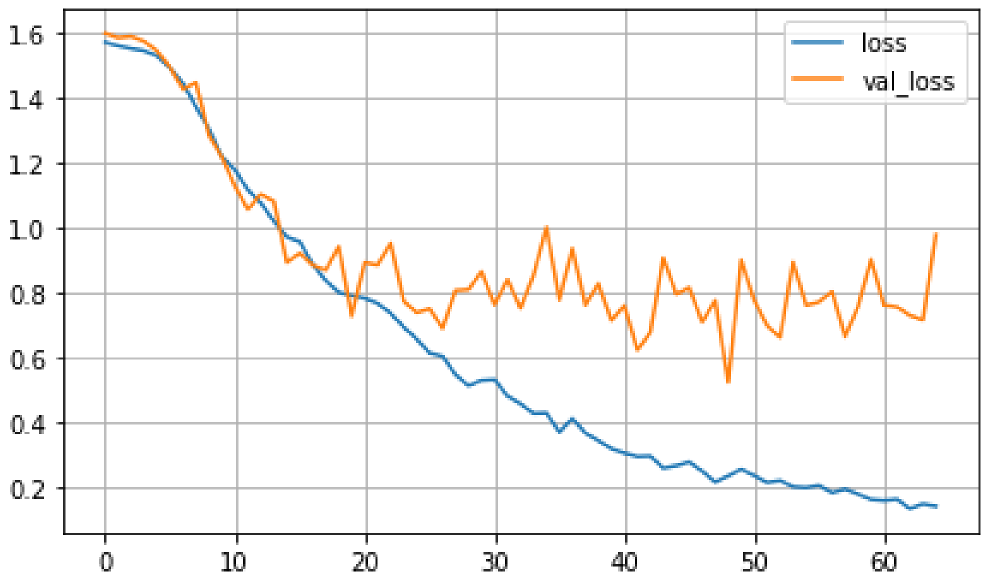
Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

