AIML Capstone Project: CV - Car Detection

Final Project Report

Arun Sharma (Mentor)
Aravindan T
Jimmy R Fernandes
Kunal Pandya
Padmapriya Gopalakrishnan
Rajeshkanna Ala

Table of Contents

Summary and Problem Statement

Computer vision can be used to automate supervision and generate action appropriate action trigger if the event is predicted from the image of interest. For example a car moving on the road can be easily identified by a camera as make of the car, type, colour, number plates etc.

DATA DESCRIPTION*:*

The dataset contains 16,185 images of 196 classes of cars. The data is split into 8,144 training images and 8,041 testing images, where each class has been split roughly in a 50-50 split.

Classes are typically at the level of Make, Model, Year, e.g. 2012 Tesla Model S or 2012 BMW M3 coupe.

- Train Images: Consists of real images of cars as per the make and year of the car
- Test Images: Consists of real images of cars as per the make and year of the car.
- Train Annotation: Consists of bounding box region for training images.
- Test Annotation: Consists of bounding box region for testing images.

Dataset: https://drive.google.com/drive/folders/1y6JWx2CpsOuka00uePe72jNgr7F9sK45?usp=sharing,

PROJECT OBJECTIVE: Design a DL based car identification model

EDA and Pre-processing

Step #1: **Importing the dataset in dataframes**

```
#Different car labels
car_names = pd.read_csv( 'Car names and make.csv', header=None, names = ['CarLabe
l'] )

#Train data
train_data = pd.read_csv( 'Annotations/Train Annotations.csv', skiprows=1, names
= ['ImageName', 'X1', 'Y1', 'X2' , 'Y2', 'Class'] )

#Test data
test_data = pd.read_csv( 'Annotations/Test Annotation.csv' , skiprows=1, names =
['ImageName', 'X1', 'Y1', 'X2' , 'Y2', 'Class'])
```

Display few records

```
                     CarLabel
0  AM General Hummer SUV 2000
1         Acura RL Sedan 2012
2         Acura TL Sedan 2012
3        Acura TL Type-S 2008
4        Acura TSX Sedan 2012

   ImageName   X1   Y1    X2    Y2  Class
0  00001.jpg   39  116   569   375     14
1  00002.jpg   36  116   868   587      3
2  00003.jpg   85  109   601   381     91
3  00004.jpg  621  393  1484  1096    134
4  00005.jpg   14   36   133    99    106

   ImageName   X1   Y1   X2   Y2  Class
```

```
0   00001.jpg   30   52  246  147    181
1   00002.jpg  100   19  576  203    103
2   00003.jpg   51  105  968  659    145
3   00004.jpg   67   84  581  407    187

4   00005.jpg  140  151  593  339    185
```

Problem faced: While trying to read few images using cv2, we were getting return type as None. After troubleshooting, we found that this is due to folder name of images. Some of folder names have a '/' in it. Therefore, we decided to update such names as part of pre-processing step.

Step #2: **Find class name with '/' and update**

```
for i in range(len(car_names)) :
  if '/' in car_names.loc[i,"CarLabel"]:
    print(car_names.loc[i,"CarLabel"])
    print(i)
```

```
Ram C/V Cargo Van Minivan 2012

173
```

Thus, there was only 1 class with '/' in it's name.

```
#Replace '/' with '-' in the name
car_names.loc[173,'CarLabel'] = 'Ram C-V Cargo Van Minivan 2012'
```

Step #3: Map training and test images to corresponding classes and annotations.

```
car_names['Class'] = car_names.index + 1
car_train_df = pd.merge(train_data, car_names, how = 'left', left_on='Class', rig
ht_on='Class' )
car_train_df.head()
```

```
car_test_df = pd.merge(test_data, car_names, how = 'left', left_on='Class', right
_on='Class' )
car_test_df.head()
```

Display few records

	ImageName	X1	Y1	X2	Y2	Class	CarLabel
0	00001.jpg	30	52	246	147	181	Suzuki Aerio Sedan 2007
1	00002.jpg	100	19	576	203	103	Ferrari 458 Italia Convertible 2012
2	00003.jpg	51	105	968	659	145	Jeep Patriot SUV 2012
3	00004.jpg	67	84	581	407	187	Toyota Camry Sedan 2012
4	00005.jpg	140	151	593	339	185	Tesla Model S Sedan 2012

Step #4: **Exploratory Data Analysis**

- **For each car image label, separate year, make, model and body**

```
import nltk
nltk.download('punkt')

#Different body types
car_body_type = ["suv","sedan","type-s","type-
r","convertible","coupe","wagon","hatchback","cab","supercab","van","minivan"]
car_body_type = [item.lower() for item in car_body_type]

#Different car make
car_make = ['am', 'general', 'acura', 'aston', 'martin', 'audi', 'bmw', 'bentley',
'bugatti', 'buick', 'cadillac', 'chevrolet', 'chrysler', 'daewoo',
            'dodge', 'eagle', 'fiat', 'ferrari', 'fisker', 'ford', 'gmc', 'geo', 'h
onda', 'hyundai', 'infiniti', 'isuzu',
            'jaguar', 'jeep', 'lamborghini', 'land', 'rover', 'lincoln', 'mini', 'c
ooper', 'maybach', 'mazda', 'mclaren', 'mercedes-benz',
            'mitsubishi', 'nissan', 'plymouth', 'porsche', 'ram', 'rolls-
royce', 'scion', 'spyker', 'suzuki', 'tesla', 'toyota', 'volkswagen', 'volvo', 'sma
rt']
car_make = [item.lower() for item in car_make]

#Define dataframe to store results
eda_df = car_test_df.copy()
eda_df["year"] = None
eda_df["make"] = None
eda_df["model"] = None
eda_df["body"] = None

pattern="[0-9][0-9][0-9][0-9]"
for col in eda_df.columns:
    if col == 'CarLabel':
```

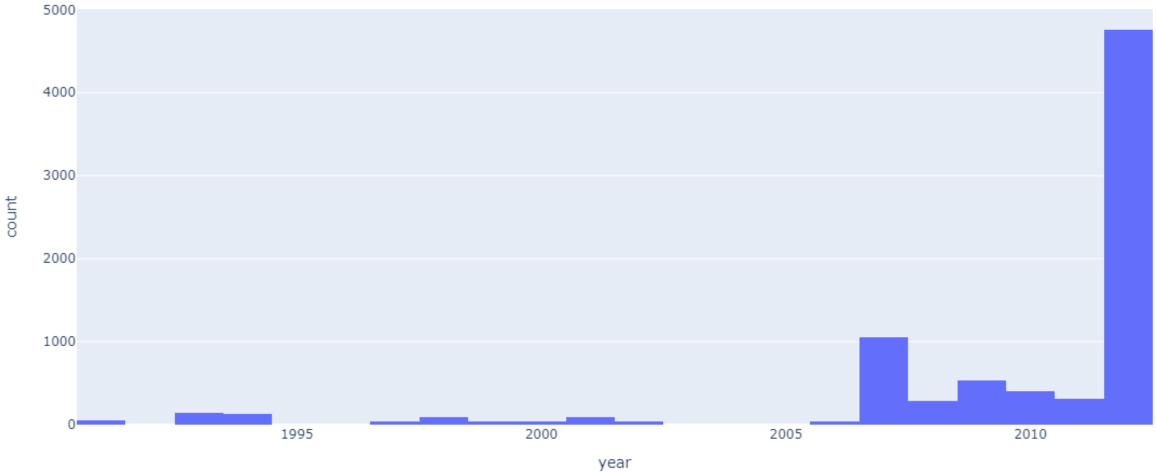
```python
    for index, row in eda_df.iterrows():
        wordsl = word_tokenize(row[col].lower())
        print(index, wordsl)

        if len(wordsl)>1:
            year = re.findall(pattern,wordsl[len(wordsl)-1])#row[0])
            if year:
                eda_df.loc[index, 'year'] = year[0]
            body = list(set(wordsl).intersection(car_body_type))
            if body:
                eda_df.loc[index, 'body'] = body[0]
            make = list(set(wordsl).intersection(car_make))
            if make:
                eda_df.loc[index, 'make'] = ' '.join(make)
            iden=year + body + make
            print(iden)
            model = list(set(wordsl).difference(iden))
            print(model)
            if model:
                eda_df.loc[index, 'model'] = ' '.join(model)
```

Display Few records:

	ImageName	X1	Y1	X2	Y2	Class	CarLabel	year	make	model	body
0	00001.jpg	30	52	246	147	181	Suzuki Aerio Sedan 2007	2007	suzuki	aerio	sedan
1	00002.jpg	100	19	576	203	103	Ferrari 458 Italia Convertible 2012	2012	ferrari	458 italia	convertible
2	00003.jpg	51	105	968	659	145	Jeep Patriot SUV 2012	2012	jeep	patriot	suv
3	00004.jpg	67	84	581	407	187	Toyota Camry Sedan 2012	2012	toyota	camry	sedan
4	00005.jpg	140	151	593	339	185	Tesla Model S Sedan 2012	2012	tesla	s model	sedan

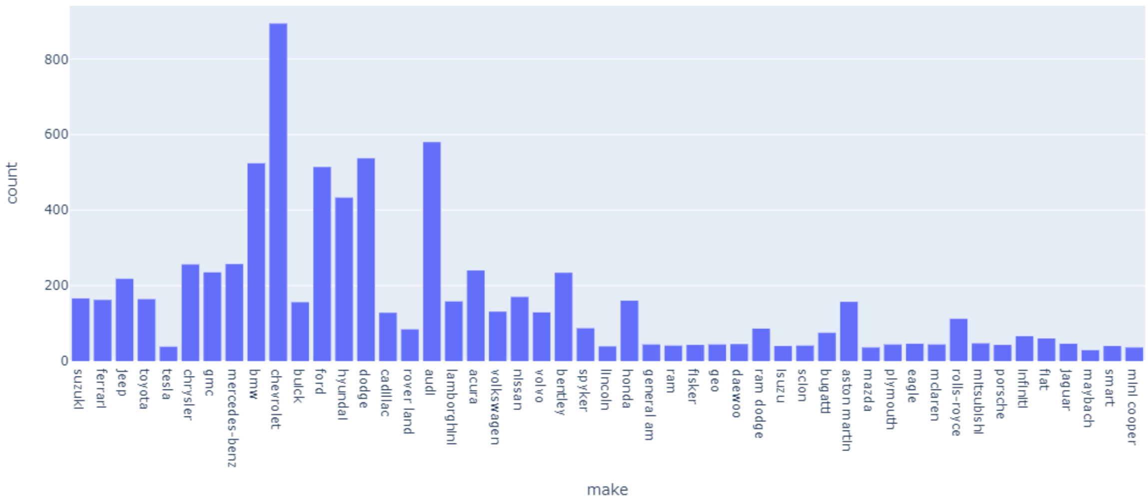
- **Display bar charts for different columns**

  - Count of different cars by year



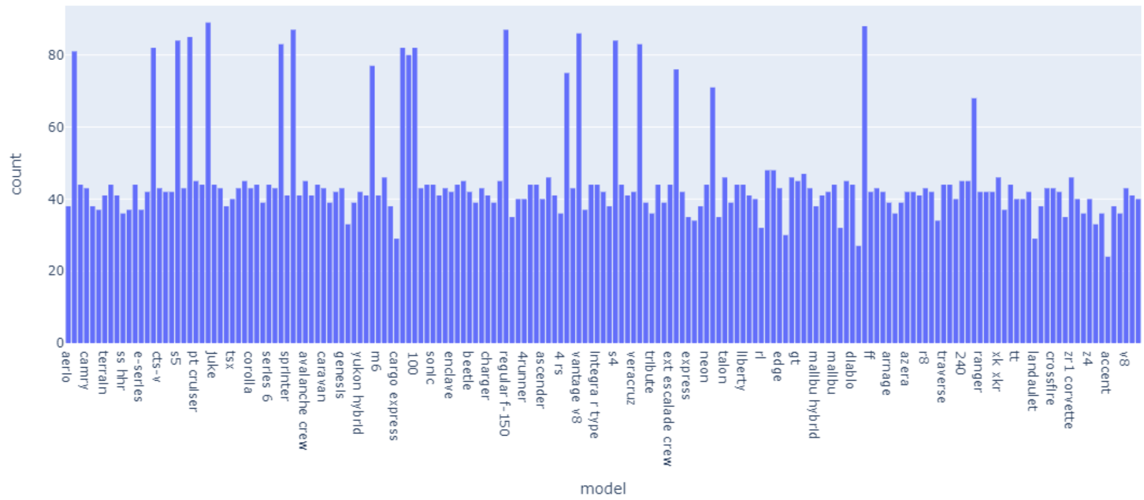
As seen, most of cars are of make year between 2007 – 2012

➢ Count of different cars by make



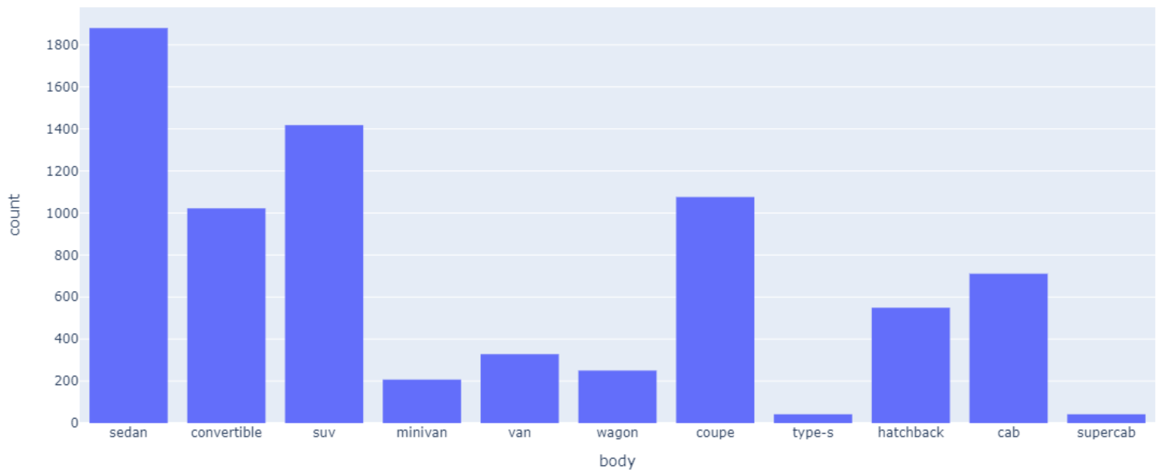
As seen, most of cars are of make Chevrolet, audi, bmw and dodge. ‘Mini Cooper’, ‘Smart’ and ‘Jaguar’ are less represented. Therefore, an effective model would be the one which can identify cars belonging to these classes.

➢ Count of different cars by model



As seen, cars evenly belong to different models, with some models having more number of cars.

➢ Count of cars by different body types

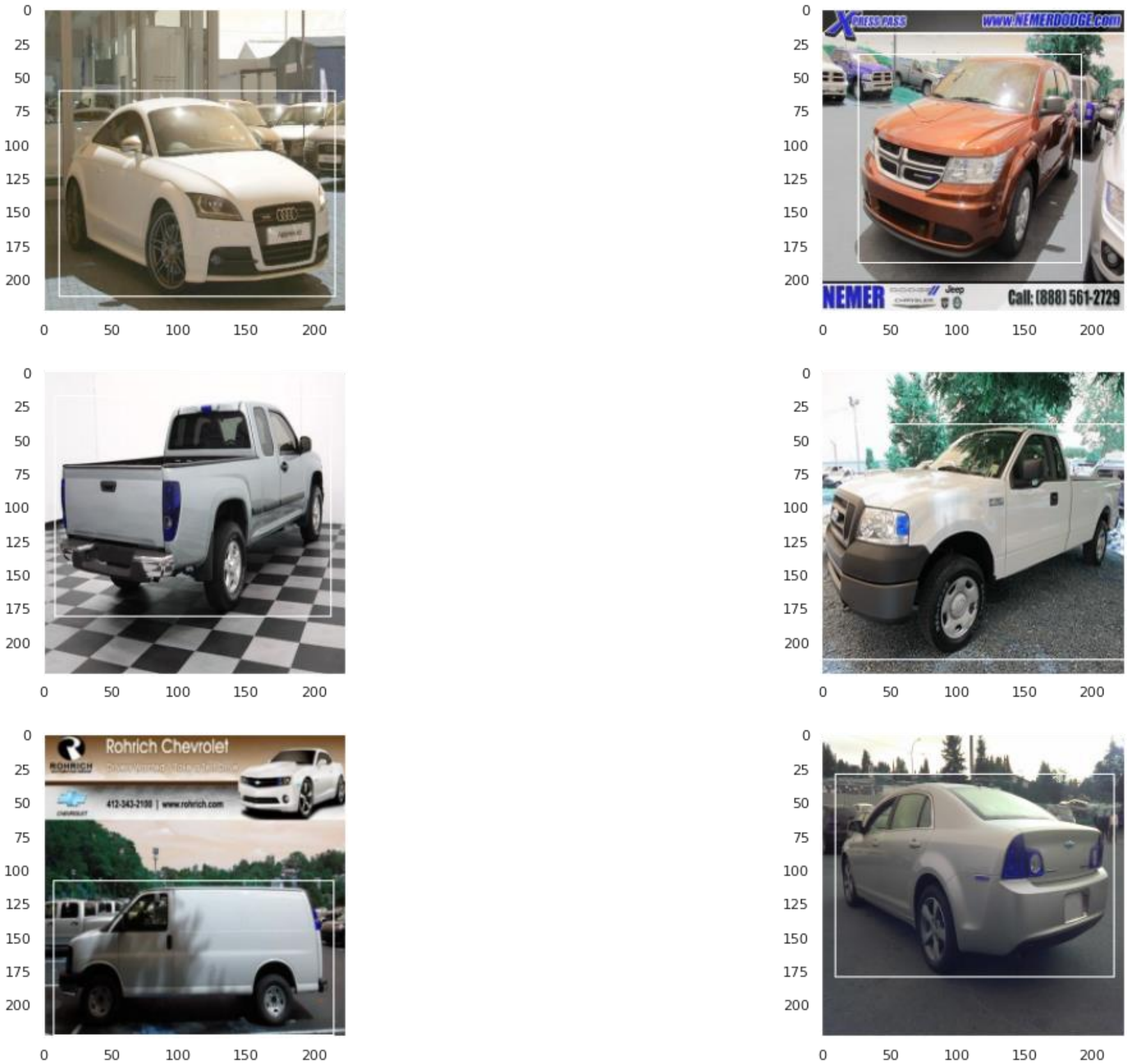


As seen, most of cars are of type Sedan

- **Display images with bounding box.**

```
IMAGE_SIZE = 224
IMAGE_HEIGHT = IMAGE_SIZE
IMAGE_WIDTH = IMAGE_SIZE

HEIGHT_CELLS = 28
WIDTH_CELLS = 28


print ( 'Generating bounding boxes images for Eg Train Data')

i = 1
      plt.figure(figsize=(20,20))
      for no in [0 , 6, 67, 89 , 99, 340 ]:
          eg_car  = car_train_df.iloc[ no ]

          path = 'Car Images/Train Images/{0}/{1}'.format( eg_car['CarLabel'], eg
      _car['ImageName'] )
          img = cv2.imread( path )
          img_shape = img.shape
          img = cv2.resize(img, dsize = (IMAGE_SIZE, IMAGE_SIZE), interpolation=c
      v2.INTER_AREA)


          x1 = int(eg_car['X1']    * IMAGE_SIZE   / img_shape[1] -
      3 )            # Normalize bounding box by image size
          y1 = int(eg_car['Y1']    * IMAGE_SIZE / img_shape[0] - 3 )         # Norm
      alize bounding box by image size
          x2 = int(eg_car['X2']    * IMAGE_SIZE / img_shape[1] + 3)            # No
      rmalize bounding box by image size
          y2 = int(eg_car['Y2']    * IMAGE_SIZE / img_shape[0] + 3 )           # No
      rmalize bounding box by image size


          cv2.rectangle(img, (x1, y1), (x2, y2), (255,255,255) )

          i +=1
          plt.subplot(4,2,i+1)
          plt.grid(False)
```

```
    plt.imshow(img);
```



