Sports image classification Project

[redacted]

The first model: (Simple CNN)

Pre-processing:

In this model First we read all data set and resize each image
to (50, 50 ,3) in function create_data() then we create for each image
their label using function create_label(image_name)

after we get our we split it to train data and test data

- 80% Train data
- 20% Test data

```
train_data, test_data = train_test_split(data, test_size=0.2, shuffle=False)
```

Then we get X and Y
For input data and there target:

```
train = train_data
test = test_data
X_train = np.array([i[0] for i in train])
y_train = [i[1] for i in train]

X_test = np.array([i[0] for i in test])
y_test = [i[1] for i in test]
```

The main model here is simple CNN. CNN is **a type of deep learning model for processing data that has a grid pattern, such as images**

- **CNN is a class of deep, feed-forward artificial neural networks, most commonly applied to analyzing visual imagery.**
- **CNNs, like neural networks, are made up of neurons with learnable weights and biases. Each neuron receives several inputs, takes a weighted sum over them, pass it through an activation function and responds with an output**

Architecture:

CNN typically consist of **convolutional layers**, **pooling layers**, **fully connected layer**

Here we have 5 convolution layers and after each convolution layers we have pooling layer and in the end we connected them in fully connected layer

- **1st convolution layer in size (32, 32, 3)**
- **2nd convolution layer in size (64, 64, 3)**
- **3rd convolution layer in size (128, 128, 3)**
- **4th convolution layer in size (32, 32, 3)**
- **5th convolution layer in size (64, 64, 3)**

With pooling layer after each convolution layer we used her activation function RELU between layers

The FC is the fully connected layer of neurons at the end of CNN. Neurons in a fully connected layer have full connections to all activations in the previous layer, as seen in regular Neural Networks and work in a similar way

```
conv_input = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 3], name='input')

conv1 = conv_2d(conv_input, 32, 3, activation='relu')
pool1 = max_pool_2d(conv1, 3)

conv2 = conv_2d(pool1, 64, 3, activation='relu')
pool2 = max_pool_2d(conv2, 3)

conv3 = conv_2d(pool2, 128, 3, activation='relu')
pool3 = max_pool_2d(conv3, 3)

conv4 = conv_2d(pool3, 32, 3, activation='relu')
pool4 = max_pool_2d(conv4, 3)

conv5 = conv_2d(pool4, 64, 3, activation='relu')
pool5 = max_pool_2d(conv5, 3)

fully_layer = fully_connected(pool5, 1024, activation='relu')
fully_layer = dropout(fully_layer, 0.5)

cnn_layers = fully_connected(fully_layer, 6, activation='softmax')

cnn_layers = regression(cnn_layers, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(cnn_layers, tensorboard_dir='log', tensorboard_verbose=7)
```

FIT The model:

Here we check if we have old trained data and weights to start with if not we start to train our model with train data and test data

Optimizer: Adam
Number of epochs : 45
Learning rate : 0.001
After that we save new updated weight

```
if (os.path.exists('model.tfl.meta')):
    model.load('./model.tfl')
else:
    model.fit({'input': X_train}, {'targets': y_train}, n_epoch=45,
                    validation_set=({'input': X_test}, {'targets': y_test}),
                    snapshot_step=200, show_metric=True, run_id=MODEL_NAME)
    model.save('model.tfl')
```

```
Training Step: 940  | total loss: 0.58342 | time: 0.793s
| Adam | epoch: 045 | loss: 0.58342 - acc: 0.9411 -- iter: 1024/1344
Training Step: 941  | total loss: 0.52658 | time: 0.837s
| Adam | epoch: 045 | loss: 0.52658 - acc: 0.9470 -- iter: 1088/1344
Training Step: 942  | total loss: 0.47510 | time: 0.887s
| Adam | epoch: 045 | loss: 0.47510 - acc: 0.9523 -- iter: 1152/1344
Training Step: 943  | total loss: 0.42856 | time: 0.940s
| Adam | epoch: 045 | loss: 0.42856 - acc: 0.9571 -- iter: 1216/1344
Training Step: 944  | total loss: 0.38808 | time: 0.986s
| Adam | epoch: 045 | loss: 0.38808 - acc: 0.9614 -- iter: 1280/1344
Training Step: 945  | total loss: 0.35012 | time: 2.041s
| Adam | epoch: 045 | loss: 0.35012 - acc: 0.9652 | val_loss: 0.72523 - val_acc: 0.7982 -- iter: 1344/1344
```