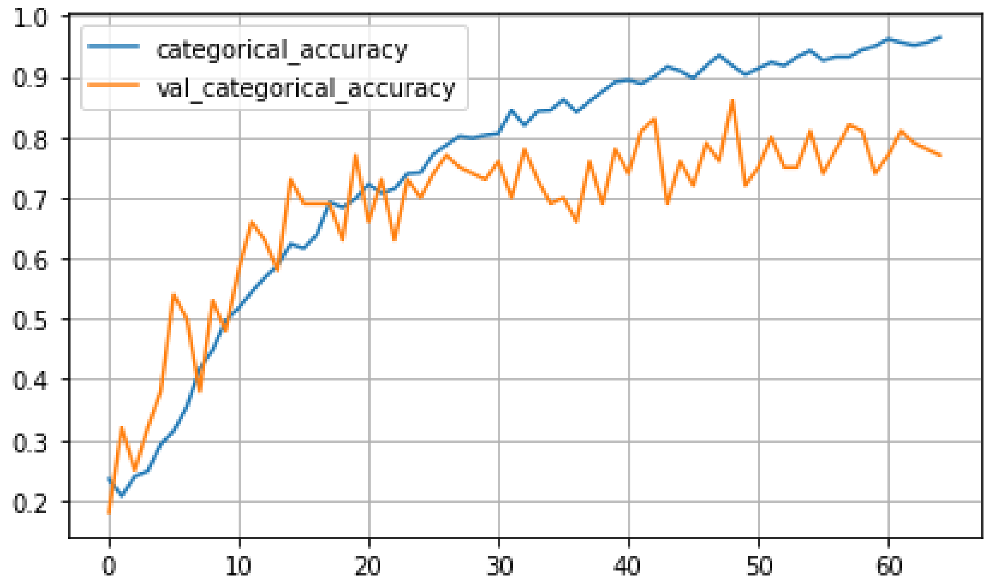
```
1.0000e-04
Epoch 57/65
34/34 [==============================] - ETA: 0s - loss: 0.1820 - categorical_accuracy: 0.9324
Epoch 57: saving model to model_init_2022-09-1518_15_00.104107\model-00057-0.18204-0.93235-0.80328-0.78000.h5
34/34 [==============================] - 62s 2s/step - loss: 0.1820 - categorical_accuracy: 0.9324 - val_loss: 0.8033 - val_categorical_accuracy: 0.7800 - lr:
1.0000e-04
Epoch 58/65
34/34 [==============================] - ETA: 0s - loss: 0.1933 - categorical_accuracy: 0.9324
Epoch 58: saving model to model_init_2022-09-1518_15_00.104107\model-00058-0.19335-0.93235-0.66319-0.82000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.1933 - categorical_accuracy: 0.9324 - val_loss: 0.6632 - val_categorical_accuracy: 0.8200 - lr:
1.0000e-04
Epoch 59/65
34/34 [==============================] - ETA: 0s - loss: 0.1778 - categorical_accuracy: 0.9441
Epoch 59: saving model to model_init_2022-09-1518_15_00.104107\model-00059-0.17778-0.94412-0.75621-0.81000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.1778 - categorical_accuracy: 0.9441 - val_loss: 0.7562 - val_categorical_accuracy: 0.8100 - lr:
1.0000e-04
Epoch 60/65
34/34 [==============================] - ETA: 0s - loss: 0.1611 - categorical_accuracy: 0.9500
Epoch 60: saving model to model_init_2022-09-1518_15_00.104107\model-00060-0.16113-0.95000-0.90170-0.74000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.1611 - categorical_accuracy: 0.9500 - val_loss: 0.9017 - val_categorical_accuracy: 0.7400 - lr:
5.0000e-05
Epoch 61/65
34/34 [==============================] - ETA: 0s - loss: 0.1579 - categorical_accuracy: 0.9618
Epoch 61: saving model to model_init_2022-09-1518_15_00.104107\model-00061-0.15789-0.96176-0.76134-0.77000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.1579 - categorical_accuracy: 0.9618 - val_loss: 0.7613 - val_categorical_accuracy: 0.7700 - lr:
5.0000e-05
Epoch 62/65
34/34 [==============================] - ETA: 0s - loss: 0.1617 - categorical_accuracy: 0.9559
Epoch 62: saving model to model_init_2022-09-1518_15_00.104107\model-00062-0.16173-0.95588-0.75560-0.81000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.1617 - categorical_accuracy: 0.9559 - val_loss: 0.7556 - val_categorical_accuracy: 0.8100 - lr:
5.0000e-05
Epoch 63/65
34/34 [==============================] - ETA: 0s - loss: 0.1333 - categorical_accuracy: 0.9515
Epoch 63: saving model to model_init_2022-09-1518_15_00.104107\model-00063-0.13329-0.95147-0.72948-0.79000.h5
34/34 [==============================] - 59s 2s/step - loss: 0.1333 - categorical_accuracy: 0.9515 - val_loss: 0.7295 - val_categorical_accuracy: 0.7900 - lr:
5.0000e-05
Epoch 64/65
34/34 [==============================] - ETA: 0s - loss: 0.1478 - categorical_accuracy: 0.9559
Epoch 64: saving model to model_init_2022-09-1518_15_00.104107\model-00064-0.14779-0.95588-0.71482-0.78000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.1478 - categorical_accuracy: 0.9559 - val_loss: 0.7148 - val_categorical_accuracy: 0.7800 - lr:
5.0000e-05
Epoch 65/65
34/34 [==============================] - ETA: 0s - loss: 0.1409 - categorical_accuracy: 0.9647
Epoch 65: saving model to model_init_2022-09-1518_15_00.104107\model-00065-0.14086-0.96471-0.97777-0.77000.h5
34/34 [==============================] - 60s 2s/step - loss: 0.1409 - categorical_accuracy: 0.9647 - val_loss: 0.9778 - val_categorical_accuracy: 0.7700 - lr:
5.0000e-05
```

In [11]:

```
plot_model(Model_ConvGRU)
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js





Inference :

- **The final model has total number of parameters : 65,221 which is lesser compared to other models that are tried.**
- **At 49th epoch, we obtained the following best results:**

> - **Training accuracy : 0.9176**
> - **Validation accuracy : 0.8600**
> - **Training loss : 0.2340**
> - **Validation loss : 0.5227**

- **The learning rate was 0.0002 till 37th epoch and it gets reduced to 0.0001 for the following epochs according to the parameters mentioned in the 'ReduceLROnPlateau'. Further, at 60th epoch, it still gets reduced to 0.00005.**

MODEL WITH TRANSFER LEARNING AND GRU

- **Transfer learning + GRU model**
- **With augmentation**
- **With dropout layers**
- **Activation function** : 'relu'
- **Optimizer** : 'Adam'
- **Image height** = 100
- **Image width** = 100
- **Batch size** = 20
- **Number of epochs** = 25
- **Metrics** : Accuracy
- **Here we incorporated imagenet weights from mobilenet data.**

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

In [12]:

```
# Instantiating the object for BuildModel() class

Model_transfer_GRU = BuildModel()

# Image height = 100
# Image width = 100
# Batch_size = 20
# Number of epochs = 25

Model_transfer_GRU.path_initialization(source_path)
Model_transfer_GRU.values_initialization(100,100,num_of_selected_frames,20,25)

from keras.applications import mobilenet
mobilenet_transfer = mobilenet.MobileNet(weights='imagenet', include_top=False)

# Define the transfer learning + GRU network model using function 'TransferLearning_GRU_Model_definition()'

Model_2 = Model_transfer_GRU.TransferLearning_GRU_Model_definition(mobilenet_transfer)
Model_2.summary()

# Pass the model to train the data and obtain the results using Model_Training() function

Model_transfer_learning = Model_transfer_GRU.Model_Training(Model_2,0.7,5,10,augment=True)
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not in [128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded as the
default.
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 time_distributed_11 (TimeDi  (None, 11, 3, 3, 1024)   3228864   
 stributed)                                                      
                                                                 
 time_distributed_12 (TimeDi  (None, 11, 3, 3, 1024)   4096      
 stributed)                                                      
                                                                 
 time_distributed_13 (TimeDi  (None, 11, 1, 1, 1024)   0         
 stributed)                                                      
                                                                 
 dropout_2 (Dropout)         (None, 11, 1, 1, 1024)    0         
                                                                 
 time_distributed_14 (TimeDi  (None, 11, 1024)         0         
 stributed)                                                      
                                                                 
 gru_1 (GRU)                 (None, 16)                50016     
                                                                 
 dropout_3 (Dropout)         (None, 16)                0         
                                                                 
 dense_3 (Dense)             (None, 64)                1088      
                                                                 
 dropout_4 (Dropout)         (None, 64)                0         
                                                                 
 dense_4 (Dense)             (None, 5)                 325       
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
=================================================================
Total params: 3,284,389
Trainable params: 3,260,453
Non-trainable params: 23,936
_________________________________________________________________
Source path =  C:/Users/gayth/Project_data/train ; batch size = 20
Epoch 1/25
34/34 [==============================] - ETA: 0s - loss: 1.6086 - categorical_accuracy: 0.2426Source path =  C:/Users/gayth/Project_data/val ; batch size = 20