- **Save training images with bounding box.**

  We will extract bounding boxes and then save those as images

```
for i in range(len(car_train_df)):
```

```
  eg_car  = car_train_df.iloc[i]
  source_path = 'Car Images/Train Images/{0}/{1}'.format( eg_car['CarLabel'], eg_
car['ImageName'] )
  dest_path = 'Car Images/Train Images Annoted/{0}/{1}'.format( eg_car['CarLabel'
], eg_car['ImageName'] )

  image = cv2.imread(source_path)
  if image is None:
    print(source_path)

  x1 = int(eg_car['X1'])
  y1 = int(eg_car['Y1'])
  x2 = int(eg_car['X2'])
  y2 = int(eg_car['Y2'])

  im2 = image[y1:y2,x1:x2]
  im2 = cv2.resize(im2, (IMAGE_SIZE, IMAGE_SIZE))

  destdirname = 'Car Images/Train Images Annoted/{0}'.format( eg_car['CarLabel'])
  destfilename= eg_car['ImageName']

  if not os.path.exists(destdirname):
    os.mkdir(destdirname)

  cv2.imwrite(os.path.join(destdirname, destfilename), im2)
```

Display few cropped training images

```
eg_car  = car_train_df.iloc[8]
path = 'Car Images/Train Images Annoted/{0}/{1}'.format( eg_car['CarLabel'], eg_c
ar['ImageName'] )
img = cv2.imread( path )
plt.grid(False)
plt.imshow(img)
```

```
eg_car  = car_train_df.iloc[16]
path = 'Car Images/Train Images Annoted/{0}/{1}'.format( eg_car['CarLabel'], eg_c
ar['ImageName'] )
img = cv2.imread( path )
plt.grid(False)
plt.imshow(img)
```

Do the same for test images.

- **Load the cropped train & test images using ImageDataGenerator**

```
train_path = 'Car Images/Train Images Annoted'

test_path = 'Car Images/Test Images Annoted'
```

```python
BATCH_SIZE = 32
IMG_SIZE = (224, 224)

train_datagen = ImageDataGenerator(
    rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
    train_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical')

validation_generator = test_datagen.flow_from_directory(
    test_path,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical')
```

```
Found 8144 images belonging to 197 classes.
Found 8041 images belonging to 197 classes.
```

Train Different Models

For this problem, we tried following models:

1) Custom CNN Classifier
2) ResNet50 (with multiple layers)
3) VGG16
4) ResNet50 (without multiple layers)
5) InceptionResNetV2

A. Custom CNN Classifier

1. Create the model

```
# Initialising the CNN classifier
classifier = Sequential()

INPUT_SIZE = (224, 224, 3)

# Add a Convolution layer with 32 kernels of 3X3 shape with activation function R
eLU
classifier.add(Conv2D(32, (3, 3), input_shape = INPUT_SIZE, activation = 'relu',
padding = 'same'))

# Add a Max Pooling layer of size 2X2
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Add another Convolution layer with 32 kernels of 3X3 shape with activation func
tion ReLU
```

```
classifier.add(Conv2D(32, (3, 3), activation = 'relu', padding = 'same'))

# Adding another pooling layer
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Add another Convolution layer with 32 kernels of 3X3 shape with activation func
tion ReLU
classifier.add(Conv2D(32, (3, 3), activation = 'relu', padding = 'same'))

# Adding another pooling layer
classifier.add(MaxPooling2D(pool_size = (2, 2)))

# Flattening the layer before fully connected layers
classifier.add(Flatten())

# Adding a fully connected layer with 512 neurons
classifier.add(Dense(units = 512, activation = 'relu'))

# Adding dropout with probability 0.5
classifier.add(Dropout(0.5))


# Adding a fully connected layer with 128 neurons
classifier.add(Dense(units = 128, activation = 'relu'))


# The final output layer with output size 197 classes for  the categorical classi
fcation
classifier.add(Dense(units = 197, activation = 'softmax'))
```

2. Summary

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d_3 (Conv2D)            (None, 224, 224, 32)      896       
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 112, 112, 32)      0         
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 112, 112, 32)      9248      
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 56, 56, 32)        0         
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 56, 56, 32)        9248      
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 28, 28, 32)        0         
_________________________________________________________________
flatten_1 (Flatten)          (None, 25088)             0         
_________________________________________________________________
dense_3 (Dense)              (None, 512)               12845568  
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0         
```

```
_________________________________________________________________
dense_4 (Dense)              (None, 128)               65664
_________________________________________________________________
dense_5 (Dense)              (None, 197)               25413
=================================================================
Total params: 12,956,037
Trainable params: 12,956,037
Non-trainable params: 0
```

---

3. Define Optimizer

```
opt = Adam(learning_rate=0.001, beta_1=0.9, beta_2=0.999, epsilon=None, decay=
0.001, amsgrad=False)
classifier.compile(optimizer = opt, loss = 'categorical_crossentropy', metrics
 = ['accuracy'])
```

4. Training [Forward pass and Backpropagation]

```
#Early stopping
early = EarlyStopping(monitor='val_accuracy',min_delta=0,patience=40,verbose=1,mo
de='auto')

# There are 3823 training images and 500 test images in total
hist_CNNClassifier = classifier.fit_generator(train_generator,
                       steps_per_epoch = int(train_generator.samples/BATCH_SIZE),
                       epochs = 20,
                       validation_data = validation_generator,
                      validation_steps = int(validation_generator.samples/BATCH_SIZE
                    ),
                       callbacks = [early])
```

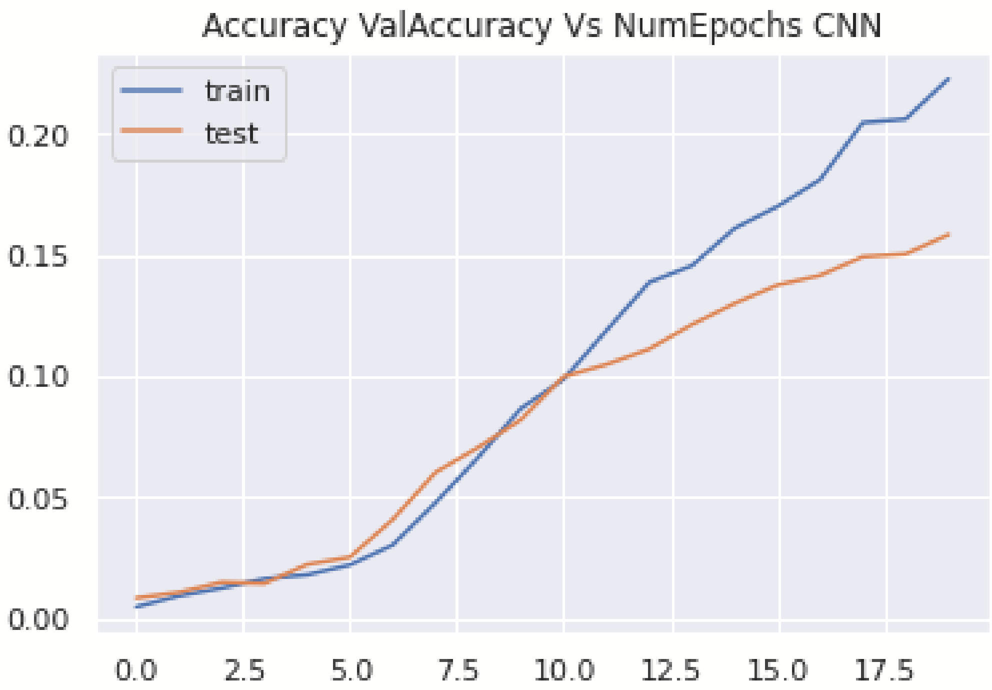
```
Epoch 1/20
254/254 [==============================] - 2523s 10s/step - loss: 5.2871 -
accuracy: 0.0048 - val_loss: 5.2809 - val_accuracy: 0.0085
Epoch 2/20
254/254 [==============================] - 188s 739ms/step - loss: 5.2378 -
accuracy: 0.0094 - val_loss: 5.1779 - val_accuracy: 0.0108
Epoch 3/20
254/254 [==============================] - 187s 736ms/step - loss: 5.1521 -
accuracy: 0.0126 - val_loss: 5.1202 - val_accuracy: 0.0149
Epoch 4/20
254/254 [==============================] - 185s 728ms/step - loss: 5.1024 -
accuracy: 0.0164 - val_loss: 5.1068 - val_accuracy: 0.0144
Epoch 5/20
254/254 [==============================] - 184s 724ms/step - loss: 5.0617 -
accuracy: 0.0180 - val_loss: 5.0464 - val_accuracy: 0.0223
```

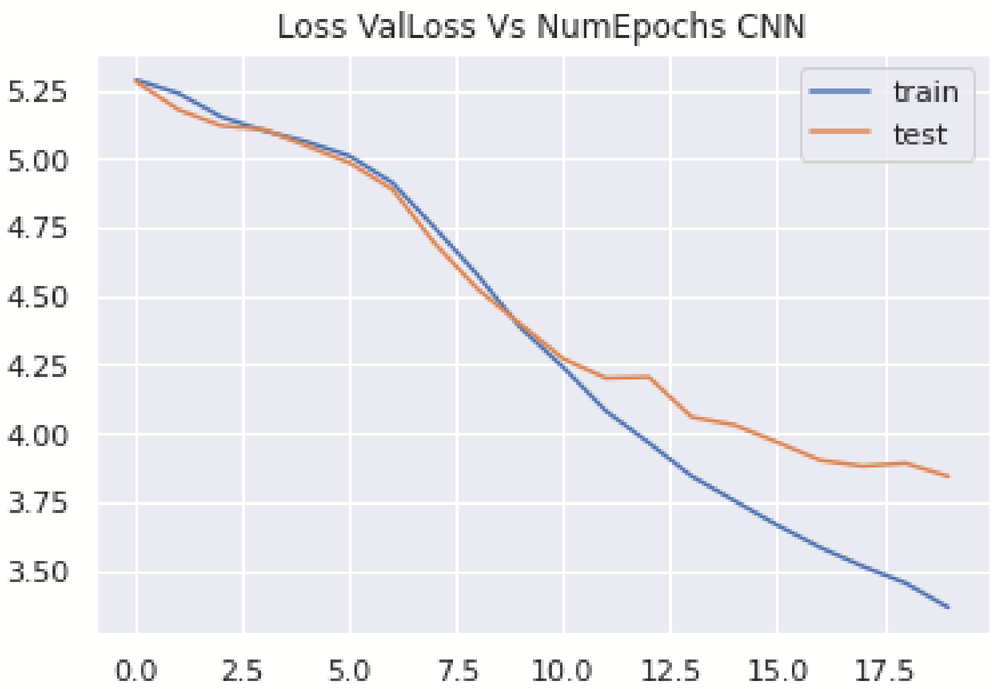
```
Epoch 6/20
254/254 [==============================] - 182s 719ms/step - loss: 5.0102 -
accuracy: 0.0221 - val_loss: 4.9850 - val_accuracy: 0.0253
Epoch 7/20
254/254 [==============================] - 184s 726ms/step - loss: 4.9130 -
accuracy: 0.0303 - val_loss: 4.8883 - val_accuracy: 0.0408
Epoch 8/20
254/254 [==============================] - 184s 724ms/step - loss: 4.7473 -
accuracy: 0.0477 - val_loss: 4.6899 - val_accuracy: 0.0603
Epoch 9/20
254/254 [==============================] - 184s 725ms/step - loss: 4.5756 -
accuracy: 0.0664 - val_loss: 4.5243 - val_accuracy: 0.0706
Epoch 10/20
254/254 [==============================] - 184s 725ms/step - loss: 4.3834 -
accuracy: 0.0867 - val_loss: 4.3964 - val_accuracy: 0.0820
Epoch 11/20
254/254 [==============================] - 184s 725ms/step - loss: 4.2401 -
accuracy: 0.0987 - val_loss: 4.2711 - val_accuracy: 0.0999
Epoch 12/20
254/254 [==============================] - 183s 723ms/step - loss: 4.0806 -
accuracy: 0.1187 - val_loss: 4.2011 - val_accuracy: 0.1048
Epoch 13/20
254/254 [==============================] - 183s 722ms/step - loss: 3.9654 -
accuracy: 0.1387 - val_loss: 4.2050 - val_accuracy: 0.1112
Epoch 14/20
254/254 [==============================] - 184s 727ms/step - loss: 3.8438 -
accuracy: 0.1456 - val_loss: 4.0591 - val_accuracy: 0.1213
Epoch 15/20
254/254 [==============================] - 185s 728ms/step - loss: 3.7540 -
accuracy: 0.1610 - val_loss: 4.0304 - val_accuracy: 0.1300
Epoch 16/20
254/254 [==============================] - 184s 726ms/step - loss: 3.6654 -
accuracy: 0.1700 - val_loss: 3.9678 - val_accuracy: 0.1376
Epoch 17/20
254/254 [==============================] - 183s 722ms/step - loss: 3.5846 -
accuracy: 0.1811 - val_loss: 3.9020 - val_accuracy: 0.1416
Epoch 18/20
254/254 [==============================] - 183s 723ms/step - loss: 3.5153 -
accuracy: 0.2046 - val_loss: 3.8804 - val_accuracy: 0.1493
Epoch 19/20
254/254 [==============================] - 184s 725ms/step - loss: 3.4546 -
accuracy: 0.2061 - val_loss: 3.8908 - val_accuracy: 0.1504
Epoch 20/20
      254/254 [==============================] - 184s 725ms/step - loss: 3.3658 -
      accuracy: 0.2226 - val_loss: 3.8429 - val_accuracy: 0.1584
```

5. Accuracy and Loss for Training and Validation

```
train_loss = hist_CNNClassifier.history['loss']
val_loss   = hist_CNNClassifier.history['val_loss']

xc = hist_CNNClassifier.epoch
plt.title("Accuracy ValAccuracy Vs NumEpochs CNN")
plt.plot(xc,hist_CNNClassifier.history['accuracy'], label='train')
plt.plot(xc,hist_CNNClassifier.history['val_accuracy'], label='test')
```

```
plt.legend()
plt.show()

plt.figure()
plt.title("Loss ValLoss Vs NumEpochs CNN")
plt.plot(xc, train_loss,label='train')
plt.plot(xc, val_loss,label='test')
plt.legend()
plt.show
```





Graph shows that model tends to increase both training and validation accuracy, and decrease training and validation loss with each epoch.

6. Evaluation

```
train_acc = classifier.evaluate_generator(train_generator,steps = int(train_ge
nerator.samples/BATCH_SIZE))
val_acc = classifier.evaluate_generator(validation_generator, steps = int(vali
dation_generator.samples/BATCH_SIZE))
print(train_acc[1])
print(val_acc[1])
```

```
0.3591289222240448
0.15861554443836212
```

Final evaluation shows that overall training accuracy is only around 36% and validation accuracy is around 16%. Therefore, this reveals both high bias and high variance issues.

7. Store result in a dataframe for final comparison of different models

```
#Store the Performance Matrix for each model in a dataframe for final comparis
on
resultsDf = pd.DataFrame({'Model':['CNN'], 'Train_Accuracy': train_acc[1],'Tes
t_Accuracy': val_acc[1],
                          })
resultsDf = resultsDf[['Model', 'Train_Accuracy','Test_Accuracy']]
resultsDf
```

	Model	Train_Accuracy	Test_Accuracy
0	CNN	0.359129	0.158616

8. Pickle the model for future use

```
classifier.save('./classifier.h5')

classifier.save_weights('./classifier_weights.h5')
```

B. ResNet50 (with multiple layers)

1. Creating the model