Accuracy :

96% train accuracy
80% test accuracy

The second model: (ResNet50)

Pre-processing:

1-splitting data into 6 classes
2-Convert dataset into batches
3-Resize each image to (225,225,3)

---

it’s a deep-learning neural network that is used as a backbone for many computer vision like object detection, and image segmentation, residual networks aren’t a new type of CNN models it’s just an update of the simple CNN to get more efficiency and speed

Architecture:

1-Identity block: The identity block is the standard block used in ResNet and corresponds to the case where the input activation has the same dimension as the output activation.

```
def identity_block(X, f, filters, stage, block):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'
    F1, F2, F3 = filters

    X_shortcut = X

    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2a', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X = Add()([X, X_shortcut])# SKIP Connection
    X = Activation('relu')(X)

    return X
```

2-Convolution block: We can use this type of block when the input and output dimensions don’t match up. The difference with the identity block is that there is a CONV2D layer in the shortcut path.

```
def convolutional_block(X, f, filters, stage, block, s=2):

    conv_name_base = 'res' + str(stage) + block + '_branch'
    bn_name_base = 'bn' + str(stage) + block + '_branch'

    F1, F2, F3 = filters

    X_shortcut = X

    X = Conv2D(filters=F1, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '2a', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2a')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F2, kernel_size=(f, f), strides=(1, 1), padding='same', name=conv_name_base + '2b', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2b')(X)
    X = Activation('relu')(X)

    X = Conv2D(filters=F3, kernel_size=(1, 1), strides=(1, 1), padding='valid', name=conv_name_base + '2c', kernel_initializer=glorot_uniform(seed=0))(X)
    X = BatchNormalization(axis=3, name=bn_name_base + '2c')(X)

    X_shortcut = Conv2D(filters=F3, kernel_size=(1, 1), strides=(s, s), padding='valid', name=conv_name_base + '1', kernel_initializer=glorot_uniform(seed=0))
    X_shortcut = BatchNormalization(axis=3, name=bn_name_base + '1')(X_shortcut)

    X = Add()([X, X_shortcut])
    X = Activation('relu')(X)

    return X
```

The full implementation of the model:

```
X_input = Input(input_shape)

X = ZeroPadding2D((3, 3))(X_input)
print(X.shape)
X = Conv2D(64, (7, 7), strides=(2, 2), name='conv1', kernel_initializer=glorot_uniform(seed=0))(X)
X = BatchNormalization(axis=3, name='bn_conv1')(X)
X = Activation('relu')(X)
X = MaxPooling2D((3, 3), strides=(2, 2))(X)

X = convolutional_block(X, f=3, filters=[64, 64, 256], stage=2, block='a', s=1)
X = identity_block(X, 3, [64, 64, 256], stage=2, block='b')
X = identity_block(X, 3, [64, 64, 256], stage=2, block='c')


X = convolutional_block(X, f=3, filters=[128, 128, 512], stage=3, block='a', s=2)
X = identity_block(X, 3, [128, 128, 512], stage=3, block='b')
X = identity_block(X, 3, [128, 128, 512], stage=3, block='c')
X = identity_block(X, 3, [128, 128, 512], stage=3, block='d')

X = convolutional_block(X, f=3, filters=[256, 256, 1024], stage=4, block='a', s=2)
X = identity_block(X, 3, [256, 256, 1024], stage=4, block='b')
X = identity_block(X, 3, [256, 256, 1024], stage=4, block='c')
X = identity_block(X, 3, [256, 256, 1024], stage=4, block='d')
X = identity_block(X, 3, [256, 256, 1024], stage=4, block='e')
X = identity_block(X, 3, [256, 256, 1024], stage=4, block='f')

X = X = convolutional_block(X, f=3, filters=[512, 512, 2048], stage=5, block='a', s=2)
X = identity_block(X, 3, [512, 512, 2048], stage=5, block='b')
X = identity_block(X, 3, [512, 512, 2048], stage=5, block='c')

X = AveragePooling2D(pool_size=(2, 2), padding='same')(X)

model = Model(inputs=X_input, outputs=X, name='ResNet50')
```