Epoch 1: saving model to model_init_2022-09-1519_27_50.351973\model-00001-1.60864-0.24265-1.42522-0.41000.h5
34/34 [==============================] - 161s 5s/step - loss: 1.6086 - categorical_accuracy: 0.2426 - val_loss: 1.4252 - val_categorical_accuracy: 0.4100 - lr:
3.0000e-04
Epoch 2/25
34/34 [==============================] - ETA: 0s - loss: 1.2616 - categorical_accuracy: 0.4691
Epoch 2: saving model to model_init_2022-09-1519_27_50.351973\model-00002-1.26157-0.46912-1.30842-0.52000.h5
34/34 [==============================] - 160s 5s/step - loss: 1.2616 - categorical_accuracy: 0.4691 - val_loss: 1.3084 - val_categorical_accuracy: 0.5200 - lr:
3.0000e-04
Epoch 3/25
34/34 [==============================] - ETA: 0s - loss: 1.0020 - categorical_accuracy: 0.6529
Epoch 3: saving model to model_init_2022-09-1519_27_50.351973\model-00003-1.00199-0.65294-0.88766-0.82000.h5
34/34 [==============================] - 161s 5s/step - loss: 1.0020 - categorical_accuracy: 0.6529 - val_loss: 0.8877 - val_categorical_accuracy: 0.8200 - lr:
3.0000e-04
Epoch 4/25
34/34 [==============================] - ETA: 0s - loss: 0.8257 - categorical_accuracy: 0.7294
Epoch 4: saving model to model_init_2022-09-1519_27_50.351973\model-00004-0.82569-0.72941-0.87204-0.71000.h5
34/34 [==============================] - 162s 5s/step - loss: 0.8257 - categorical_accuracy: 0.7294 - val_loss: 0.8720 - val_categorical_accuracy: 0.7100 - lr:
3.0000e-04
Epoch 5/25
34/34 [==============================] - ETA: 0s - loss: 0.6494 - categorical_accuracy: 0.8059
Epoch 5: saving model to model_init_2022-09-1519_27_50.351973\model-00005-0.64937-0.80588-0.71989-0.80000.h5
34/34 [==============================] - 161s 5s/step - loss: 0.6494 - categorical_accuracy: 0.8059 - val_loss: 0.7199 - val_categorical_accuracy: 0.8000 - lr:
3.0000e-04
Epoch 6/25
34/34 [==============================] - ETA: 0s - loss: 0.5291 - categorical_accuracy: 0.8544
Epoch 6: saving model to model_init_2022-09-1519_27_50.351973\model-00006-0.52913-0.85441-0.67285-0.84000.h5
34/34 [==============================] - 162s 5s/step - loss: 0.5291 - categorical_accuracy: 0.8544 - val_loss: 0.6729 - val_categorical_accuracy: 0.8400 - lr:
3.0000e-04
Epoch 7/25
34/34 [==============================] - ETA: 0s - loss: 0.4061 - categorical_accuracy: 0.8809
Epoch 7: saving model to model_init_2022-09-1519_27_50.351973\model-00007-0.40610-0.88088-0.65363-0.82000.h5
34/34 [==============================] - 162s 5s/step - loss: 0.4061 - categorical_accuracy: 0.8809 - val_loss: 0.6536 - val_categorical_accuracy: 0.8200 - lr:
3.0000e-04
Epoch 8/25
34/34 [==============================] - ETA: 0s - loss: 0.3690 - categorical_accuracy: 0.9147
Epoch 8: saving model to model_init_2022-09-1519_27_50.351973\model-00008-0.36899-0.91471-0.45396-0.85000.h5
34/34 [==============================] - 166s 5s/step - loss: 0.3690 - categorical_accuracy: 0.9147 - val_loss: 0.4540 - val_categorical_accuracy: 0.8500 - lr:
3.0000e-04
Epoch 9/25
34/34 [==============================] - ETA: 0s - loss: 0.2934 - categorical_accuracy: 0.9309
Epoch 9: saving model to model_init_2022-09-1519_27_50.351973\model-00009-0.29338-0.93088-0.61264-0.86000.h5
34/34 [==============================] - 166s 5s/step - loss: 0.2934 - categorical_accuracy: 0.9309 - val_loss: 0.6126 - val_categorical_accuracy: 0.8600 - lr:
3.0000e-04
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