```
resnet50 = resnet50
conv_model = resnet50.ResNet50(weights='imagenet', include_top=False, input_shape
=(224,224,3))
x = Flatten()(conv_model.layers[-1].output)
x = Dense(512, activation='relu')(x)
x = Dense(224, activation='relu')(x)
x = Dense(224, activation='relu')(x)
x = Dense(224, activation='relu')(x)
x = Dense(224, activation='relu')(x)
x = Dense(224, activation='sigmoid')(x)
x = Dense(224, activation='sigmoid')(x)
predictions = Dense(197, activation='softmax')(x)

full_model = Model(inputs=conv_model.input, outputs=predictions)
```

2. Summary of model

```
full_model.summary()
```

```
Model: "model_1"
__________________________________________________________________________________
__________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================
=================
input_3 (InputLayer)            [(None, 224, 224, 3) 0
__________________________________________________________________________________
__________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_3[0][0]
__________________________________________________________________________________
__________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472        conv1_pad[0][0]
__________________________________________________________________________________
__________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256         conv1_conv[0][0]
__________________________________________________________________________________
__________________
conv1_relu (Activation)         (None, 112, 112, 64) 0           conv1_bn[0][0]
__________________________________________________________________________________
__________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0           conv1_relu[0][0]
__________________________________________________________________________________
__________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0           pool1_pad[0][0]
__________________________________________________________________________________
__________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160        pool1_pool[0][0]
__________________________________________________________________________________
__________________
```

```
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0
conv2_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928
conv2_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0
conv2_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640       pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640
conv2_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_add (Add)          (None, 56, 56, 256)  0
conv2_block1_0_bn[0][0]

conv2_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_out (Activation)   (None, 56, 56, 256)  0
conv2_block1_add[0][0]
__________________________________________________________________________________________________
conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448
conv2_block1_out[0][0]
__________________________________________________________________________________________________
conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0
conv2_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928
conv2_block2_1_relu[0][0]
__________________________________________________________________________________________________
```

```
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0
conv2_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640
conv2_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_add (Add)          (None, 56, 56, 256)  0
conv2_block1_out[0][0]

conv2_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_out (Activation)   (None, 56, 56, 256)  0
conv2_block2_add[0][0]
__________________________________________________________________________________________________
conv2_block3_1_conv (Conv2D)    (None, 56, 56, 64)   16448
conv2_block2_out[0][0]
__________________________________________________________________________________________________
conv2_block3_1_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_1_relu (Activation (None, 56, 56, 64)   0
conv2_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_2_conv (Conv2D)    (None, 56, 56, 64)   36928
conv2_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block3_2_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_2_relu (Activation (None, 56, 56, 64)   0
conv2_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_3_conv (Conv2D)    (None, 56, 56, 256)  16640
conv2_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block3_3_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_add (Add)          (None, 56, 56, 256)  0
conv2_block2_out[0][0]
```

```
conv2_block3_3_bn[0][0]
__________________________________________________________________________________
________________
conv2_block3_out (Activation)   (None, 56, 56, 256)  0
conv2_block3_add[0][0]
__________________________________________________________________________________
________________
conv3_block1_1_conv (Conv2D)    (None, 28, 28, 128)  32896
conv2_block3_out[0][0]
__________________________________________________________________________________
________________
conv3_block1_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block1_1_conv[0][0]
__________________________________________________________________________________
________________
conv3_block1_1_relu (Activation (None, 28, 28, 128)  0
conv3_block1_1_bn[0][0]
__________________________________________________________________________________
________________
conv3_block1_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block1_1_relu[0][0]
__________________________________________________________________________________
________________
conv3_block1_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block1_2_conv[0][0]
__________________________________________________________________________________
________________
conv3_block1_2_relu (Activation (None, 28, 28, 128)  0
conv3_block1_2_bn[0][0]
__________________________________________________________________________________
________________
conv3_block1_0_conv (Conv2D)    (None, 28, 28, 512)  131584
conv2_block3_out[0][0]
__________________________________________________________________________________
________________
conv3_block1_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block1_2_relu[0][0]
__________________________________________________________________________________
________________
conv3_block1_0_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block1_0_conv[0][0]
__________________________________________________________________________________
________________
conv3_block1_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block1_3_conv[0][0]
__________________________________________________________________________________
________________
conv3_block1_add (Add)          (None, 28, 28, 512)  0
conv3_block1_0_bn[0][0]

conv3_block1_3_bn[0][0]
__________________________________________________________________________________
________________
conv3_block1_out (Activation)   (None, 28, 28, 512)  0
conv3_block1_add[0][0]
__________________________________________________________________________________
________________
conv3_block2_1_conv (Conv2D)    (None, 28, 28, 128)  65664
conv3_block1_out[0][0]
```

```
__________________________________________________________________________________________________
conv3_block2_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_1_relu (Activation (None, 28, 28, 128)  0
conv3_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_2_relu (Activation (None, 28, 28, 128)  0
conv3_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_add (Add)          (None, 28, 28, 512)  0
conv3_block1_out[0][0]

conv3_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_out (Activation)   (None, 28, 28, 512)  0
conv3_block2_add[0][0]
__________________________________________________________________________________________________
conv3_block3_1_conv (Conv2D)    (None, 28, 28, 128)  65664
conv3_block2_out[0][0]
__________________________________________________________________________________________________
conv3_block3_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_1_relu (Activation (None, 28, 28, 128)  0
conv3_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block3_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block3_2_conv[0][0]
__________________________________________________________________________________________________
```

```
conv3_block3_2_relu (Activation (None, 28, 28, 128)  0
conv3_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block3_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_add (Add)          (None, 28, 28, 512)  0
conv3_block2_out[0][0]

conv3_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_out (Activation)   (None, 28, 28, 512)  0
conv3_block3_add[0][0]
__________________________________________________________________________________________________
conv3_block4_1_conv (Conv2D)    (None, 28, 28, 128)  65664
conv3_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block4_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block4_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_1_relu (Activation (None, 28, 28, 128)  0
conv3_block4_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block4_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block4_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_2_relu (Activation (None, 28, 28, 128)  0
conv3_block4_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block4_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block4_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_add (Add)          (None, 28, 28, 512)  0
conv3_block3_out[0][0]

conv3_block4_3_bn[0][0]
__________________________________________________________________________________________________
```

```
conv3_block4_out (Activation)   (None, 28, 28, 512)  0
conv3_block4_add[0][0]
__________________________________________________________________________________________________
conv4_block1_1_conv (Conv2D)    (None, 14, 14, 256)  131328
conv3_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block1_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_1_relu (Activation (None, 14, 14, 256)  0
conv4_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block1_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_2_relu (Activation (None, 14, 14, 256)  0
conv4_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_0_conv (Conv2D)    (None, 14, 14, 1024) 525312
conv3_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block1_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block1_0_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_add (Add)          (None, 14, 14, 1024) 0
conv4_block1_0_bn[0][0]

conv4_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_out (Activation)   (None, 14, 14, 1024) 0
conv4_block1_add[0][0]
__________________________________________________________________________________________________
conv4_block2_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block1_out[0][0]
__________________________________________________________________________________________________
conv4_block2_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block2_1_conv[0][0]
```

```
__________________________________________________________________________________________________
conv4_block2_1_relu (Activation (None, 14, 14, 256)  0           conv4_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block2_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_2_relu (Activation (None, 14, 14, 256)  0           conv4_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_3_conv (Conv2D)    (None, 14, 14, 1024) 263168      conv4_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block2_3_bn (BatchNormali (None, 14, 14, 1024) 4096        conv4_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_add (Add)          (None, 14, 14, 1024) 0           conv4_block1_out[0][0]
                                                                 conv4_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_out (Activation)   (None, 14, 14, 1024) 0           conv4_block2_add[0][0]
__________________________________________________________________________________________________
conv4_block3_1_conv (Conv2D)    (None, 14, 14, 256)  262400      conv4_block2_out[0][0]
__________________________________________________________________________________________________
conv4_block3_1_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_1_relu (Activation (None, 14, 14, 256)  0           conv4_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_2_conv (Conv2D)    (None, 14, 14, 256)  590080      conv4_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_2_bn (BatchNormali (None, 14, 14, 256)  1024        conv4_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_2_relu (Activation (None, 14, 14, 256)  0           conv4_block3_2_bn[0][0]
__________________________________________________________________________________________________
```

```
conv4_block3_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_add (Add)          (None, 14, 14, 1024) 0
conv4_block2_out[0][0]

conv4_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_out (Activation)   (None, 14, 14, 1024) 0
conv4_block3_add[0][0]
__________________________________________________________________________________________________
conv4_block4_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block3_out[0][0]
__________________________________________________________________________________________________
conv4_block4_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block4_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_1_relu (Activation (None, 14, 14, 256)  0
conv4_block4_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block4_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block4_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block4_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_2_relu (Activation (None, 14, 14, 256)  0
conv4_block4_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block4_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block4_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block4_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block4_add (Add)          (None, 14, 14, 1024) 0
conv4_block3_out[0][0]

conv4_block4_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block4_out (Activation)   (None, 14, 14, 1024) 0
conv4_block4_add[0][0]
__________________________________________________________________________________________________
```

```
conv4_block5_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block4_out[0][0]
________________________________________________________________________________
__________________
conv4_block5_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block5_1_conv[0][0]
________________________________________________________________________________
__________________
conv4_block5_1_relu (Activation (None, 14, 14, 256)  0
conv4_block5_1_bn[0][0]
________________________________________________________________________________
__________________
conv4_block5_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block5_1_relu[0][0]
________________________________________________________________________________
__________________
conv4_block5_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block5_2_conv[0][0]
________________________________________________________________________________
__________________
conv4_block5_2_relu (Activation (None, 14, 14, 256)  0
conv4_block5_2_bn[0][0]
________________________________________________________________________________
__________________
conv4_block5_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block5_2_relu[0][0]
________________________________________________________________________________
__________________
conv4_block5_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block5_3_conv[0][0]
________________________________________________________________________________
__________________
conv4_block5_add (Add)          (None, 14, 14, 1024) 0
conv4_block4_out[0][0]

conv4_block5_3_bn[0][0]
________________________________________________________________________________
__________________
conv4_block5_out (Activation)   (None, 14, 14, 1024) 0
conv4_block5_add[0][0]
________________________________________________________________________________
__________________
conv4_block6_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block5_out[0][0]
________________________________________________________________________________
__________________
conv4_block6_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block6_1_conv[0][0]
________________________________________________________________________________
__________________
conv4_block6_1_relu (Activation (None, 14, 14, 256)  0
conv4_block6_1_bn[0][0]
________________________________________________________________________________
__________________
conv4_block6_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block6_1_relu[0][0]
________________________________________________________________________________
__________________
conv4_block6_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block6_2_conv[0][0]
```

```
__________________________________________________________________________________________________
conv4_block6_2_relu (Activation (None, 14, 14, 256)  0
conv4_block6_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block6_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block6_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block6_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_add (Add)          (None, 14, 14, 1024) 0
conv4_block5_out[0][0]

conv4_block6_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_out (Activation)   (None, 14, 14, 1024) 0
conv4_block6_add[0][0]
__________________________________________________________________________________________________
conv5_block1_1_conv (Conv2D)    (None, 7, 7, 512)    524800
conv4_block6_out[0][0]
__________________________________________________________________________________________________
conv5_block1_1_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_1_relu (Activation (None, 7, 7, 512)    0
conv5_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_2_conv (Conv2D)    (None, 7, 7, 512)    2359808
conv5_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block1_2_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_2_relu (Activation (None, 7, 7, 512)    0
conv5_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_0_conv (Conv2D)    (None, 7, 7, 2048)   2099200
conv4_block6_out[0][0]
__________________________________________________________________________________________________
conv5_block1_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624
conv5_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block1_0_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block1_0_conv[0][0]
__________________________________________________________________________________________________
```

```
conv5_block1_3_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_add (Add)          (None, 7, 7, 2048)   0
conv5_block1_0_bn[0][0]

conv5_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_out (Activation)   (None, 7, 7, 2048)   0
conv5_block1_add[0][0]
__________________________________________________________________________________________________
conv5_block2_1_conv (Conv2D)    (None, 7, 7, 512)    1049088
conv5_block1_out[0][0]
__________________________________________________________________________________________________
conv5_block2_1_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_1_relu (Activation (None, 7, 7, 512)    0
conv5_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_2_conv (Conv2D)    (None, 7, 7, 512)    2359808
conv5_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_2_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_2_relu (Activation (None, 7, 7, 512)    0
conv5_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624
conv5_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block2_3_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block2_add (Add)          (None, 7, 7, 2048)   0
conv5_block1_out[0][0]

conv5_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block2_out (Activation)   (None, 7, 7, 2048)   0
conv5_block2_add[0][0]
__________________________________________________________________________________________________
conv5_block3_1_conv (Conv2D)    (None, 7, 7, 512)    1049088
conv5_block2_out[0][0]
__________________________________________________________________________________________________
```

```
conv5_block3_1_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block3_1_conv[0][0]
__________________________________________________________________________________
________________
conv5_block3_1_relu (Activation (None, 7, 7, 512)    0
conv5_block3_1_bn[0][0]
__________________________________________________________________________________
________________
conv5_block3_2_conv (Conv2D)    (None, 7, 7, 512)    2359808
conv5_block3_1_relu[0][0]
__________________________________________________________________________________
________________
conv5_block3_2_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block3_2_conv[0][0]
__________________________________________________________________________________
________________
conv5_block3_2_relu (Activation (None, 7, 7, 512)    0
conv5_block3_2_bn[0][0]
__________________________________________________________________________________
________________
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624
conv5_block3_2_relu[0][0]
__________________________________________________________________________________
________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block3_3_conv[0][0]
__________________________________________________________________________________
________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0
conv5_block2_out[0][0]

conv5_block3_3_bn[0][0]
__________________________________________________________________________________
________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0
conv5_block3_add[0][0]
__________________________________________________________________________________
________________
flatten_5 (Flatten)             (None, 100352)       0
conv5_block3_out[0][0]
__________________________________________________________________________________
________________
dense_18 (Dense)                (None, 512)          51380736    flatten_5[0][0]
__________________________________________________________________________________
________________
dense_19 (Dense)                (None, 224)          114912      dense_18[0][0]
__________________________________________________________________________________
________________
dense_20 (Dense)                (None, 224)          50400       dense_19[0][0]
__________________________________________________________________________________
________________
dense_21 (Dense)                (None, 224)          50400       dense_20[0][0]
__________________________________________________________________________________
________________
dense_22 (Dense)                (None, 224)          50400       dense_21[0][0]
__________________________________________________________________________________
________________
dense_23 (Dense)                (None, 224)          50400       dense_22[0][0]
__________________________________________________________________________________
________________
dense_24 (Dense)                (None, 224)          50400       dense_23[0][0]
```

```
________________________________________________________________________________
_________________
dense_25 (Dense)                (None, 197)          44325       dense_24[0][0]
================================================================================
=================
Total params: 75,379,685
Trainable params: 75,326,565
Non-trainable params: 53,120
      __________________________________________________________________________
_______________________
```

3. Define optimizer

```
opt= Adam(learning_rate=0.001)
```

4. Training [Forward pass and Backpropagation]

```
#Compile
full_model.compile(optimizer= opt, loss = 'categorical_crossentropy', metrics
= ['accuracy'])

#Early stopping
early = EarlyStopping(monitor='val_accuracy',min_delta=0,patience=40,verbose=1
,mode='auto')

res_classifier=full_model.fit_generator(train_generator,steps_per_epoch = 2, e
pochs =30, validation_data = validation_generator,
            validation_steps = 1,callbacks = [early])
```

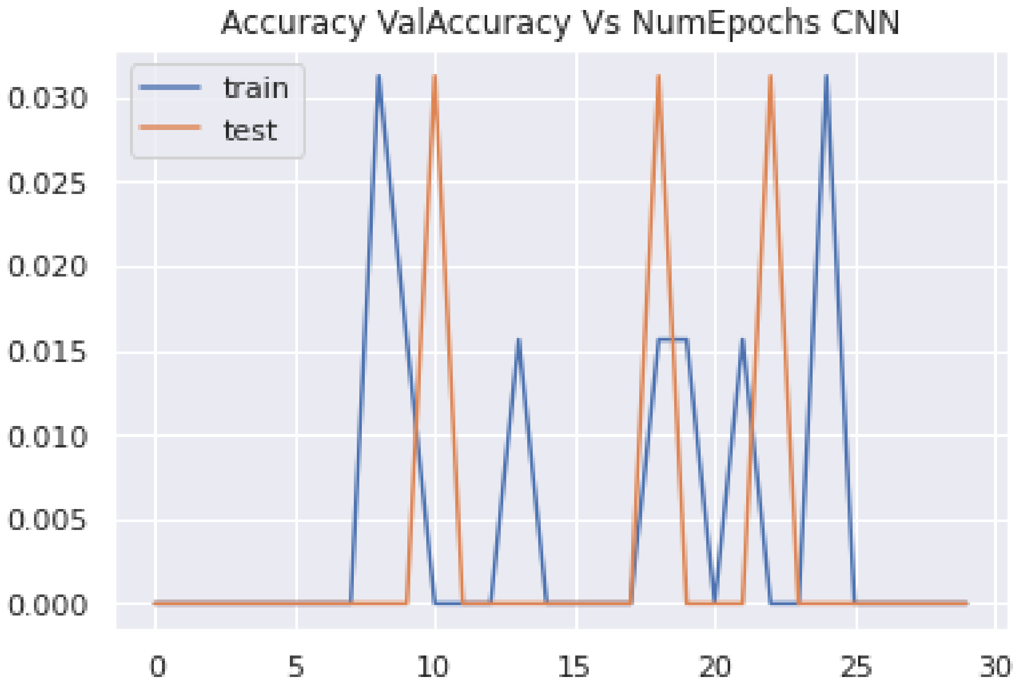
```
Epoch 1/30
2/2 [==============================] - 11s 2s/step - loss: 5.3621 - accuracy:
0.0000e+00 - val_loss: 5.5525 - val_accuracy: 0.0000e+00
Epoch 2/30
2/2 [==============================] - 2s 914ms/step - loss: 5.4601 -
accuracy: 0.0000e+00 - val_loss: 5.7100 - val_accuracy: 0.0000e+00
Epoch 3/30
2/2 [==============================] - 2s 1s/step - loss: 5.4117 - accuracy:
0.0000e+00 - val_loss: 5.5345 - val_accuracy: 0.0000e+00
Epoch 4/30
2/2 [==============================] - 2s 993ms/step - loss: 5.3791 -
accuracy: 0.0000e+00 - val_loss: 5.5610 - val_accuracy: 0.0000e+00
Epoch 5/30
```