Training summary:

```
Model: "model"
```

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	[(None, 150, 150, 3)	0	
zero_padding2d (ZeroPadding2D)	(None, 156, 156, 3)	0	input_1[0][0]
conv1 (Conv2D)	(None, 75, 75, 64)	9472	zero_padding2d[0][0]
bn_conv1 (BatchNormalization)	(None, 75, 75, 64)	256	conv1[0][0]
activation (Activation)	(None, 75, 75, 64)	0	bn_conv1[0][0]
max_pooling2d (MaxPooling2D)	(None, 37, 37, 64)	0	activation[0][0]
res2a_branch2a (Conv2D)	(None, 37, 37, 64)	4160	max_pooling2d[0][0]
bn2a_branch2a (BatchNormalizati	(None, 37, 37, 64)	256	res2a_branch2a[0][0]
activation_1 (Activation)	(None, 37, 37, 64)	0	bn2a_branch2a[0][0]
res2a_branch2b (Conv2D)	(None, 37, 37, 64)	36928	activation_1[0][0]
bn2a_branch2b (BatchNormalizati	(None, 37, 37, 64)	256	res2a_branch2b[0][0]
activation_2 (Activation)	(None, 37, 37, 64)	0	bn2a_branch2b[0][0]
res2a_branch2c (Conv2D)	(None, 37, 37, 256)	16640	activation_2[0][0]
res2a_branch1 (Conv2D)	(None, 37, 37, 256)	16640	max_pooling2d[0][0]
bn2a_branch2c (BatchNormalizati	(None, 37, 37, 256)	1024	res2a_branch2c[0][0]
bn2a_branch1 (BatchNormalizatio	(None, 37, 37, 256)	1024	res2a_branch1[0][0]
add (Add)	(None, 37, 37, 256)	0	bn2a_branch2c[0][0] bn2a_branch1[0][0]
activation_3 (Activation)	(None, 37, 37, 256)	0	add[0][0]
res2b_branch2a (Conv2D)	(None, 37, 37, 64)	16448	activation_3[0][0]
bn2b_branch2a (BatchNormalizati	(None, 37, 37, 64)	256	res2b_branch2a[0][0]
activation_4 (Activation)	(None, 37, 37, 64)	0	bn2b_branch2a[0][0]
res2b_branch2b (Conv2D)	(None, 37, 37, 64)	36928	activation_4[0][0]
bn2b_branch2b (BatchNormalizati	(None, 37, 37, 64)	256	res2b_branch2b[0][0]
activation_5 (Activation)	(None, 37, 37, 64)	0	bn2b_branch2b[0][0]
res2b_branch2c (Conv2D)	(None, 37, 37, 256)	16640	activation_5[0][0]