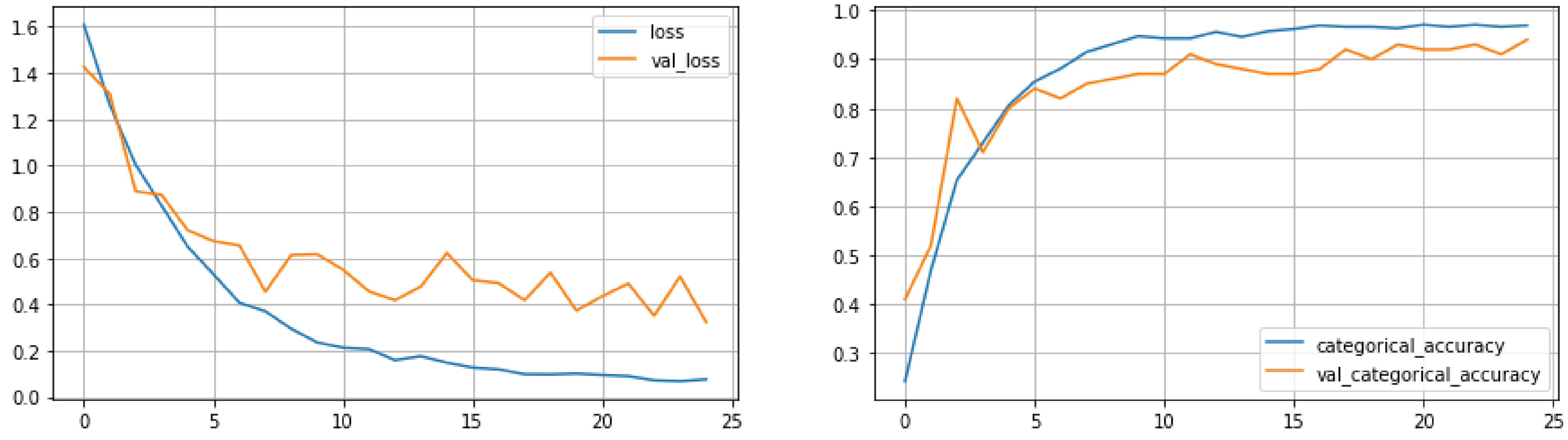
```
Epoch 10/25
34/34 [==============================] - ETA: 0s - loss: 0.2345 - categorical_accuracy: 0.9471
Epoch 10: saving model to model_init_2022-09-1519_27_50.351973\model-00010-0.23446-0.94706-0.61611-0.87000.h5
34/34 [==============================] - 171s 5s/step - loss: 0.2345 - categorical_accuracy: 0.9471 - val_loss: 0.6161 - val_categorical_accuracy: 0.8700 - lr:
3.0000e-04
Epoch 11/25
34/34 [==============================] - ETA: 0s - loss: 0.2118 - categorical_accuracy: 0.9426
Epoch 11: saving model to model_init_2022-09-1519_27_50.351973\model-00011-0.21178-0.94265-0.54848-0.87000.h5
34/34 [==============================] - 181s 5s/step - loss: 0.2118 - categorical_accuracy: 0.9426 - val_loss: 0.5485 - val_categorical_accuracy: 0.8700 - lr:
3.0000e-04
Epoch 12/25
34/34 [==============================] - ETA: 0s - loss: 0.2056 - categorical_accuracy: 0.9426
Epoch 12: saving model to model_init_2022-09-1519_27_50.351973\model-00012-0.20557-0.94265-0.45537-0.91000.h5
34/34 [==============================] - 175s 5s/step - loss: 0.2056 - categorical_accuracy: 0.9426 - val_loss: 0.4554 - val_categorical_accuracy: 0.9100 - lr:
3.0000e-04
Epoch 13/25
34/34 [==============================] - ETA: 0s - loss: 0.1579 - categorical_accuracy: 0.9559
Epoch 13: saving model to model_init_2022-09-1519_27_50.351973\model-00013-0.15789-0.95588-0.41692-0.89000.h5
34/34 [==============================] - 171s 5s/step - loss: 0.1579 - categorical_accuracy: 0.9559 - val_loss: 0.4169 - val_categorical_accuracy: 0.8900 - lr:
3.0000e-04
Epoch 14/25
34/34 [==============================] - ETA: 0s - loss: 0.1748 - categorical_accuracy: 0.9456
Epoch 14: saving model to model_init_2022-09-1519_27_50.351973\model-00014-0.17480-0.94559-0.47664-0.88000.h5
34/34 [==============================] - 174s 5s/step - loss: 0.1748 - categorical_accuracy: 0.9456 - val_loss: 0.4766 - val_categorical_accuracy: 0.8800 - lr:
3.0000e-04
Epoch 15/25
34/34 [==============================] - ETA: 0s - loss: 0.1469 - categorical_accuracy: 0.9574
Epoch 15: saving model to model_init_2022-09-1519_27_50.351973\model-00015-0.14694-0.95735-0.62097-0.87000.h5
34/34 [==============================] - 172s 5s/step - loss: 0.1469 - categorical_accuracy: 0.9574 - val_loss: 0.6210 - val_categorical_accuracy: 0.8700 - lr:
3.0000e-04
Epoch 16/25
34/34 [==============================] - ETA: 0s - loss: 0.1255 - categorical_accuracy: 0.9618
Epoch 16: saving model to model_init_2022-09-1519_27_50.351973\model-00016-0.12551-0.96176-0.50373-0.87000.h5
34/34 [==============================] - 171s 5s/step - loss: 0.1255 - categorical_accuracy: 0.9618 - val_loss: 0.5037 - val_categorical_accuracy: 0.8700 - lr:
3.0000e-04
Epoch 17/25
34/34 [==============================] - ETA: 0s - loss: 0.1181 - categorical_accuracy: 0.9691
Epoch 17: saving model to model_init_2022-09-1519_27_50.351973\model-00017-0.11810-0.96912-0.49122-0.88000.h5
34/34 [==============================] - 166s 5s/step - loss: 0.1181 - categorical_accuracy: 0.9691 - val_loss: 0.4912 - val_categorical_accuracy: 0.8800 - lr:
3.0000e-04
Epoch 18/25
34/34 [==============================] - ETA: 0s - loss: 0.0967 - categorical_accuracy: 0.9662
Epoch 18: saving model to model_init_2022-09-1519_27_50.351973\model-00018-0.09675-0.96618-0.41711-0.92000.h5
34/34 [==============================] - 175s 5s/step - loss: 0.0967 - categorical_accuracy: 0.9662 - val_loss: 0.4171 - val_categorical_accuracy: 0.9200 - lr:
3.0000e-04
Epoch 19/25
34/34 [==============================] - ETA: 0s - loss: 0.0960 - categorical_accuracy: 0.9662
Epoch 19: saving model to model_init_2022-09-1519_27_50.351973\model-00019-0.09600-0.96618-0.53633-0.90000.h5
34/34 [==============================] - 181s 5s/step - loss: 0.0960 - categorical_accuracy: 0.9662 - val_loss: 0.5363 - val_categorical_accuracy: 0.9000 - lr:
2.1000e-04
Epoch 20/25
34/34 [==============================] - ETA: 0s - loss: 0.0994 - categorical_accuracy: 0.9632
Epoch 20: saving model to model_init_2022-09-1519_27_50.351973\model-00020-0.09937-0.96324-0.37235-0.93000.h5
```