```
2/2 [==============================] - 2s 1s/step - loss: 5.5947 - accuracy:
0.0000e+00 - val_loss: 5.6320 - val_accuracy: 0.0000e+00
Epoch 6/30
2/2 [==============================] - 2s 1s/step - loss: 5.3780 - accuracy:
0.0000e+00 - val_loss: 5.4623 - val_accuracy: 0.0000e+00
Epoch 7/30
2/2 [==============================] - 2s 1s/step - loss: 5.4169 - accuracy:
0.0000e+00 - val_loss: 5.5907 - val_accuracy: 0.0000e+00
Epoch 8/30
2/2 [==============================] - 2s 1s/step - loss: 5.4999 - accuracy:
0.0000e+00 - val_loss: 5.4237 - val_accuracy: 0.0000e+00
Epoch 9/30
2/2 [==============================] - 2s 965ms/step - loss: 5.4621 -
accuracy: 0.0312 - val_loss: 5.4819 - val_accuracy: 0.0000e+00
Epoch 10/30
2/2 [==============================] - 2s 1s/step - loss: 5.4418 - accuracy:
0.0156 - val_loss: 5.4853 - val_accuracy: 0.0000e+00
Epoch 11/30
2/2 [==============================] - 2s 1s/step - loss: 5.5179 - accuracy:
0.0000e+00 - val_loss: 5.3407 - val_accuracy: 0.0312
Epoch 12/30
2/2 [==============================] - 2s 955ms/step - loss: 5.3203 -
accuracy: 0.0000e+00 - val_loss: 5.5048 - val_accuracy: 0.0000e+00
Epoch 13/30
2/2 [==============================] - 2s 1s/step - loss: 5.3476 - accuracy:
0.0000e+00 - val_loss: 5.3331 - val_accuracy: 0.0000e+00
Epoch 14/30
2/2 [==============================] - 2s 954ms/step - loss: 5.2757 -
accuracy: 0.0156 - val_loss: 5.3322 - val_accuracy: 0.0000e+00
Epoch 15/30
2/2 [==============================] - 2s 1s/step - loss: 5.3678 - accuracy:
0.0000e+00 - val_loss: 5.4033 - val_accuracy: 0.0000e+00
Epoch 16/30
2/2 [==============================] - 2s 1s/step - loss: 5.4252 - accuracy:
0.0000e+00 - val_loss: 5.3253 - val_accuracy: 0.0000e+00
Epoch 17/30
2/2 [==============================] - 2s 1s/step - loss: 5.4463 - accuracy:
0.0000e+00 - val_loss: 5.3175 - val_accuracy: 0.0000e+00
Epoch 18/30
2/2 [==============================] - 2s 1s/step - loss: 5.4036 - accuracy:
0.0000e+00 - val_loss: 5.3590 - val_accuracy: 0.0000e+00
Epoch 19/30
2/2 [==============================] - 2s 987ms/step - loss: 5.3229 -
accuracy: 0.0156 - val_loss: 5.3885 - val_accuracy: 0.0312
Epoch 20/30
2/2 [==============================] - 2s 1s/step - loss: 5.2921 - accuracy:
0.0156 - val_loss: 5.3115 - val_accuracy: 0.0000e+00
Epoch 21/30
2/2 [==============================] - 2s 957ms/step - loss: 5.3349 -
accuracy: 0.0000e+00 - val_loss: 5.2952 - val_accuracy: 0.0000e+00
Epoch 22/30
2/2 [==============================] - 2s 993ms/step - loss: 5.3338 -
accuracy: 0.0156 - val_loss: 5.3405 - val_accuracy: 0.0000e+00
Epoch 23/30
2/2 [==============================] - 2s 994ms/step - loss: 5.3303 -
accuracy: 0.0000e+00 - val_loss: 5.3367 - val_accuracy: 0.0312
Epoch 24/30
2/2 [==============================] - 2s 1s/step - loss: 5.3819 - accuracy:
0.0000e+00 - val_loss: 5.2927 - val_accuracy: 0.0000e+00
Epoch 25/30
2/2 [==============================] - 2s 1s/step - loss: 5.3205 - accuracy:
```

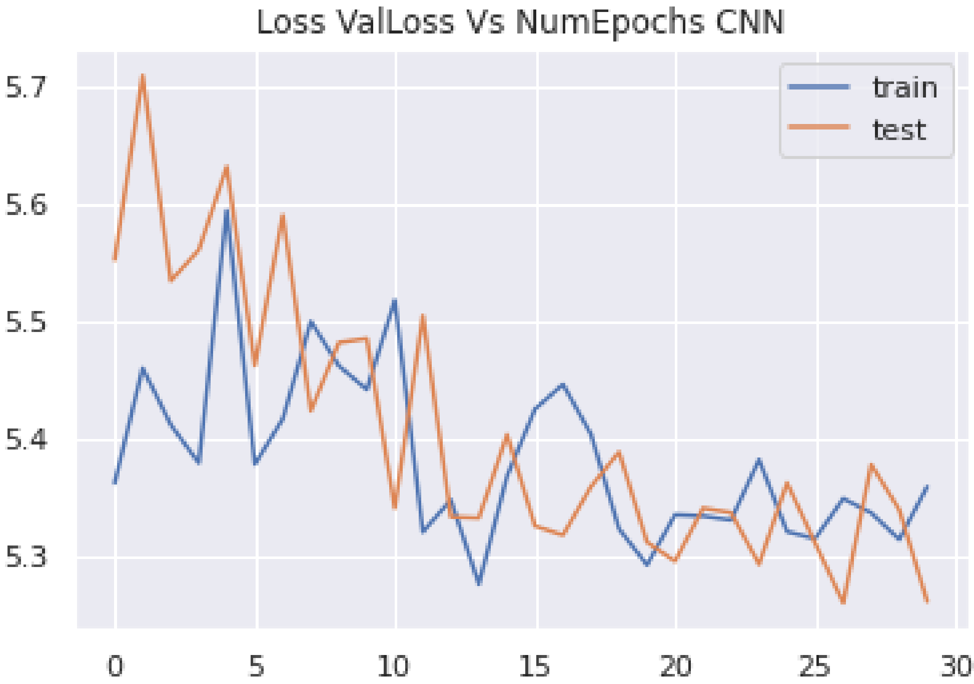
```
0.0312 - val_loss: 5.3621 - val_accuracy: 0.0000e+00
Epoch 26/30
2/2 [==============================] - 2s 1s/step - loss: 5.3144 - accuracy:
0.0000e+00 - val_loss: 5.3109 - val_accuracy: 0.0000e+00
Epoch 27/30
2/2 [==============================] - 2s 1s/step - loss: 5.3487 - accuracy:
0.0000e+00 - val_loss: 5.2597 - val_accuracy: 0.0000e+00
Epoch 28/30
2/2 [==============================] - 2s 958ms/step - loss: 5.3365 -
accuracy: 0.0000e+00 - val_loss: 5.3778 - val_accuracy: 0.0000e+00
Epoch 29/30
2/2 [==============================] - 2s 1s/step - loss: 5.3138 - accuracy:
0.0000e+00 - val_loss: 5.3392 - val_accuracy: 0.0000e+00
Epoch 30/30
        2/2 [==============================] - 2s 1s/step - loss: 5.3589 -
        accuracy: 0.0000e+00 - val_loss: 5.2610 - val_accuracy: 0.0000e+00
```

5. Plot Accuracy and Loss for Training and Validation

```
train_loss = res_classifier.history['loss']
val_loss   = res_classifier.history['val_loss']

xc = res_classifier.epoch
plt.title("Accuracy ValAccuracy Vs NumEpochs CNN")
plt.plot(xc,res_classifier.history['accuracy'], label='train')
plt.plot(xc,res_classifier.history['val_accuracy'], label='test')
plt.legend()
plt.show()

plt.figure()
plt.title("Loss ValLoss Vs NumEpochs CNN")
plt.plot(xc, train_loss,label='train')
plt.plot(xc, val_loss,label='test')
plt.legend()
plt.show
```



Plot shows that model tries to touch peaks and troughs with increasing epochs.



Plot shows that model tries to reduce loss for both training and validation dataset with each epoch.

6. **Evaluation**

```
train_acc = full_model.evaluate_generator(train_generator,steps = int(train_generator.samples/BATCH_SIZE))
val_acc = full_model.evaluate_generator(validation_generator, steps = int(validation_generator.samples/BATCH_SIZE))


print(train_acc[1])
print(val_acc[1])
```

```
0.005290354136377573
0.005229083821177483
```

Thus, this model performs very poorly on both training and validation dataset.

7. **Adding results to dataframe for final comparison**

```
#Adding Performance metrics of ResNet50 to the list
tempResultsDf = pd.DataFrame({'Model':['ResNet50'],  'Train_Accuracy': train_acc[1],'Test_Accuracy': val_acc[1]})
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf = resultsDf[['Model', 'Train_Accuracy','Test_Accuracy']]
resultsDf
```

	Model	Train_Accuracy	Test_Accuracy
0	CNN	0.359129	0.158616
0	ResNet50	0.005290	0.005229

C. VGG16

1. Creating the model

```
print ( 'VGG with custom FC Layers')
vgg_conv = VGG16(weights='imagenet', include_top=False, input_shape= (224,224,3))
```

```
# Freeze all the layers except for the last layer:
for layer in vgg_conv.layers:
    layer.trainable = False

x = Flatten()(vgg_conv.output)
x = Dense(197, activation='softmax')(x)
vgg_model = Model(vgg_conv.input, x)
```

```
VGG with custom FC Layers
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels_notop.h5
58892288/58889256 [==============================] - 1s 0us/step
58900480/58889256 [==============================] - 1s 0us/step
```

2. Summary of model

```
vgg_model.summary()

Model: "model_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_5 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
```

```
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten_7 (Flatten)          (None, 25088)             0
_________________________________________________________________
dense_27 (Dense)             (None, 197)               4942533
=================================================================
Total params: 19,657,221
Trainable params: 4,942,533
Non-trainable params: 14,714,688
```

3. Training [Forward pass and Backpropagation]

```
#Compile the model with Adam optimizer
vgg_model.compile(optimizer = Adam(learning_rate=0.001), loss = 'categorical_c
rossentropy', metrics = ['accuracy'])


early = EarlyStopping(monitor='val_accuracy',min_delta=0.01,patience=20,verbos
e=1,mode='auto')

#Training
vgg_classifier = vgg_model.fit_generator(train_generator, epochs =30, validati
on_data = validation_generator, callbacks = [early]  )
```

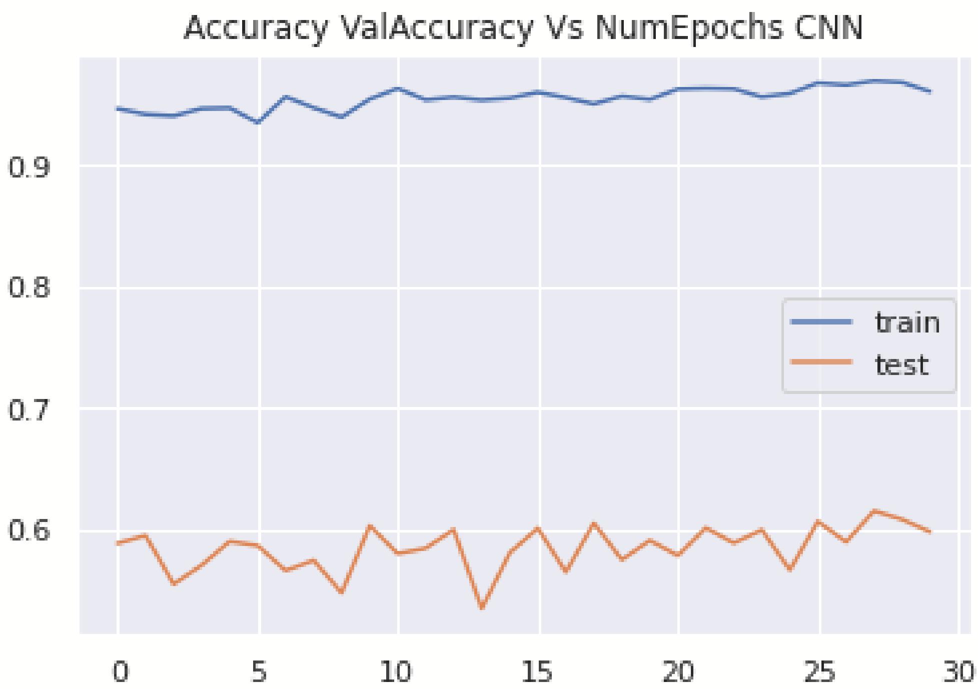
```
Epoch 1/30
255/255 [==============================] - 187s 731ms/step - loss: 0.3102 -
accuracy: 0.9467 - val_loss: 4.6444 - val_accuracy: 0.5884
Epoch 2/30
255/255 [==============================] - 183s 718ms/step - loss: 0.3438 -
accuracy: 0.9422 - val_loss: 4.7269 - val_accuracy: 0.5948
Epoch 3/30
255/255 [==============================] - 181s 711ms/step - loss: 0.3808 -
accuracy: 0.9408 - val_loss: 5.6650 - val_accuracy: 0.5547
Epoch 4/30
255/255 [==============================] - 180s 708ms/step - loss: 0.3101 -
accuracy: 0.9468 - val_loss: 5.3064 - val_accuracy: 0.5703
Epoch 5/30
255/255 [==============================] - 181s 709ms/step - loss: 0.3306 -
accuracy: 0.9473 - val_loss: 5.1069 - val_accuracy: 0.5900
Epoch 6/30
255/255 [==============================] - 180s 707ms/step - loss: 0.4257 -
accuracy: 0.9353 - val_loss: 5.2278 - val_accuracy: 0.5865
Epoch 7/30
255/255 [==============================] - 180s 708ms/step - loss: 0.2552 -
accuracy: 0.9565 - val_loss: 5.8028 - val_accuracy: 0.5659
Epoch 8/30
255/255 [==============================] - 180s 707ms/step - loss: 0.3380 -
accuracy: 0.9474 - val_loss: 5.7594 - val_accuracy: 0.5743
```

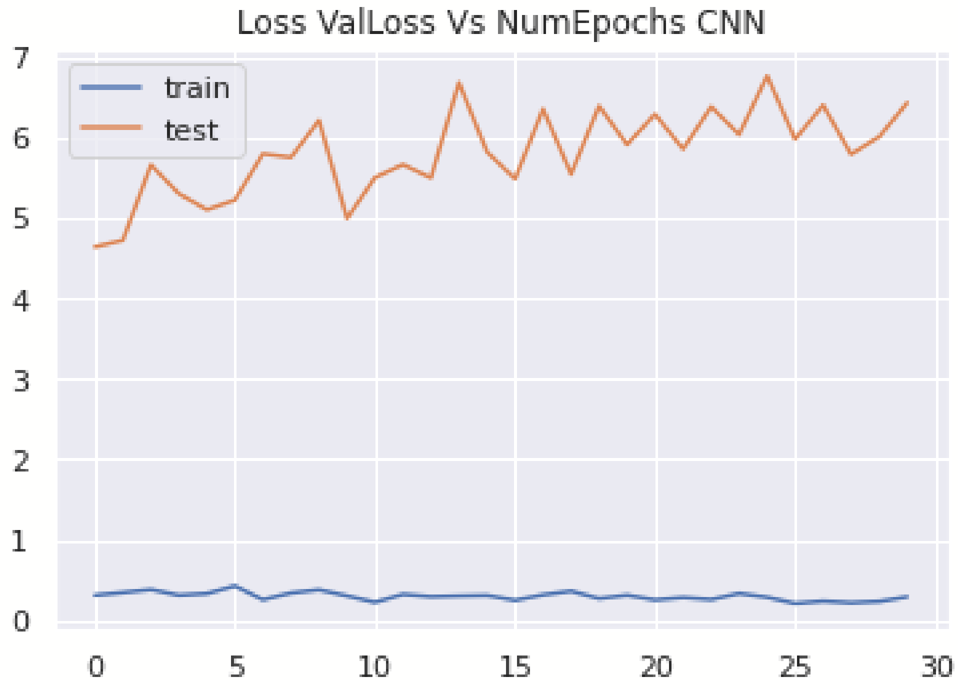
```
Epoch 9/30
255/255 [==============================] - 180s 705ms/step - loss: 0.3781 -
accuracy: 0.9398 - val_loss: 6.2191 - val_accuracy: 0.5476
Epoch 10/30
255/255 [==============================] - 180s 706ms/step - loss: 0.3021 -
accuracy: 0.9546 - val_loss: 4.9970 - val_accuracy: 0.6029
Epoch 11/30
255/255 [==============================] - 181s 709ms/step - loss: 0.2226 -
accuracy: 0.9634 - val_loss: 5.5075 - val_accuracy: 0.5799
Epoch 12/30
255/255 [==============================] - 181s 708ms/step - loss: 0.3211 -
accuracy: 0.9540 - val_loss: 5.6694 - val_accuracy: 0.5843
Epoch 13/30
255/255 [==============================] - 182s 715ms/step - loss: 0.2927 -
accuracy: 0.9563 - val_loss: 5.5035 - val_accuracy: 0.5997
Epoch 14/30
255/255 [==============================] - 183s 717ms/step - loss: 0.3017 -
accuracy: 0.9538 - val_loss: 6.6898 - val_accuracy: 0.5348
Epoch 15/30
255/255 [==============================] - 182s 715ms/step - loss: 0.3076 -
accuracy: 0.9554 - val_loss: 5.8260 - val_accuracy: 0.5806
Epoch 16/30
255/255 [==============================] - 183s 717ms/step - loss: 0.2479 -
accuracy: 0.9602 - val_loss: 5.4916 - val_accuracy: 0.6008
Epoch 17/30
255/255 [==============================] - 184s 723ms/step - loss: 0.3149 -
accuracy: 0.9558 - val_loss: 6.3606 - val_accuracy: 0.5646
Epoch 18/30
255/255 [==============================] - 184s 720ms/step - loss: 0.3615 -
accuracy: 0.9506 - val_loss: 5.5514 - val_accuracy: 0.6050
Epoch 19/30
255/255 [==============================] - 181s 711ms/step - loss: 0.2693 -
accuracy: 0.9570 - val_loss: 6.3985 - val_accuracy: 0.5748
Epoch 20/30
255/255 [==============================] - 182s 715ms/step - loss: 0.3107 -
accuracy: 0.9543 - val_loss: 5.9179 - val_accuracy: 0.5907
Epoch 21/30
255/255 [==============================] - 182s 713ms/step - loss: 0.2523 -
accuracy: 0.9629 - val_loss: 6.2966 - val_accuracy: 0.5783
Epoch 22/30
255/255 [==============================] - 182s 713ms/step - loss: 0.2820 -
accuracy: 0.9635 - val_loss: 5.8626 - val_accuracy: 0.6010
Epoch 23/30
255/255 [==============================] - 181s 709ms/step - loss: 0.2569 -
accuracy: 0.9630 - val_loss: 6.3886 - val_accuracy: 0.5886
Epoch 24/30
255/255 [==============================] - 181s 711ms/step - loss: 0.3322 -
accuracy: 0.9564 - val_loss: 6.0438 - val_accuracy: 0.5994
Epoch 25/30
255/255 [==============================] - 182s 714ms/step - loss: 0.2849 -
accuracy: 0.9592 - val_loss: 6.7721 - val_accuracy: 0.5663
Epoch 26/30
255/255 [==============================] - 181s 710ms/step - loss: 0.2081 -
accuracy: 0.9680 - val_loss: 5.9842 - val_accuracy: 0.6064
Epoch 27/30
255/255 [==============================] - 183s 717ms/step - loss: 0.2362 -
accuracy: 0.9662 - val_loss: 6.4080 - val_accuracy: 0.5895
Epoch 28/30
255/255 [==============================] - 182s 713ms/step - loss: 0.2165 -
accuracy: 0.9697 - val_loss: 5.7963 - val_accuracy: 0.6151
Epoch 29/30
```