bn2b_branch2c (BatchNormalizati	(None, 37, 37, 256)	1024	res2b_branch2c[0][0]
add_1 (Add)	(None, 37, 37, 256)	0	bn2b_branch2c[0][0] activation_3[0][0]
activation_6 (Activation)	(None, 37, 37, 256)	0	add_1[0][0]
res2c_branch2a (Conv2D)	(None, 37, 37, 64)	16448	activation_6[0][0]
bn2c_branch2a (BatchNormalizati	(None, 37, 37, 64)	256	res2c_branch2a[0][0]
activation_7 (Activation)	(None, 37, 37, 64)	0	bn2c_branch2a[0][0]
res2c_branch2b (Conv2D)	(None, 37, 37, 64)	36928	activation_7[0][0]
bn2c_branch2b (BatchNormalizati	(None, 37, 37, 64)	256	res2c_branch2b[0][0]
activation_8 (Activation)	(None, 37, 37, 64)	0	bn2c_branch2b[0][0]
res2c_branch2c (Conv2D)	(None, 37, 37, 256)	16640	activation_8[0][0]
bn2c_branch2c (BatchNormalizati	(None, 37, 37, 256)	1024	res2c_branch2c[0][0]
add_2 (Add)	(None, 37, 37, 256)	0	bn2c_branch2c[0][0] activation_6[0][0]
activation_9 (Activation)	(None, 37, 37, 256)	0	add_2[0][0]
res3a_branch2a (Conv2D)	(None, 19, 19, 128)	32896	activation_9[0][0]
bn3a_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3a_branch2a[0][0]
activation_10 (Activation)	(None, 19, 19, 128)	0	bn3a_branch2a[0][0]
res3a_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_10[0][0]
bn3a_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3a_branch2b[0][0]
activation_11 (Activation)	(None, 19, 19, 128)	0	bn3a_branch2b[0][0]
res3a_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_11[0][0]
res3a_branch1 (Conv2D)	(None, 19, 19, 512)	131584	activation_9[0][0]
bn3a_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3a_branch2c[0][0]
bn3a_branch1 (BatchNormalizatio	(None, 19, 19, 512)	2048	res3a_branch1[0][0]
add_3 (Add)	(None, 19, 19, 512)	0	bn3a_branch2c[0][0] bn3a_branch1[0][0]
activation_12 (Activation)	(None, 19, 19, 512)	0	add_3[0][0]
res3b_branch2a (Conv2D)	(None, 19, 19, 128)	65664	activation_12[0][0]
bn3b_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3b_branch2a[0][0]
activation_13 (Activation)	(None, 19, 19, 128)	0	bn3b_branch2a[0][0]
res3b_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_13[0][0]
bn3b_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3b_branch2b[0][0]
activation_14 (Activation)	(None, 19, 19, 128)	0	bn3b_branch2b[0][0]
res3b_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_14[0][0]
bn3b_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3b_branch2c[0][0]
add_4 (Add)	(None, 19, 19, 512)	0	bn3b_branch2c[0][0] activation_12[0][0]
activation_15 (Activation)	(None, 19, 19, 512)	0	add_4[0][0]
res3c_branch2a (Conv2D)	(None, 19, 19, 128)	65664	activation_15[0][0]
bn3c_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3c_branch2a[0][0]
activation_16 (Activation)	(None, 19, 19, 128)	0	bn3c_branch2a[0][0]
res3c_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_16[0][0]
bn3c_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3c_branch2b[0][0]
activation_17 (Activation)	(None, 19, 19, 128)	0	bn3c_branch2b[0][0]
res3c_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_17[0][0]
bn3c_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3c_branch2c[0][0]
add_5 (Add)	(None, 19, 19, 512)	0	bn3c_branch2c[0][0] activation_15[0][0]
activation_18 (Activation)	(None, 19, 19, 512)	0	add_5[0][0]
res3d_branch2a (Conv2D)	(None, 19, 19, 128)	65664	activation_18[0][0]
bn3d_branch2a (BatchNormalizati	(None, 19, 19, 128)	512	res3d_branch2a[0][0]
activation_19 (Activation)	(None, 19, 19, 128)	0	bn3d_branch2a[0][0]
res3d_branch2b (Conv2D)	(None, 19, 19, 128)	147584	activation_19[0][0]
bn3d_branch2b (BatchNormalizati	(None, 19, 19, 128)	512	res3d_branch2b[0][0]
activation_20 (Activation)	(None, 19, 19, 128)	0	bn3d_branch2b[0][0]
res3d_branch2c (Conv2D)	(None, 19, 19, 512)	66048	activation_20[0][0]