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

```
34/34 [==============================] - 176s 5s/step - loss: 0.0994 - categorical_accuracy: 0.9632 - val_loss: 0.3723 - val_categorical_accuracy: 0.9300 - lr:
2.1000e-04
Epoch 21/25
34/34 [==============================] - ETA: 0s - loss: 0.0934 - categorical_accuracy: 0.9706
Epoch 21: saving model to model_init_2022-09-1519_27_50.351973\model-00021-0.09338-0.97059-0.43427-0.92000.h5
34/34 [==============================] - 176s 5s/step - loss: 0.0934 - categorical_accuracy: 0.9706 - val_loss: 0.4343 - val_categorical_accuracy: 0.9200 - lr:
2.1000e-04
Epoch 22/25
34/34 [==============================] - ETA: 0s - loss: 0.0887 - categorical_accuracy: 0.9662
Epoch 22: saving model to model_init_2022-09-1519_27_50.351973\model-00022-0.08870-0.96618-0.48960-0.92000.h5
34/34 [==============================] - 183s 5s/step - loss: 0.0887 - categorical_accuracy: 0.9662 - val_loss: 0.4896 - val_categorical_accuracy: 0.9200 - lr:
2.1000e-04
Epoch 23/25
34/34 [==============================] - ETA: 0s - loss: 0.0711 - categorical_accuracy: 0.9706
Epoch 23: saving model to model_init_2022-09-1519_27_50.351973\model-00023-0.07105-0.97059-0.34939-0.93000.h5
34/34 [==============================] - 175s 5s/step - loss: 0.0711 - categorical_accuracy: 0.9706 - val_loss: 0.3494 - val_categorical_accuracy: 0.9300 - lr:
2.1000e-04
Epoch 24/25
34/34 [==============================] - ETA: 0s - loss: 0.0670 - categorical_accuracy: 0.9662
Epoch 24: saving model to model_init_2022-09-1519_27_50.351973\model-00024-0.06697-0.96618-0.52003-0.91000.h5
34/34 [==============================] - 180s 5s/step - loss: 0.0670 - categorical_accuracy: 0.9662 - val_loss: 0.5200 - val_categorical_accuracy: 0.9100 - lr:
2.1000e-04
Epoch 25/25
34/34 [==============================] - ETA: 0s - loss: 0.0753 - categorical_accuracy: 0.9691
Epoch 25: saving model to model_init_2022-09-1519_27_50.351973\model-00025-0.07527-0.96912-0.32305-0.94000.h5
34/34 [==============================] - 166s 5s/step - loss: 0.0753 - categorical_accuracy: 0.9691 - val_loss: 0.3231 - val_categorical_accuracy: 0.9400 - lr:
2.1000e-04
```

In [13]:

```
plot_model(Model_transfer_learning)
```



Inference :

- **As we can see the model with transfer learning performs the best with higher accuracy rate in both training and validation sets as well.**

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js

- **At 25th epoch, we obtained the following best results:**

  > - **Training accuracy : 0.9691**
  > - **Validation accuracy : 0.9400**
  > - **Training loss : 0.0753**
  > - **Validation loss : 0.3231**

- **The learning rate was 0.0003 till 18th epoch and it gets reduced to 0.0002 for the following epochs according to the parameters mentioned in the 'ReduceLROnPlateau'.**
- **Number of parameters involved in this model is too high and hence if at all the computational efficacies are tolerable, we can go with this model otherwise the previous model can be used which has lesser number of parameters.**

Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js