```
255/255 [==============================] - 182s 713ms/step - loss: 0.2329 - accuracy: 0.9684 - val_loss: 6.0159 - val_accuracy: 0.6081
Epoch 30/30
255/255 [==============================] - 180s 707ms/step - loss: 0.2903 - accuracy: 0.9608 - val_loss: 6.4375 - val_accuracy: 0.5978
```

4. Plot Accuracy and Loss

```
train_loss = vgg_classifier.history['loss']
val_loss   = vgg_classifier.history['val_loss']

xc = vgg_classifier.epoch
plt.title("Accuracy ValAccuracy Vs NumEpochs CNN")
plt.plot(xc,vgg_classifier.history['accuracy'], label='train')
plt.plot(xc,vgg_classifier.history['val_accuracy'], label='test')
plt.legend()
plt.show()

plt.figure()
plt.title("Loss ValLoss Vs NumEpochs CNN")
plt.plot(xc, train_loss,label='train')
plt.plot(xc, val_loss,label='test')
plt.legend()
plt.show
```





Plot shows that both training and validation accuracy, and training and validation loss remains more or less constant over epochs.

5. Evaluation

```
train_acc = vgg_model.evaluate_generator(train_generator,steps = int(train_gen
erator.samples/BATCH_SIZE))
val_acc = vgg_model.evaluate_generator(validation_generator, steps = int(valid
ation_generator.samples/BATCH_SIZE))

print(train_acc[1])
print(val_acc[1])
```

```
0.9386072754859924
0.5639940500259399
```

Therefore, model shows a high training accuracy of around 94%. But validation accuracy is low at 56%. This shows a high variance problem.

6. **Add result to dataframe for final comparison**

```
#Adding Performance metrics of ResNet50 to the list
tempResultsDf = pd.DataFrame({'Model':['VGG16'],  'Train_Accuracy': train_acc[
1],'Test_Accuracy': val_acc[1]})
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf = resultsDf[['Model', 'Train_Accuracy','Test_Accuracy']]
resultsDf
```

	Model	Train_Accuracy	Test_Accuracy
0	CNN	0.359129	0.158616
0	ResNet50	0.005290	0.005229
0	VGG16	0.938607	0.563994

7. **Save model for future use**

```
vgg_model.save('./vgg.h5')

vgg_model.save_weights('./vgg_weights.h5')
```

D. ResNet50 (without multiple layers)

1. **Creating the model**

```
resnet_conv = ResNet50(weights='imagenet', include_top=False, input_shape=(224
,224,3))

# Freeze all the layers except for the last layer:
for layer in resnet_conv.layers:
    layer.trainable = False
```

```
x2 = Flatten()(resnet_conv.output)
x2 = Dense(197, activation='sigmoid')(x2)
resnet = Model(resnet_conv.input, x2)
```

2. Summary of model

```
resnet.summary()
```

```
Model: "model_4"
______________________________________________________________________________
____________________
Layer (type)                    Output Shape         Param #     Connected to
==============================================================================
====================
input_5 (InputLayer)            [(None, 224, 224, 3) 0
______________________________________________________________________________
____________________
conv1_pad (ZeroPadding2D)       (None, 230, 230, 3)  0           input_5[0][0]
______________________________________________________________________________
____________________
conv1_conv (Conv2D)             (None, 112, 112, 64) 9472
conv1_pad[0][0]
______________________________________________________________________________
____________________
conv1_bn (BatchNormalization)   (None, 112, 112, 64) 256
conv1_conv[0][0]
______________________________________________________________________________
____________________
conv1_relu (Activation)         (None, 112, 112, 64) 0
conv1_bn[0][0]
______________________________________________________________________________
____________________
pool1_pad (ZeroPadding2D)       (None, 114, 114, 64) 0
conv1_relu[0][0]
______________________________________________________________________________
____________________
pool1_pool (MaxPooling2D)       (None, 56, 56, 64)   0
pool1_pad[0][0]
______________________________________________________________________________
____________________
conv2_block1_1_conv (Conv2D)    (None, 56, 56, 64)   4160
pool1_pool[0][0]
______________________________________________________________________________
____________________
conv2_block1_1_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block1_1_conv[0][0]
______________________________________________________________________________
____________________
conv2_block1_1_relu (Activation (None, 56, 56, 64)   0
conv2_block1_1_bn[0][0]
______________________________________________________________________________
____________________
conv2_block1_2_conv (Conv2D)    (None, 56, 56, 64)   36928
conv2_block1_1_relu[0][0]
```

```
__________________________________________________________________________________________________
conv2_block1_2_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_2_relu (Activation (None, 56, 56, 64)   0
conv2_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_0_conv (Conv2D)    (None, 56, 56, 256)  16640
pool1_pool[0][0]
__________________________________________________________________________________________________
conv2_block1_3_conv (Conv2D)    (None, 56, 56, 256)  16640
conv2_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block1_0_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_3_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block1_add (Add)          (None, 56, 56, 256)  0
conv2_block1_0_bn[0][0]

conv2_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block1_out (Activation)   (None, 56, 56, 256)  0
conv2_block1_add[0][0]
__________________________________________________________________________________________________
conv2_block2_1_conv (Conv2D)    (None, 56, 56, 64)   16448
conv2_block1_out[0][0]
__________________________________________________________________________________________________
conv2_block2_1_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_1_relu (Activation (None, 56, 56, 64)   0
conv2_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_2_conv (Conv2D)    (None, 56, 56, 64)   36928
conv2_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_2_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_2_relu (Activation (None, 56, 56, 64)   0
conv2_block2_2_bn[0][0]
__________________________________________________________________________________________________
```

```
conv2_block2_3_conv (Conv2D)    (None, 56, 56, 256)  16640
conv2_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block2_3_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block2_add (Add)          (None, 56, 56, 256)  0
conv2_block1_out[0][0]

conv2_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block2_out (Activation)   (None, 56, 56, 256)  0
conv2_block2_add[0][0]
__________________________________________________________________________________________________
conv2_block3_1_conv (Conv2D)    (None, 56, 56, 64)   16448
conv2_block2_out[0][0]
__________________________________________________________________________________________________
conv2_block3_1_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_1_relu (Activation (None, 56, 56, 64)   0
conv2_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_2_conv (Conv2D)    (None, 56, 56, 64)   36928
conv2_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv2_block3_2_bn (BatchNormali (None, 56, 56, 64)   256
conv2_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_2_relu (Activation (None, 56, 56, 64)   0
conv2_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_3_conv (Conv2D)    (None, 56, 56, 256)  16640
conv2_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv2_block3_3_bn (BatchNormali (None, 56, 56, 256)  1024
conv2_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv2_block3_add (Add)          (None, 56, 56, 256)  0
conv2_block2_out[0][0]

conv2_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv2_block3_out (Activation)   (None, 56, 56, 256)  0
conv2_block3_add[0][0]
__________________________________________________________________________________________________
```

```
conv3_block1_1_conv (Conv2D)    (None, 28, 28, 128)  32896
conv2_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block1_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_1_relu (Activation (None, 28, 28, 128)  0
conv3_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block1_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_2_relu (Activation (None, 28, 28, 128)  0
conv3_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_0_conv (Conv2D)    (None, 28, 28, 512)  131584
conv2_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block1_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block1_0_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block1_add (Add)          (None, 28, 28, 512)  0
conv3_block1_0_bn[0][0]

conv3_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block1_out (Activation)   (None, 28, 28, 512)  0
conv3_block1_add[0][0]
__________________________________________________________________________________________________
conv3_block2_1_conv (Conv2D)    (None, 28, 28, 128)  65664
conv3_block1_out[0][0]
__________________________________________________________________________________________________
conv3_block2_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_1_relu (Activation (None, 28, 28, 128)  0
conv3_block2_1_bn[0][0]
```

```
__________________________________________________________________________________________________
conv3_block2_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block2_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_2_relu (Activation (None, 28, 28, 128)  0
conv3_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block2_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block2_add (Add)          (None, 28, 28, 512)  0
conv3_block1_out[0][0]

conv3_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block2_out (Activation)   (None, 28, 28, 512)  0
conv3_block2_add[0][0]
__________________________________________________________________________________________________
conv3_block3_1_conv (Conv2D)    (None, 28, 28, 128)  65664
conv3_block2_out[0][0]
__________________________________________________________________________________________________
conv3_block3_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_1_relu (Activation (None, 28, 28, 128)  0
conv3_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block3_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_2_relu (Activation (None, 28, 28, 128)  0
conv3_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block3_2_relu[0][0]
__________________________________________________________________________________________________
```

```
conv3_block3_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block3_add (Add)          (None, 28, 28, 512)  0
conv3_block2_out[0][0]

conv3_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block3_out (Activation)   (None, 28, 28, 512)  0
conv3_block3_add[0][0]
__________________________________________________________________________________________________
conv3_block4_1_conv (Conv2D)    (None, 28, 28, 128)  65664
conv3_block3_out[0][0]
__________________________________________________________________________________________________
conv3_block4_1_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block4_1_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_1_relu (Activation (None, 28, 28, 128)  0
conv3_block4_1_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_2_conv (Conv2D)    (None, 28, 28, 128)  147584
conv3_block4_1_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_2_bn (BatchNormali (None, 28, 28, 128)  512
conv3_block4_2_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_2_relu (Activation (None, 28, 28, 128)  0
conv3_block4_2_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_3_conv (Conv2D)    (None, 28, 28, 512)  66048
conv3_block4_2_relu[0][0]
__________________________________________________________________________________________________
conv3_block4_3_bn (BatchNormali (None, 28, 28, 512)  2048
conv3_block4_3_conv[0][0]
__________________________________________________________________________________________________
conv3_block4_add (Add)          (None, 28, 28, 512)  0
conv3_block3_out[0][0]

conv3_block4_3_bn[0][0]
__________________________________________________________________________________________________
conv3_block4_out (Activation)   (None, 28, 28, 512)  0
conv3_block4_add[0][0]
__________________________________________________________________________________________________
conv4_block1_1_conv (Conv2D)    (None, 14, 14, 256)  131328
conv3_block4_out[0][0]
__________________________________________________________________________________________________
```

```
conv4_block1_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_1_relu (Activation (None, 14, 14, 256)  0
conv4_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block1_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_2_relu (Activation (None, 14, 14, 256)  0
conv4_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_0_conv (Conv2D)    (None, 14, 14, 1024) 525312
conv3_block4_out[0][0]
__________________________________________________________________________________________________
conv4_block1_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block1_0_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block1_add (Add)          (None, 14, 14, 1024) 0
conv4_block1_0_bn[0][0]

conv4_block1_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block1_out (Activation)   (None, 14, 14, 1024) 0
conv4_block1_add[0][0]
__________________________________________________________________________________________________
conv4_block2_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block1_out[0][0]
__________________________________________________________________________________________________
conv4_block2_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block2_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_1_relu (Activation (None, 14, 14, 256)  0
conv4_block2_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block2_1_relu[0][0]
```

```
__________________________________________________________________________________________________
conv4_block2_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block2_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_2_relu (Activation (None, 14, 14, 256)  0
conv4_block2_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block2_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block2_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block2_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block2_add (Add)          (None, 14, 14, 1024) 0
conv4_block1_out[0][0]

conv4_block2_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block2_out (Activation)   (None, 14, 14, 1024) 0
conv4_block2_add[0][0]
__________________________________________________________________________________________________
conv4_block3_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block2_out[0][0]
__________________________________________________________________________________________________
conv4_block3_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block3_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_1_relu (Activation (None, 14, 14, 256)  0
conv4_block3_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block3_2_relu (Activation (None, 14, 14, 256)  0
conv4_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block3_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block3_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block3_3_conv[0][0]
__________________________________________________________________________________________________
```

```
conv4_block3_add (Add)          (None, 14, 14, 1024) 0
conv4_block2_out[0][0]

conv4_block3_3_bn[0][0]
________________________________________________________________________________
____________________
conv4_block3_out (Activation)   (None, 14, 14, 1024) 0
conv4_block3_add[0][0]
________________________________________________________________________________
____________________
conv4_block4_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block3_out[0][0]
________________________________________________________________________________
____________________
conv4_block4_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block4_1_conv[0][0]
________________________________________________________________________________
____________________
conv4_block4_1_relu (Activation (None, 14, 14, 256)  0
conv4_block4_1_bn[0][0]
________________________________________________________________________________
____________________
conv4_block4_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block4_1_relu[0][0]
________________________________________________________________________________
____________________
conv4_block4_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block4_2_conv[0][0]
________________________________________________________________________________
____________________
conv4_block4_2_relu (Activation (None, 14, 14, 256)  0
conv4_block4_2_bn[0][0]
________________________________________________________________________________
____________________
conv4_block4_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block4_2_relu[0][0]
________________________________________________________________________________
____________________
conv4_block4_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block4_3_conv[0][0]
________________________________________________________________________________
____________________
conv4_block4_add (Add)          (None, 14, 14, 1024) 0
conv4_block3_out[0][0]

conv4_block4_3_bn[0][0]
________________________________________________________________________________
____________________
conv4_block4_out (Activation)   (None, 14, 14, 1024) 0
conv4_block4_add[0][0]
________________________________________________________________________________
____________________
conv4_block5_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block4_out[0][0]
________________________________________________________________________________
____________________
conv4_block5_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block5_1_conv[0][0]
________________________________________________________________________________
____________________
```

```
conv4_block5_1_relu (Activation (None, 14, 14, 256)  0
conv4_block5_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block5_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block5_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block5_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_2_relu (Activation (None, 14, 14, 256)  0
conv4_block5_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block5_2_relu[0][0]
__________________________________________________________________________________________________
conv4_block5_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block5_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block5_add (Add)          (None, 14, 14, 1024) 0
conv4_block4_out[0][0]

conv4_block5_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block5_out (Activation)   (None, 14, 14, 1024) 0
conv4_block5_add[0][0]
__________________________________________________________________________________________________
conv4_block6_1_conv (Conv2D)    (None, 14, 14, 256)  262400
conv4_block5_out[0][0]
__________________________________________________________________________________________________
conv4_block6_1_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block6_1_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_1_relu (Activation (None, 14, 14, 256)  0
conv4_block6_1_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_2_conv (Conv2D)    (None, 14, 14, 256)  590080
conv4_block6_1_relu[0][0]
__________________________________________________________________________________________________
conv4_block6_2_bn (BatchNormali (None, 14, 14, 256)  1024
conv4_block6_2_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_2_relu (Activation (None, 14, 14, 256)  0
conv4_block6_2_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_3_conv (Conv2D)    (None, 14, 14, 1024) 263168
conv4_block6_2_relu[0][0]
```