bn3d_branch2c (BatchNormalizati	(None, 19, 19, 512)	2048	res3d_branch2c[0][0]
add_6 (Add)	(None, 19, 19, 512)	0	bn3d_branch2c[0][0] activation_18[0][0]
activation_21 (Activation)	(None, 19, 19, 512)	0	add_6[0][0]
res4a_branch2a (Conv2D)	(None, 10, 10, 256)	131328	activation_21[0][0]
bn4a_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4a_branch2a[0][0]
activation_22 (Activation)	(None, 10, 10, 256)	0	bn4a_branch2a[0][0]
res4a_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_22[0][0]
bn4a_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4a_branch2b[0][0]
activation_23 (Activation)	(None, 10, 10, 256)	0	bn4a_branch2b[0][0]
res4a_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_23[0][0]
res4a_branch1 (Conv2D)	(None, 10, 10, 1024)	525312	activation_21[0][0]
bn4a_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4a_branch2c[0][0]
bn4a_branch1 (BatchNormalizatio	(None, 10, 10, 1024)	4096	res4a_branch1[0][0]
add_7 (Add)	(None, 10, 10, 1024)	0	bn4a_branch2c[0][0] bn4a_branch1[0][0]
activation_24 (Activation)	(None, 10, 10, 1024)	0	add_7[0][0]
res4b_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_24[0][0]
bn4b_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4b_branch2a[0][0]
activation_25 (Activation)	(None, 10, 10, 256)	0	bn4b_branch2a[0][0]
res4b_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_25[0][0]
bn4b_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4b_branch2b[0][0]
activation_26 (Activation)	(None, 10, 10, 256)	0	bn4b_branch2b[0][0]
res4b_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_26[0][0]
bn4b_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4b_branch2c[0][0]
add_8 (Add)	(None, 10, 10, 1024)	0	bn4b_branch2c[0][0] activation_24[0][0]
activation_27 (Activation)	(None, 10, 10, 1024)	0	add_8[0][0]
res4c_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_27[0][0]
bn4c_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4c_branch2a[0][0]
activation_28 (Activation)	(None, 10, 10, 256)	0	bn4c_branch2a[0][0]
res4c_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_28[0][0]
bn4c_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4c_branch2b[0][0]
activation_29 (Activation)	(None, 10, 10, 256)	0	bn4c_branch2b[0][0]
res4c_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_29[0][0]
bn4c_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4c_branch2c[0][0]
add_9 (Add)	(None, 10, 10, 1024)	0	bn4c_branch2c[0][0] activation_27[0][0]
activation_30 (Activation)	(None, 10, 10, 1024)	0	add_9[0][0]
res4d_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_30[0][0]
bn4d_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4d_branch2a[0][0]
activation_31 (Activation)	(None, 10, 10, 256)	0	bn4d_branch2a[0][0]
res4d_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_31[0][0]
bn4d_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4d_branch2b[0][0]
activation_32 (Activation)	(None, 10, 10, 256)	0	bn4d_branch2b[0][0]
res4d_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_32[0][0]
bn4d_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4d_branch2c[0][0]
add_10 (Add)	(None, 10, 10, 1024)	0	bn4d_branch2c[0][0] activation_30[0][0]
activation_33 (Activation)	(None, 10, 10, 1024)	0	add_10[0][0]
res4e_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_33[0][0]
bn4e_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4e_branch2a[0][0]
activation_34 (Activation)	(None, 10, 10, 256)	0	bn4e_branch2a[0][0]
res4e_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_34[0][0]
bn4e_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4e_branch2b[0][0]
activation_35 (Activation)	(None, 10, 10, 256)	0	bn4e_branch2b[0][0]
res4e_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_35[0][0]