```
__________________________________________________________________________________________________
conv4_block6_3_bn (BatchNormali (None, 14, 14, 1024) 4096
conv4_block6_3_conv[0][0]
__________________________________________________________________________________________________
conv4_block6_add (Add)          (None, 14, 14, 1024) 0
conv4_block5_out[0][0]

conv4_block6_3_bn[0][0]
__________________________________________________________________________________________________
conv4_block6_out (Activation)   (None, 14, 14, 1024) 0
conv4_block6_add[0][0]
__________________________________________________________________________________________________
conv5_block1_1_conv (Conv2D)    (None, 7, 7, 512)    524800
conv4_block6_out[0][0]
__________________________________________________________________________________________________
conv5_block1_1_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block1_1_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_1_relu (Activation (None, 7, 7, 512)    0
conv5_block1_1_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_2_conv (Conv2D)    (None, 7, 7, 512)    2359808
conv5_block1_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block1_2_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block1_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_2_relu (Activation (None, 7, 7, 512)    0
conv5_block1_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block1_0_conv (Conv2D)    (None, 7, 7, 2048)   2099200
conv4_block6_out[0][0]
__________________________________________________________________________________________________
conv5_block1_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624
conv5_block1_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block1_0_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block1_0_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_3_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block1_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block1_add (Add)          (None, 7, 7, 2048)   0
conv5_block1_0_bn[0][0]

conv5_block1_3_bn[0][0]
```

```
____________________________________________________________________________________________________
conv5_block1_out (Activation)   (None, 7, 7, 2048)   0
conv5_block1_add[0][0]
____________________________________________________________________________________________________
conv5_block2_1_conv (Conv2D)    (None, 7, 7, 512)    1049088
conv5_block1_out[0][0]
____________________________________________________________________________________________________
conv5_block2_1_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block2_1_conv[0][0]
____________________________________________________________________________________________________
conv5_block2_1_relu (Activation (None, 7, 7, 512)    0
conv5_block2_1_bn[0][0]
____________________________________________________________________________________________________
conv5_block2_2_conv (Conv2D)    (None, 7, 7, 512)    2359808
conv5_block2_1_relu[0][0]
____________________________________________________________________________________________________
conv5_block2_2_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block2_2_conv[0][0]
____________________________________________________________________________________________________
conv5_block2_2_relu (Activation (None, 7, 7, 512)    0
conv5_block2_2_bn[0][0]
____________________________________________________________________________________________________
conv5_block2_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624
conv5_block2_2_relu[0][0]
____________________________________________________________________________________________________
conv5_block2_3_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block2_3_conv[0][0]
____________________________________________________________________________________________________
conv5_block2_add (Add)          (None, 7, 7, 2048)   0
conv5_block1_out[0][0]

conv5_block2_3_bn[0][0]
____________________________________________________________________________________________________
conv5_block2_out (Activation)   (None, 7, 7, 2048)   0
conv5_block2_add[0][0]
____________________________________________________________________________________________________
conv5_block3_1_conv (Conv2D)    (None, 7, 7, 512)    1049088
conv5_block2_out[0][0]
____________________________________________________________________________________________________
conv5_block3_1_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block3_1_conv[0][0]
____________________________________________________________________________________________________
conv5_block3_1_relu (Activation (None, 7, 7, 512)    0
conv5_block3_1_bn[0][0]
____________________________________________________________________________________________________
```

```
conv5_block3_2_conv (Conv2D)    (None, 7, 7, 512)    2359808
conv5_block3_1_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_2_bn (BatchNormali (None, 7, 7, 512)    2048
conv5_block3_2_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_2_relu (Activation (None, 7, 7, 512)    0
conv5_block3_2_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_3_conv (Conv2D)    (None, 7, 7, 2048)   1050624
conv5_block3_2_relu[0][0]
__________________________________________________________________________________________________
conv5_block3_3_bn (BatchNormali (None, 7, 7, 2048)   8192
conv5_block3_3_conv[0][0]
__________________________________________________________________________________________________
conv5_block3_add (Add)          (None, 7, 7, 2048)   0
conv5_block2_out[0][0]

conv5_block3_3_bn[0][0]
__________________________________________________________________________________________________
conv5_block3_out (Activation)   (None, 7, 7, 2048)   0
conv5_block3_add[0][0]
__________________________________________________________________________________________________
flatten_5 (Flatten)             (None, 100352)       0
conv5_block3_out[0][0]
__________________________________________________________________________________________________
dense_14 (Dense)                (None, 197)          19769541
flatten_5[0][0]
==================================================================================================
Total params: 43,357,253
Trainable params: 19,769,541
Non-trainable params: 23,587,712
__________________________________________________________________________________________________
```

3. Training [Forward pass and Backpropagation]

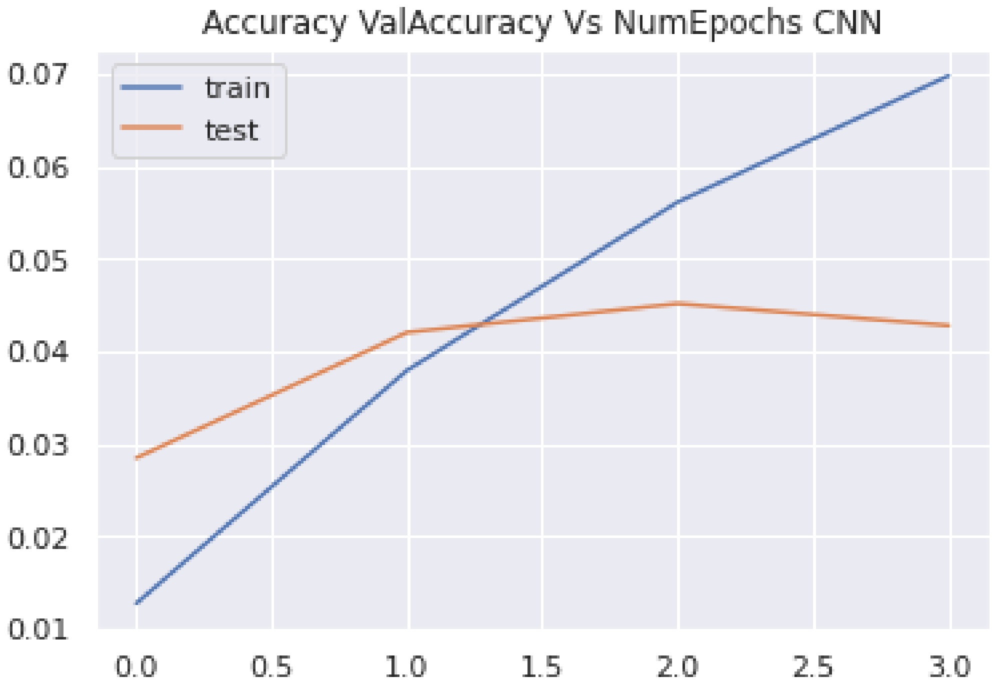
```
#Compile with optimizer
resnet.compile(optimizer = Adam(learning_rate=0.001), loss = 'categorical_crossentropy', metrics = ['accuracy'])

early = EarlyStopping(monitor='val_accuracy',min_delta=0.01,patience=2,verbose=1,mode='auto')

#Training
```

```
resnet_classifier = resnet.fit_generator(train_generator,epochs =30, validatio
n_data = validation_generator, callbacks = [early]  )
```

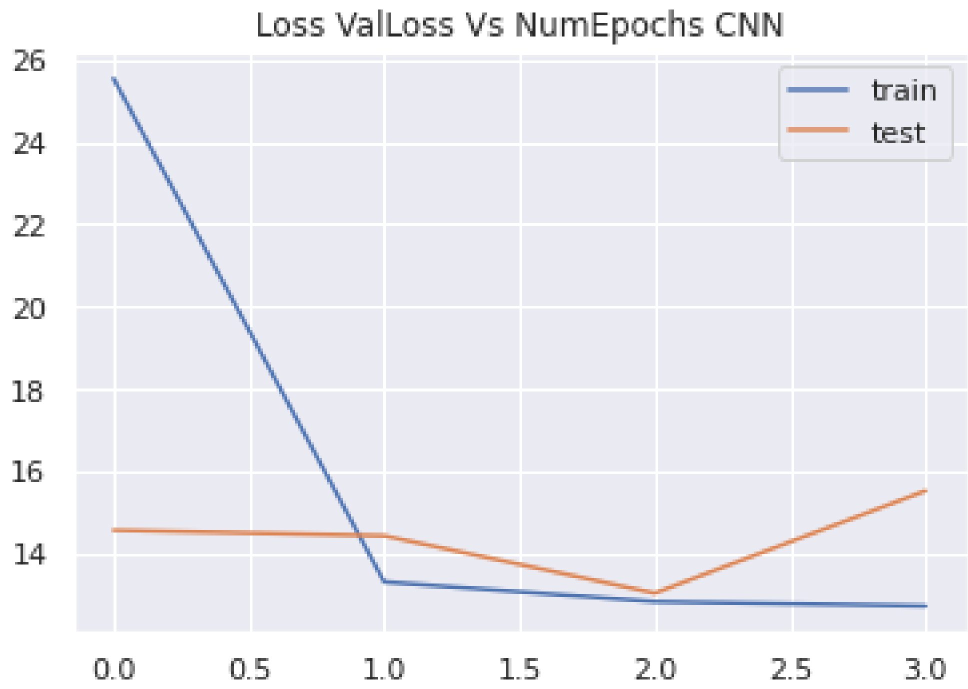
```
Epoch 1/30
255/255 [==============================] - 188s 726ms/step - loss: 25.5408 -
accuracy: 0.0128 - val_loss: 14.5744 - val_accuracy: 0.0285
Epoch 2/30
255/255 [==============================] - 182s 716ms/step - loss: 13.3197 -
accuracy: 0.0379 - val_loss: 14.4397 - val_accuracy: 0.0420
Epoch 3/30
255/255 [==============================] - 183s 718ms/step - loss: 12.8443 -
accuracy: 0.0561 - val_loss: 13.0493 - val_accuracy: 0.0451
Epoch 4/30
255/255 [==============================] - 183s 718ms/step - loss: 12.7389 -
accuracy: 0.0697 - val_loss: 15.5320 - val_accuracy: 0.0428
        Epoch 00004: early stopping
```

4. Plot Accuracy and Loss

```
train_loss = resnet_classifier.history['loss']
val_loss   = resnet_classifier.history['val_loss']

xc = resnet_classifier.epoch
plt.title("Accuracy ValAccuracy Vs NumEpochs CNN")
plt.plot(xc,resnet_classifier.history['accuracy'], label='train')
plt.plot(xc,resnet_classifier.history['val_accuracy'], label='test')
plt.legend()
plt.show()

plt.figure()
plt.title("Loss ValLoss Vs NumEpochs CNN")
plt.plot(xc, train_loss,label='train')
plt.plot(xc, val_loss,label='test')
plt.legend()
plt.show
```



As seen in the graph, training set accuracy continue to increase with each epoch. However, validation set accuracy doesn't change after few epochs.



As seen above, training dataset loss shows a sharp drop after initial few epochs and then becomes constant. Validation dataset loss shows a marginal drop after few epochs and then starts to increase again.

5. Evaluation

```
train_acc = resnet.evaluate_generator(train_generator,steps = int(train_genera
tor.samples/BATCH_SIZE))
val_acc = resnet.evaluate_generator(validation_generator, steps = int(validati
on_generator.samples/BATCH_SIZE))

print(train_acc[1])
print(val_acc[1])
```

```
0.07221949100494385
0.04282868653535843
```

As seen, this model gives a very low training and validation accuracy of 7% and 4% respectively.

6. Adding result to dataframe for comparison

```
#Adding Performance metrics of Custom ResNet50 to the list
tempResultsDf = pd.DataFrame({'Model':['ResNet Custom FC'],  'Train_Accuracy':
 train_acc[1],'Test_Accuracy': val_acc[1]})
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf = resultsDf[['Model', 'Train_Accuracy','Test_Accuracy']]
resultsDf
```

	Model	Train_Accuracy	Test_Accuracy
0	CNN	0.359129	0.158616
0	ResNet50	0.005290	0.005229
0	VGG16	0.938607	0.563994
0	ResNet Custom FC	0.072219	0.042829

7. Save model for future use

```
resnet.save('./resnet.h5')

resnet.save_weights('./resnet_weights.h5')
```

E. InceptionResNetV2

1. Creating the model

```
base_model = InceptionResNetV2(include_top=False, input_shape = INPUT_SIZE)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_resnet_v2/inception_resnet_v2_weights_tf_dim_ordering_tf_kernels_notop.h5
219062272/219055592 [==============================] - 2s 0us/step
219070464/219055592 [==============================] - 2s 0us/step
```

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(197, activation='softmax')
])
```

2. Summary of model

```
classification_model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_resnet_v2 (Functio (None, 5, 5, 1536)        54336736
_________________________________________________________________
global_average_pooling2d (Gl (None, 1536)              0
_________________________________________________________________
dense_15 (Dense)             (None, 128)               196736
_________________________________________________________________
batch_normalization_203 (Bat (None, 128)               512
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_16 (Dense)             (None, 197)               25413
=================================================================
Total params: 54,559,397
Trainable params: 54,498,597
Non-trainable params: 60,800
```

3. Define optimizer

```
lr=0.001
classification_model.compile(loss='categorical_crossentropy', optimizer=Adam(l
r=lr), metrics=['accuracy'])
```

4. Early stopping and Model Checkpoint

```
patience = 1
stop_patience = 3
factor = 0.5

callbacks = [
    tf.keras.callbacks.ModelCheckpoint("classify_model.h5", save_best_only=Tru
e, verbose = 0),
    tf.keras.callbacks.EarlyStopping(patience=stop_patience, monitor='val_loss
', verbose=1),
    tf.keras.callbacks.ReduceLROnPlateau(monitor='val_loss', factor=factor, pa
tience=patience, verbose=1)
]
```

5. Training [Forward pass and Backpropagation]

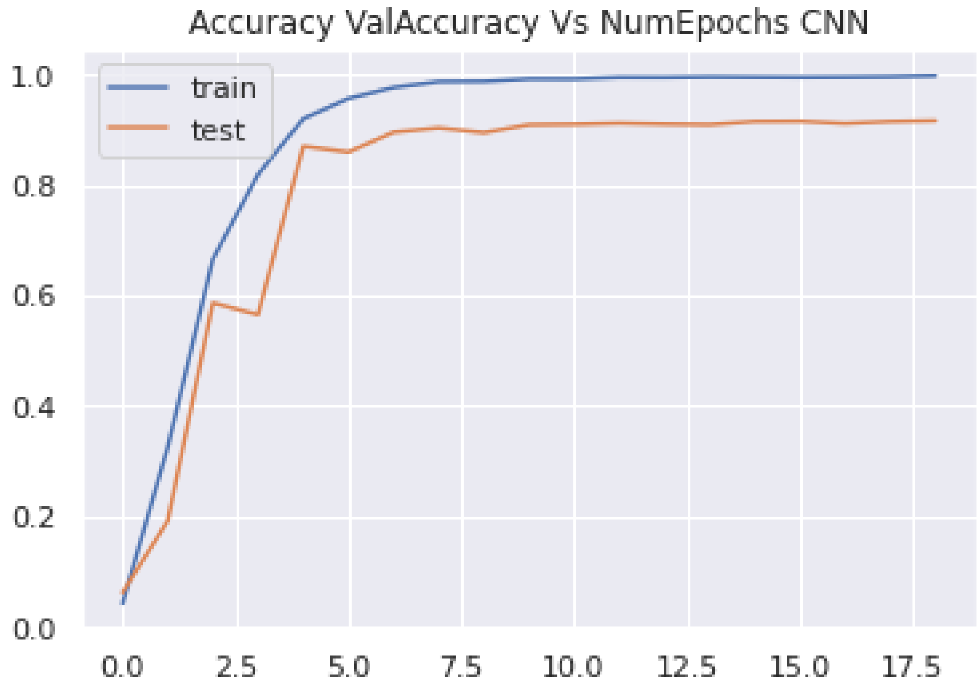
```
epochs = 30
history = classification_model.fit(train_generator, validation_data=validation
_generator, epochs=epochs, callbacks=callbacks, verbose=1)
```

```
Epoch 1/30
255/255 [==============================] - 219s 774ms/step - loss: 5.0064 -
accuracy: 0.0424 - val_loss: 4.8299 - val_accuracy: 0.0622
Epoch 2/30
255/255 [==============================] - 193s 756ms/step - loss: 3.0798 -
accuracy: 0.3256 - val_loss: 3.7128 - val_accuracy: 0.1919
Epoch 3/30
255/255 [==============================] - 192s 754ms/step - loss: 1.4914 -
accuracy: 0.6661 - val_loss: 1.6520 - val_accuracy: 0.5869
Epoch 4/30
255/255 [==============================] - 193s 755ms/step - loss: 0.7913 -
accuracy: 0.8196 - val_loss: 1.7750 - val_accuracy: 0.5655

Epoch 00004: ReduceLROnPlateau reducing learning rate to
0.0005000000237487257.
Epoch 5/30
255/255 [==============================] - 190s 745ms/step - loss: 0.3456 -
accuracy: 0.9209 - val_loss: 0.5200 - val_accuracy: 0.8710
Epoch 6/30
255/255 [==============================] - 193s 753ms/step - loss: 0.2035 -
accuracy: 0.9573 - val_loss: 0.5306 - val_accuracy: 0.8605
```

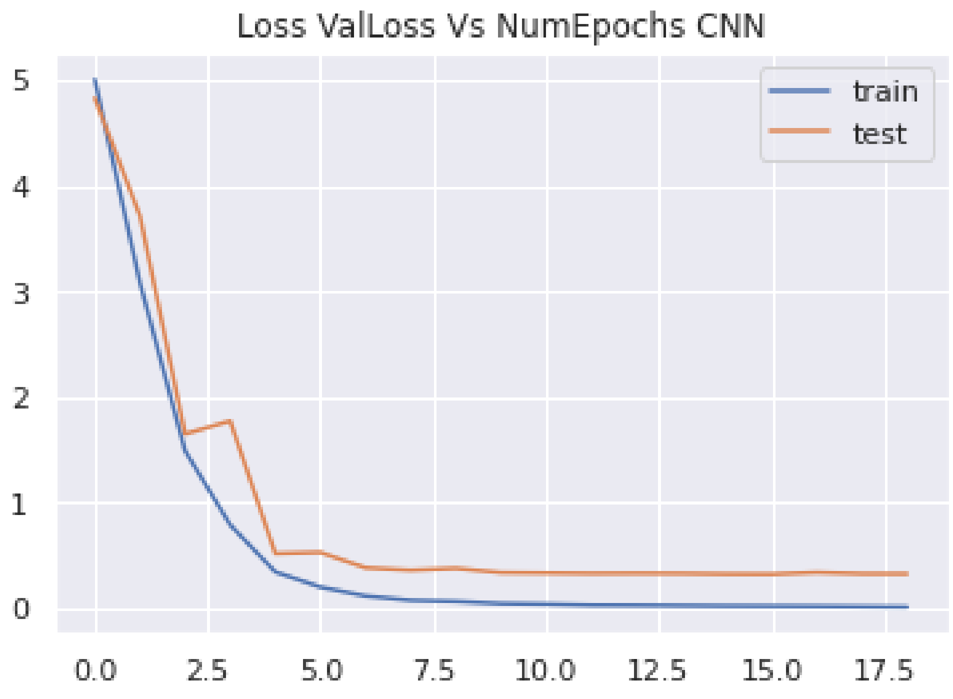
```
Epoch 00006: ReduceLROnPlateau reducing learning rate to
0.0002500000118743628.
Epoch 7/30
255/255 [==============================] - 191s 748ms/step - loss: 0.1194 -
accuracy: 0.9773 - val_loss: 0.3848 - val_accuracy: 0.8965
Epoch 8/30
255/255 [==============================] - 193s 755ms/step - loss: 0.0785 -
accuracy: 0.9877 - val_loss: 0.3624 - val_accuracy: 0.9042
Epoch 9/30
255/255 [==============================] - 193s 756ms/step - loss: 0.0683 -
accuracy: 0.9882 - val_loss: 0.3795 - val_accuracy: 0.8957

Epoch 00009: ReduceLROnPlateau reducing learning rate to
0.0001250000059371814.
Epoch 10/30
161/255 [=================>...........] - ETA: 50s - loss: 0.0471 - accuracy:
0.9930
```

6. Plot Accuracy and Loss for Training and Validation

```
train_loss = history.history['loss']
val_loss   = history.history['val_loss']

xc = history.epoch
plt.title("Accuracy ValAccuracy Vs NumEpochs CNN")
plt.plot(xc,history.history['accuracy'], label='train')
plt.plot(xc,history.history['val_accuracy'], label='test')
plt.legend()
plt.show()

plt.figure()
plt.title("Loss ValLoss Vs NumEpochs CNN")
plt.plot(xc, train_loss,label='train')
plt.plot(xc, val_loss,label='test')
plt.legend()
plt.show
```



As seen from the graph above, both training and validation accuracy continue to increase for initial epochs and then becomes constant after reaching near 100%. This shows that we could have probably trained model for lesser number of epochs.



As seen from the graph above, both training and validation loss continue to decrease for initial epochs and then becomes constant after reaching near 0. This shows that we could have probably trained model for lesser number of epochs.

7. Evaluation

```
train_acc = classification_model.evaluate_generator(train_generator,steps = in
t(train_generator.samples/BATCH_SIZE))
val_acc = classification_model.evaluate_generator(validation_generator, steps
= int(validation_generator.samples/BATCH_SIZE))

print(train_acc[1])
print(val_acc[1])
```

```
0.9977854490280151
0.9172061681747437
```

Thus model works really great and shows a near perfect accuracy of 99.77% for training dataset, and very high accuracy of 91.7% for validation dataset.

8. Adding result to dataframe for comparison

```
#Adding Performance metrics of InceptionResNetv2 to the list
tempResultsDf = pd.DataFrame({'Model':['InceptionResNetv2'],  'Train_Accuracy'
: train_acc[1],'Test_Accuracy': val_acc[1]})
resultsDf = pd.concat([resultsDf, tempResultsDf])
resultsDf = resultsDf[['Model', 'Train_Accuracy','Test_Accuracy']]
resultsDf
```

	Model	Train_Accuracy	Test_Accuracy
0	CNN	0.359129	0.158616
0	ResNet50	0.005290	0.005229
0	VGG16	0.938607	0.563994
0	ResNet Custom FC	0.072219	0.042829
0	InceptionResNetv2	0.997785	0.917206

Comparing Models

```
resultsDf
```

	Model	Train_Accuracy	Test_Accuracy
0	CNN	0.359129	0.158616
0	ResNet50	0.005290	0.005229
0	VGG16	0.938607	0.563994
0	ResNet Custom FC	0.072219	0.042829
0	InceptionResNetv2	0.997785	0.917206

As seen from the table above, we tried different models for this classification problem.

InceptionResNetv2 **gives the best accuracy. Therefore, it is our final selected model.**

```
final_model = classification_model
```

- **Pickle model for future use**

```
final_model.save('./final_model.h5')
```

Predictions

Let us use final model to predict some test car images

```
final_model = keras.models.load_model('final_model.h5')
```

```
from google.colab import files

uploaded = files.upload()
```

Saving test4.jpg to test4.jpg

```
path = 'test1.jpg'
img = cv2.imread( path )
plt.grid(False)
plt.imshow(img)
```

```
from tensorflow.keras.utils import img_to_array, load_img
import cv2
img = cv2.resize(img, (224,224),)
img.shape
```

(224, 224, 3)

```
pixels = img.astype('float32')
pixels /= 255.0
print(pixels.shape)
```

(224, 224, 3)

```
#Expanding the dimensions of the numpy array to match the dimension expected by
predict method
pixels = np.expand_dims(pixels, axis=0)
print(pixels.shape)
```

(1, 224, 224, 3)

```
prediction = final_model.predict(pixels)
prediction = np.argmax(prediction, axis = 1)
print(prediction)
```

[135]

```
predicted_label = car_names[car_names['Class'] == prediction[0]]
print(predicted_label)
```

	CarLabel	Class
134	Hyundai Elantra Sedan 2007	135

Thus our model is able to make correct prediction.

Next Steps:

For next milestone, we will work on following steps:

1) Try to fine tune our selected model to reduce variance.
2) Design a clickable UI which can automate tasks performed under milestone 1
3) Design a clickable UI which can automate tasks performed under milestone 2
4) Design a clickable UI based interface which can allow the user to browse & input the image, output the class and the bounding box or mask
5) Create final report

Final Model Tuning

For final model, we got training accuracy of 99.77% and validation accuracy of 91.72%. We tried different approaches to reduce variation.

1) Added L2 regularization

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu', kernel_regularizer='l2'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.20),
    tf.keras.layers.Dense(197, activation='softmax')
])
```

Epoch 1/20
255/255 [==============================] - 12463s 49s/step - loss: 5.7889 - accuracy: 0.0662 - val_loss: 4.9012 - val_accuracy: 0.0751
Epoch 2/20

```
255/255 [==============================] - 162s 635ms/step - loss: 3.0829 - accuracy: 0.3622 - val_loss: 3.2536 - val_accuracy: 0.3043
Epoch 3/20
255/255 [==============================] - 162s 633ms/step - loss: 1.7808 - accuracy: 0.6287 - val_loss: 2.2073 - val_accuracy: 0.5366
Epoch 4/20
255/255 [==============================] - 162s 634ms/step - loss: 1.1926 - accuracy: 0.7564 - val_loss: 1.9011 - val_accuracy: 0.5601
Epoch 5/20
255/255 [==============================] - 161s 631ms/step - loss: 0.8764 - accuracy: 0.8237 - val_loss: 1.8182 - val_accuracy: 0.5788
Epoch 6/20
255/255 [==============================] - 162s 633ms/step - loss: 0.7422 - accuracy: 0.8556 - val_loss: 1.6738 - val_accuracy: 0.6272
Epoch 7/20
255/255 [==============================] - 162s 634ms/step - loss: 0.6438 - accuracy: 0.8767 - val_loss: 1.4621 - val_accuracy: 0.6658
Epoch 8/20
255/255 [==============================] - 161s 631ms/step - loss: 0.5826 - accuracy: 0.8891 - val_loss: 1.6377 - val_accuracy: 0.6319

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 9/20
255/255 [==============================] - 160s 628ms/step - loss: 0.3303 - accuracy: 0.9486 - val_loss: 0.6863 - val_accuracy: 0.8447
Epoch 10/20
255/255 [==============================] - 161s 631ms/step - loss: 0.2142 - accuracy: 0.9650 - val_loss: 0.6510 - val_accuracy: 0.8450
Epoch 11/20
255/255 [==============================] - 161s 633ms/step - loss: 0.1775 - accuracy: 0.9730 - val_loss: 0.7043 - val_accuracy: 0.8402

Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 12/20
255/255 [==============================] - 161s 629ms/step - loss: 0.1347 - accuracy: 0.9832 - val_loss: 0.8370 - val_accuracy: 0.8064

Epoch 00012: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 13/20
255/255 [==============================] - 160s 628ms/step - loss: 0.0857 - accuracy: 0.9912 - val_loss: 0.3929 - val_accuracy: 0.9052
Epoch 14/20
255/255 [==============================] - 162s 635ms/step - loss: 0.0709 - accuracy: 0.9930 - val_loss: 0.3856 - val_accuracy: 0.9072
Epoch 15/20
255/255 [==============================] - 162s 633ms/step - loss: 0.0605 - accuracy: 0.9932 - val_loss: 0.4006 - val_accuracy: 0.9032

Epoch 00015: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 16/20
255/255 [==============================] - 160s 629ms/step - loss: 0.0483 - accuracy: 0.9958 - val_loss: 0.3652 - val_accuracy: 0.9111
Epoch 17/20
255/255 [==============================] - 163s 638ms/step - loss: 0.0440 - accuracy: 0.9958 - val_loss: 0.3718 - val_accuracy: 0.9092

Epoch 00017: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 18/20
255/255 [==============================] - 161s 633ms/step - loss: 0.0387 - accuracy: 0.9963 - val_loss: 0.3618 - val_accuracy: 0.9113
Epoch 19/20
```

255/255 [==============================] - 162s 636ms/step - loss: 0.0336 - accuracy: 0.9974 - val_loss: 0.3576 - val_accuracy: 0.9133
Epoch 20/20
255/255 [==============================] - 162s 633ms/step - loss: 0.0337 - accuracy: 0.9968 - val_loss: 0.3616 - val_accuracy: 0.9124

Epoch 00020: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
CodeText

```
print(train_acc[1])
print(val_acc[1])
```

```
0.998031497001648

0.9073705077171326
```

2) Added 1 additional layer

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.20),
    tf.keras.layers.Dense(197, activation='softmax')
])
```

```
Epoch 1/20
255/255 [==============================] - 181s 650ms/step - loss: 1.8494 - accuracy: 0.7157 - val_loss: 1.6354 - val_accuracy: 0.7091
Epoch 2/20
255/255 [==============================] - 164s 641ms/step - loss: 0.4688 - accuracy: 0.9128 - val_loss: 1.4693 - val_accuracy: 0.6673
Epoch 3/20
255/255 [==============================] - 163s 640ms/step - loss: 0.3327 - accuracy: 0.9261 - val_loss: 1.1654 - val_accuracy: 0.7326
Epoch 4/20
255/255 [==============================] - 162s 635ms/step - loss: 0.2844 - accuracy: 0.9278 - val_loss: 1.2723 - val_accuracy: 0.7077

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 5/20
255/255 [==============================] - 160s 628ms/step - loss: 0.1142 - accuracy: 0.9691 - val_loss: 0.4884 - val_accuracy: 0.8800
Epoch 6/20
255/255 [==============================] - 161s 631ms/step - loss: 0.0572 - accuracy: 0.9872 - val_loss: 0.4386 - val_accuracy: 0.8929
Epoch 7/20
255/255 [==============================] - 161s 631ms/step - loss: 0.0561 - accuracy: 0.9851 - val_loss: 0.4896 - val_accuracy: 0.8857

Epoch 00007: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 8/20
```

```
255/255 [==============================] - 160s 627ms/step - loss: 0.0338 - accuracy: 0.9917 - val_loss: 0.3944 - val_accuracy: 0.9072
Epoch 9/20
255/255 [==============================] - 161s 631ms/step - loss: 0.0240 - accuracy: 0.9939 - val_loss: 0.4032 - val_accuracy: 0.9047

Epoch 00009: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 10/20
255/255 [==============================] - 160s 627ms/step - loss: 0.0202 - accuracy: 0.9956 - val_loss: 0.3805 - val_accuracy: 0.9107
Epoch 11/20
255/255 [==============================] - 161s 632ms/step - loss: 0.0148 - accuracy: 0.9967 - val_loss: 0.3840 - val_accuracy: 0.9108

Epoch 00011: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 12/20
255/255 [==============================] - 160s 627ms/step - loss: 0.0134 - accuracy: 0.9968 - val_loss: 0.3789 - val_accuracy: 0.9138
Epoch 13/20
255/255 [==============================] - 162s 634ms/step - loss: 0.0141 - accuracy: 0.9966 - val_loss: 0.3768 - val_accuracy: 0.9139
Epoch 14/20
255/255 [==============================] - 162s 634ms/step - loss: 0.0122 - accuracy: 0.9969 - val_loss: 0.3785 - val_accuracy: 0.9159

Epoch 00014: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 15/20
255/255 [==============================] - 160s 628ms/step - loss: 0.0112 - accuracy: 0.9972 - val_loss: 0.3782 - val_accuracy: 0.9152

Epoch 00015: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
Epoch 16/20
255/255 [==============================] - 160s 629ms/step - loss: 0.0112 - accuracy: 0.9971 - val_loss: 0.3782 - val_accuracy: 0.9148

Epoch 00016: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.

      Epoch 00016: early stopping
```

```
print(train_acc[1])
print(val_acc[1])
```

```
0.9985235929489136

0.9147161245346069
```

3) Added 2 additional layers

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(256, activation='relu'),
```

```
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.20),
    tf.keras.layers.Dense(197, activation='softmax')
])
```

```
Epoch 1/25
255/255 [==============================] - 190s 745ms/step - loss: 0.0982 - accuracy: 0.9819 - val_loss: 0.3994 - val_accuracy: 0.8879
Epoch 2/25
255/255 [==============================] - 190s 746ms/step - loss: 0.0859 - accuracy: 0.9866 - val_loss: 0.3909 - val_accuracy: 0.8964
Epoch 3/25
255/255 [==============================] - 191s 747ms/step - loss: 0.0760 - accuracy: 0.9888 - val_loss: 0.4057 - val_accuracy: 0.8889

Epoch 00003: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 4/25
255/255 [==============================] - 189s 739ms/step - loss: 0.0626 - accuracy: 0.9929 - val_loss: 0.3721 - val_accuracy: 0.8978
Epoch 5/25
255/255 [==============================] - 189s 742ms/step - loss: 0.0544 - accuracy: 0.9930 - val_loss: 0.3717 - val_accuracy: 0.8996
Epoch 6/25
255/255 [==============================] - 190s 742ms/step - loss: 0.0520 - accuracy: 0.9939 - val_loss: 0.3848 - val_accuracy: 0.8965

Epoch 00006: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 7/25
255/255 [==============================] - 189s 741ms/step - loss: 0.0447 - accuracy: 0.9948 - val_loss: 0.3703 - val_accuracy: 0.8994
Epoch 8/25
255/255 [==============================] - 191s 747ms/step - loss: 0.0435 - accuracy: 0.9951 - val_loss: 0.3731 - val_accuracy: 0.8990

Epoch 00008: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 9/25
255/255 [==============================] - 187s 735ms/step - loss: 0.0411 - accuracy: 0.9957 - val_loss: 0.3665 - val_accuracy: 0.8994
Epoch 10/25
255/255 [==============================] - 189s 742ms/step - loss: 0.0400 - accuracy: 0.9962 - val_loss: 0.3626 - val_accuracy: 0.9005
Epoch 11/25
255/255 [==============================] - 192s 754ms/step - loss: 0.0383 - accuracy: 0.9942 - val_loss: 0.3633 - val_accuracy: 0.9032

Epoch 00011: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
Epoch 12/25
255/255 [==============================] - 190s 746ms/step - loss: 0.0361 - accuracy: 0.9962 - val_loss: 0.3610 - val_accuracy: 0.9015
Epoch 13/25
255/255 [==============================] - 191s 748ms/step - loss: 0.0321 - accuracy: 0.9972 - val_loss: 0.3613 - val_accuracy: 0.9014

Epoch 00013: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
Epoch 14/25
```

```
255/255 [==============================] - 189s 740ms/step - loss: 0.0338 - accuracy: 0.9972 - val_loss: 0.3619 - val_accuracy: 0.9014

Epoch 00014: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.
Epoch 15/25
255/255 [==============================] - 188s 737ms/step - loss: 0.0320 - accuracy: 0.9967 - val_loss: 0.3620 - val_accuracy: 0.9016

Epoch 00015: ReduceLROnPlateau reducing learning rate to 1.9531250927684596e-06.