bn4e_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4e_branch2c[0][0]
add_11 (Add)	(None, 10, 10, 1024)	0	bn4e_branch2c[0][0] activation_33[0][0]
activation_36 (Activation)	(None, 10, 10, 1024)	0	add_11[0][0]
res4f_branch2a (Conv2D)	(None, 10, 10, 256)	262400	activation_36[0][0]
bn4f_branch2a (BatchNormalizati	(None, 10, 10, 256)	1024	res4f_branch2a[0][0]
activation_37 (Activation)	(None, 10, 10, 256)	0	bn4f_branch2a[0][0]
res4f_branch2b (Conv2D)	(None, 10, 10, 256)	590080	activation_37[0][0]
bn4f_branch2b (BatchNormalizati	(None, 10, 10, 256)	1024	res4f_branch2b[0][0]
activation_38 (Activation)	(None, 10, 10, 256)	0	bn4f_branch2b[0][0]
res4f_branch2c (Conv2D)	(None, 10, 10, 1024)	263168	activation_38[0][0]
bn4f_branch2c (BatchNormalizati	(None, 10, 10, 1024)	4096	res4f_branch2c[0][0]
add_12 (Add)	(None, 10, 10, 1024)	0	bn4f_branch2c[0][0] activation_36[0][0]
activation_39 (Activation)	(None, 10, 10, 1024)	0	add_12[0][0]
res5a_branch2a (Conv2D)	(None, 5, 5, 512)	524800	activation_39[0][0]
bn5a_branch2a (BatchNormalizati	(None, 5, 5, 512)	2048	res5a_branch2a[0][0]
activation_40 (Activation)	(None, 5, 5, 512)	0	bn5a_branch2a[0][0]
res5a_branch2b (Conv2D)	(None, 5, 5, 512)	2359808	activation_40[0][0]
bn5a_branch2b (BatchNormalizati	(None, 5, 5, 512)	2048	res5a_branch2b[0][0]
activation_41 (Activation)	(None, 5, 5, 512)	0	bn5a_branch2b[0][0]
res5a_branch2c (Conv2D)	(None, 5, 5, 2048)	1050624	activation_41[0][0]
res5a_branch1 (Conv2D)	(None, 5, 5, 2048)	2099200	activation_39[0][0]
bn5a_branch2c (BatchNormalizati	(None, 5, 5, 2048)	8192	res5a_branch2c[0][0]
bn5a_branch1 (BatchNormalizatio	(None, 5, 5, 2048)	8192	res5a_branch1[0][0]
add_13 (Add)	(None, 5, 5, 2048)	0	bn5a_branch2c[0][0] bn5a_branch1[0][0]
activation_42 (Activation)	(None, 5, 5, 2048)	0	add_13[0][0]
res5b_branch2a (Conv2D)	(None, 5, 5, 512)	1049088	activation_42[0][0]
bn5b_branch2a (BatchNormalizati	(None, 5, 5, 512)	2048	res5b_branch2a[0][0]
activation_43 (Activation)	(None, 5, 5, 512)	0	bn5b_branch2a[0][0]
res5b_branch2b (Conv2D)	(None, 5, 5, 512)	2359808	activation_43[0][0]
bn5b_branch2b (BatchNormalizati	(None, 5, 5, 512)	2048	res5b_branch2b[0][0]
activation_44 (Activation)	(None, 5, 5, 512)	0	bn5b_branch2b[0][0]
res5b_branch2c (Conv2D)	(None, 5, 5, 2048)	1050624	activation_44[0][0]
bn5b_branch2c (BatchNormalizati	(None, 5, 5, 2048)	8192	res5b_branch2c[0][0]
add_14 (Add)	(None, 5, 5, 2048)	0	bn5b_branch2c[0][0] activation_42[0][0]
activation_45 (Activation)	(None, 5, 5, 2048)	0	add_14[0][0]
res5c_branch2a (Conv2D)	(None, 5, 5, 512)	1049088	activation_45[0][0]
bn5c_branch2a (BatchNormalizati	(None, 5, 5, 512)	2048	res5c_branch2a[0][0]
activation_46 (Activation)	(None, 5, 5, 512)	0	bn5c_branch2a[0][0]
res5c_branch2b (Conv2D)	(None, 5, 5, 512)	2359808	activation_46[0][0]
bn5c_branch2b (BatchNormalizati	(None, 5, 5, 512)	2048	res5c_branch2b[0][0]
activation_47 (Activation)	(None, 5, 5, 512)	0	bn5c_branch2b[0][0]
res5c_branch2c (Conv2D)	(None, 5, 5, 2048)	1050624	activation_47[0][0]
bn5c_branch2c (BatchNormalizati	(None, 5, 5, 2048)	8192	res5c_branch2c[0][0]
add_15 (Add)	(None, 5, 5, 2048)	0	bn5c_branch2c[0][0] activation_45[0][0]
activation_48 (Activation)	(None, 5, 5, 2048)	0	add_15[0][0]
average_pooling2d (AveragePooli	(None, 3, 3, 2048)	0	activation_48[0][0]
flatten (Flatten)	(None, 18432)	0	average_pooling2d[0][0]
fc1 (Dense)	(None, 256)	4718848	flatten[0][0]
fc2 (Dense)	(None, 128)	32896	fc1[0][0]
fc3 (Dense)	(None, 6)	774	fc2[0][0]

```
Total params: 28,340,230
Trainable params: 28,287,110
Non-trainable params: 53,120
```

Model training:

- **Using Adam optimizer with a learning rate of 0.001 and categorical cross-entropy loss function with 70 epochs**
- **training accuracy= 99% and validation accuracy = 92%**
- **Testing accuracy=85%**