Epoch 00015: early stopping
```

```
print(train_acc[1])
print(val_acc[1])
```

```
0.9976624250411987

0.9016434550285339
```

4) Added 1 additional layer after Batch Normalization

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.Dropout(0.20),
    tf.keras.layers.Dense(197, activation='softmax')
])
```

```
Epoch 1/25
255/255 [==============================] - 212s 755ms/step - loss: 2.1494 - accuracy: 0.5803 - val_loss: 1.3532 - val_accuracy: 0.6718
Epoch 2/25
255/255 [==============================] - 191s 751ms/step - loss: 0.5785 - accuracy: 0.8374 - val_loss: 1.2510 - val_accuracy: 0.6851
Epoch 3/25
255/255 [==============================] - 190s 745ms/step - loss: 0.4462 - accuracy: 0.8680 - val_loss: 0.9357 - val_accuracy: 0.7510
Epoch 4/25
255/255 [==============================] - 190s 746ms/step - loss: 0.3684 - accuracy: 0.8867 - val_loss: 1.0145 - val_accuracy: 0.7447

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 5/25
255/255 [==============================] - 189s 740ms/step - loss: 0.1473 - accuracy: 0.9552 - val_loss: 0.4592 - val_accuracy: 0.8785
Epoch 6/25
255/255 [==============================] - 190s 745ms/step - loss: 0.1031 - accuracy: 0.9691 - val_loss: 0.5824 - val_accuracy: 0.8595

Epoch 00006: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
```

```
Epoch 7/25
255/255 [==============================] - 189s 740ms/step - loss: 0.0659 - accuracy: 0.9810 - val_loss: 0.4346 - val_accuracy: 0.8896
Epoch 8/25
255/255 [==============================] - 190s 745ms/step - loss: 0.0525 - accuracy: 0.9853 - val_loss: 0.4293 - val_accuracy: 0.8947
Epoch 9/25
255/255 [==============================] - 190s 746ms/step - loss: 0.0439 - accuracy: 0.9896 - val_loss: 0.4344 - val_accuracy: 0.8938

Epoch 00009: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 10/25
255/255 [==============================] - 191s 748ms/step - loss: 0.0304 - accuracy: 0.9924 - val_loss: 0.4108 - val_accuracy: 0.8989
Epoch 11/25
255/255 [==============================] - 191s 747ms/step - loss: 0.0302 - accuracy: 0.9932 - val_loss: 0.4131 - val_accuracy: 0.9004

Epoch 00011: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 12/25
255/255 [==============================] - 190s 744ms/step - loss: 0.0263 - accuracy: 0.9942 - val_loss: 0.4059 - val_accuracy: 0.9032
Epoch 13/25
255/255 [==============================] - 192s 751ms/step - loss: 0.0252 - accuracy: 0.9941 - val_loss: 0.4084 - val_accuracy: 0.9036

Epoch 00013: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 14/25
255/255 [==============================] - 189s 743ms/step - loss: 0.0223 - accuracy: 0.9942 - val_loss: 0.4065 - val_accuracy: 0.9051

Epoch 00014: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
Epoch 15/25
255/255 [==============================] - 189s 741ms/step - loss: 0.0188 - accuracy: 0.9959 - val_loss: 0.4067 - val_accuracy: 0.9047

Epoch 00015: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.

      Epoch 00015: early stopping
```

```
print(train_acc[1])
print(val_acc[1])
```

```
0.9972932934761047

0.904631495475769
```

5) Keep 1 layer and increase dropout to 25%

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.BatchNormalization(),
```

```
    tf.keras.layers.Dropout(0.25),
    tf.keras.layers.Dense(197, activation='softmax')
])
```

```
print(train_acc[1])
print(val_acc[1])
```

```
0.9979084730148315

0.9109810590744019
```

6) Keep 1 layer and decrease dropout to 15%

```
Epoch 1/20
255/255 [==============================] - 214s 759ms/step - loss: 1.0674 - accuracy: 0.8652 - val_loss: 1.1154 - val_accuracy: 0.8023
Epoch 2/20
255/255 [==============================] - 189s 742ms/step - loss: 0.2601 - accuracy: 0.9494 - val_loss: 0.9886 - val_accuracy: 0.7881
Epoch 3/20
255/255 [==============================] - 189s 739ms/step - loss: 0.1824 - accuracy: 0.9637 - val_loss: 0.9440 - val_accuracy: 0.7969
Epoch 4/20
255/255 [==============================] - 189s 740ms/step - loss: 0.1481 - accuracy: 0.9677 - val_loss: 0.8737 - val_accuracy: 0.8028
Epoch 5/20
255/255 [==============================] - 188s 736ms/step - loss: 0.1312 - accuracy: 0.9686 - val_loss: 1.0381 - val_accuracy: 0.7796

Epoch 00005: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 6/20
255/255 [==============================] - 186s 728ms/step - loss: 0.0584 - accuracy: 0.9866 - val_loss: 0.5209 - val_accuracy: 0.8836
Epoch 7/20
255/255 [==============================] - 189s 741ms/step - loss: 0.0354 - accuracy: 0.9924 - val_loss: 0.4613 - val_accuracy: 0.8934
Epoch 8/20
255/255 [==============================] - 188s 738ms/step - loss: 0.0338 - accuracy: 0.9918 - val_loss: 0.5103 - val_accuracy: 0.8832

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 9/20
255/255 [==============================] - 186s 730ms/step - loss: 0.0221 - accuracy: 0.9951 - val_loss: 0.4445 - val_accuracy: 0.9013
Epoch 10/20
255/255 [==============================] - 189s 742ms/step - loss: 0.0158 - accuracy: 0.9952 - val_loss: 0.4359 - val_accuracy: 0.9004
Epoch 11/20
255/255 [==============================] - 190s 743ms/step - loss: 0.0123 - accuracy: 0.9974 - val_loss: 0.4518 - val_accuracy: 0.8974

Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 12/20
255/255 [==============================] - 188s 735ms/step - loss: 0.0111 - accuracy: 0.9968 - val_loss: 0.4405 - val_accuracy: 0.9005
```

```
Epoch 00012: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 13/20
255/255 [==============================] - 188s 739ms/step - loss: 0.0093 - accuracy: 0.9973 - val_loss: 0.4349 - val_accuracy: 0.9027
Epoch 14/20
255/255 [==============================] - 189s 742ms/step - loss: 0.0090 - accuracy: 0.9977 - val_loss: 0.4347 - val_accuracy: 0.9027
Epoch 15/20
255/255 [==============================] - 189s 741ms/step - loss: 0.0085 - accuracy: 0.9971 - val_loss: 0.4355 - val_accuracy: 0.9044

Epoch 00015: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 16/20
255/255 [==============================] - 187s 733ms/step - loss: 0.0086 - accuracy: 0.9975 - val_loss: 0.4333 - val_accuracy: 0.9041
Epoch 17/20
255/255 [==============================] - 187s 730ms/step - loss: 0.0070 - accuracy: 0.9982 - val_loss: 0.4330 - val_accuracy: 0.9047
Epoch 18/20
255/255 [==============================] - 189s 737ms/step - loss: 0.0068 - accuracy: 0.9975 - val_loss: 0.4335 - val_accuracy: 0.9052

Epoch 00018: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.
Epoch 19/20
255/255 [==============================] - 187s 735ms/step - loss: 0.0074 - accuracy: 0.9975 - val_loss: 0.4336 - val_accuracy: 0.9052

Epoch 00019: ReduceLROnPlateau reducing learning rate to 7.812500371073838e-06.
Epoch 20/20
255/255 [==============================] - 190s 743ms/step - loss: 0.0059 - accuracy: 0.9982 - val_loss: 0.4340 - val_accuracy: 0.9051

Epoch 00020: ReduceLROnPlateau reducing learning rate to 3.906250185536919e-06.

      Epoch 00020: early stopping
```

```
print(train_acc[1])
print(val_acc[1])
```

```
0.998031497001648

0.9050049781799316
```

7) Keep 1 layer and decrease dropout to 15%

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(128, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.15),
```

```
    tf.keras.layers.Dense(197, activation='softmax')
])
```

```
Epoch 1/20
255/255 [==============================] - 6167s 24s/step - loss: 4.6427 - accuracy: 0.0732 - val_loss: 4.7016 - val_accuracy: 0.0624
Epoch 2/20
255/255 [==============================] - 187s 731ms/step - loss: 2.6085 - accuracy: 0.4381 - val_loss: 3.2084 - val_accuracy: 0.2502
Epoch 3/20
255/255 [==============================] - 185s 724ms/step - loss: 1.2336 - accuracy: 0.7225 - val_loss: 1.8304 - val_accuracy: 0.5743
Epoch 4/20
255/255 [==============================] - 184s 723ms/step - loss: 0.6837 - accuracy: 0.8362 - val_loss: 1.5504 - val_accuracy: 0.6253
Epoch 5/20
255/255 [==============================] - 182s 714ms/step - loss: 0.4596 - accuracy: 0.8864 - val_loss: 1.4041 - val_accuracy: 0.6272
Epoch 6/20
255/255 [==============================] - 185s 725ms/step - loss: 0.3434 - accuracy: 0.9110 - val_loss: 0.8972 - val_accuracy: 0.7687
Epoch 7/20
255/255 [==============================] - 182s 715ms/step - loss: 0.2779 - accuracy: 0.9279 - val_loss: 0.8614 - val_accuracy: 0.7747
Epoch 8/20
255/255 [==============================] - 187s 732ms/step - loss: 0.2316 - accuracy: 0.9349 - val_loss: 1.3778 - val_accuracy: 0.6507

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 9/20
255/255 [==============================] - 184s 722ms/step - loss: 0.1083 - accuracy: 0.9725 - val_loss: 0.4502 - val_accuracy: 0.8785
Epoch 10/20
255/255 [==============================] - 187s 732ms/step - loss: 0.0624 - accuracy: 0.9861 - val_loss: 0.3728 - val_accuracy: 0.8948
Epoch 11/20
255/255 [==============================] - 189s 740ms/step - loss: 0.0505 - accuracy: 0.9889 - val_loss: 0.4187 - val_accuracy: 0.8917

Epoch 00011: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 12/20
255/255 [==============================] - 188s 735ms/step - loss: 0.0374 - accuracy: 0.9919 - val_loss: 0.3541 - val_accuracy: 0.9061
Epoch 13/20
255/255 [==============================] - 191s 749ms/step - loss: 0.0276 - accuracy: 0.9939 - val_loss: 0.3443 - val_accuracy: 0.9087
Epoch 14/20
255/255 [==============================] - 186s 730ms/step - loss: 0.0209 - accuracy: 0.9958 - val_loss: 0.3580 - val_accuracy: 0.9073

Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 15/20
255/255 [==============================] - 188s 735ms/step - loss: 0.0231 - accuracy: 0.9948 - val_loss: 0.3346 - val_accuracy: 0.9141
Epoch 16/20
255/255 [==============================] - 187s 735ms/step - loss: 0.0170 - accuracy: 0.9963 - val_loss: 0.3387 - val_accuracy: 0.9147

Epoch 00016: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 17/20
```

```
255/255 [==============================] - 190s 743ms/step - loss: 0.0159 - accuracy: 0.9971 - val_loss: 0.3312 - val_accuracy: 0.9168
Epoch 18/20
255/255 [==============================] - 186s 730ms/step - loss: 0.0126 - accuracy: 0.9975 - val_loss: 0.3306 - val_accuracy: 0.9154
Epoch 19/20
255/255 [==============================] - 185s 726ms/step - loss: 0.0123 - accuracy: 0.9975 - val_loss: 0.3323 - val_accuracy: 0.9151

Epoch 00019: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 20/20
255/255 [==============================] - 185s 725ms/step - loss: 0.0103 - accuracy: 0.9975 - val_loss: 0.3309 - val_accuracy: 0.9152


Epoch 00020: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05
```

```
print(train_acc[1])
print(val_acc[1])
```

0.9986466765403748

0.9150896668434143

8) Keep 1 layer with 256 neurons and 20% dropout

```
classification_model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.GlobalAveragePooling2D(),
    tf.keras.layers.Dense(256, activation='relu'),
    tf.keras.layers.BatchNormalization(),
    tf.keras.layers.Dropout(0.20),
    tf.keras.layers.Dense(197, activation='softmax')
])
```

```
Epoch 1/20
255/255 [==============================] - 209s 735ms/step - loss: 0.9082 - accuracy: 0.8531 - val_loss: 1.2007 - val_accuracy: 0.7589
Epoch 2/20
255/255 [==============================] - 186s 728ms/step - loss: 0.2712 - accuracy: 0.9416 - val_loss: 2.2045 - val_accuracy: 0.5580

Epoch 00002: ReduceLROnPlateau reducing learning rate to 0.0005000000237487257.
Epoch 3/20
255/255 [==============================] - 192s 754ms/step - loss: 0.0894 - accuracy: 0.9810 - val_loss: 0.4741 - val_accuracy: 0.8861
Epoch 4/20
255/255 [==============================] - 193s 759ms/step - loss: 0.0443 - accuracy: 0.9894 - val_loss: 0.5016 - val_accuracy: 0.8819

Epoch 00004: ReduceLROnPlateau reducing learning rate to 0.0002500000118743628.
Epoch 5/20
```

```
255/255 [==============================] - 194s 759ms/step - loss: 0.0305 - accuracy: 0.9937 - val_loss: 0.4325 - val_accuracy: 0.9036
Epoch 6/20
255/255 [==============================] - 194s 761ms/step - loss: 0.0269 - accuracy: 0.9937 - val_loss: 0.4283 - val_accuracy: 0.9011
Epoch 7/20
255/255 [==============================] - 193s 758ms/step - loss: 0.0200 - accuracy: 0.9959 - val_loss: 0.4143 - val_accuracy: 0.9060
Epoch 8/20
255/255 [==============================] - 189s 741ms/step - loss: 0.0209 - accuracy: 0.9952 - val_loss: 0.4454 - val_accuracy: 0.9021

Epoch 00008: ReduceLROnPlateau reducing learning rate to 0.0001250000059371814.
Epoch 9/20
255/255 [==============================] - 185s 725ms/step - loss: 0.0155 - accuracy: 0.9966 - val_loss: 0.3985 - val_accuracy: 0.9111
Epoch 10/20
255/255 [==============================] - 189s 741ms/step - loss: 0.0129 - accuracy: 0.9971 - val_loss: 0.3972 - val_accuracy: 0.9126
Epoch 11/20
255/255 [==============================] - 186s 728ms/step - loss: 0.0113 - accuracy: 0.9972 - val_loss: 0.4021 - val_accuracy: 0.9120

Epoch 00011: ReduceLROnPlateau reducing learning rate to 6.25000029685907e-05.
Epoch 12/20
255/255 [==============================] - 187s 735ms/step - loss: 0.0091 - accuracy: 0.9975 - val_loss: 0.3997 - val_accuracy: 0.9118

Epoch 00012: ReduceLROnPlateau reducing learning rate to 3.125000148429535e-05.
Epoch 13/20
255/255 [==============================] - 185s 724ms/step - loss: 0.0086 - accuracy: 0.9972 - val_loss: 0.3986 - val_accuracy: 0.9131

Epoch 00013: ReduceLROnPlateau reducing learning rate to 1.5625000742147677e-05.

Epoch 00013: early stopping
```

```
print(train_acc[1])
print(val_acc[1])
```

```
0.9976624250411987

0.9129731059074402
```

Thus, maximum validation accuracy that we could extract from this model is 91.72%

User Interface Development

- We have used **Tkinter** for UI development.

- For object detection, we first tried pre-trained **MobileNet v2 model**

```
def create_model(trainable=False):
    model = MobileNetV2(input_shape= (96,96,3), include_top=False, alpha=ALPHA)

    # to freeze layers
    for layer in model.layers:
        layer.trainable = trainable

    out =  ( model.layers[-1].output)

    x = Conv2D(4, kernel_size=3)(out)
    x = Reshape((4,), name="coords")(x)


    return Model(inputs=model.input, outputs=x


def main():
  model = create_model()
  model.summary()
  train_datagen = DataGenerator(car_train_df,'Train')
  validation_datagen = Validation(generator=DataGenerator(car_test_df,'Test'))

  model.compile(loss="mean_squared_error", optimizer="adam", metrics=[])

  checkpoint = ModelCheckpoint("model-
{val_iou:.2f}.h5", monitor="val_iou", verbose=1, save_best_only=True,
                                 save_weights_only=True, mode="max")
  stop = EarlyStopping(monitor="val_iou", patience=PATIENCE, mode="max")
  reduce_lr = ReduceLROnPlateau(monitor="val_iou", factor=0.2, patience=10, min_l
r=1e-7, verbose=1, mode="max")
    #callback.set_model(model)

  model.fit(train_datagen,
                        epochs=EPOCHS,
                        callbacks=[ validation_datagen,checkpoint, reduce_lr, sto
p],
                        workers=THREADS,
                        use_multiprocessing=MULTI_PROCESSING,
                        shuffle=True,
                        verbose=1)
```

```
if __name__ == "__main__":
    main()
```

But using this model we got overall IOU on only around 0.7.

Therefore, we decided to use **ImageAI library** for object detection. This model gave us very high IOU of more than 0.9

Overall code for UI:

#Import Required libraries

```
from tensorflow import keras
from imageai.Detection import ObjectDetection
import os


import tkinter as tk
from tkinter import ttk
from tkinter import Button
from tkinter import filedialog
from PIL import ImageTk, Image
win = tk.Tk()
win.title("Car Classification")

# Set the resolution of window
win.geometry('550x300')

# Allow Window to be resizable
win.resizable(width = True, height = True)
```

1) Functions to automate tasks for milestone 1

1. Import the data
2. Map training and testing images to its classes.
3. Map training and testing images to its annotations.

Output: Images mapped to its class and annotation ready to be used for deep learning

➢ Function and UI control to Import data

```
def import_data():
    global car_names, train_data, test_date

    #Different car labels
    car_names = pd.read_csv( 'Car names and make.csv', header=None, names = ['Car
Label'] )

    #Train data
    train_data = pd.read_csv( 'Annotations/Train Annotations.csv', skiprows=1, na
mes = ['ImageName', 'X1', 'Y1', 'X2' , 'Y2', 'Class'] )

    #Test data
    test_data = pd.read_csv( 'Annotations/Test Annotation.csv' , skiprows=1, name
s = ['ImageName', 'X1', 'Y1', 'X2' , 'Y2', 'Class'])

    #Replace '/' with '-' in the name
    car_names.loc[173,'CarLabel'] = 'Ram C-V Cargo Van Minivan 2012'


    Data=ttk.Label(win,text="Data Successfully Imported")
    Data.grid(row=0,column=1,sticky=tk.W)




# Import Data Button

databutton = Button(win, text="Import Data", command=import_data, fg='blue')
databutton.grid(row=0,column=0)
```

➢ Function and UI control to map training and test images to classes and annotations

```
def map_images():

    global car_train_df, car_test_df

    #Map training images to corresponding classes and annotations
```

```
    car_names['Class'] = car_names.index + 1
    car_train_df = pd.merge(train_data, car_names, how = 'left', left_on='Class',
 right_on='Class' )

    #Map training images to corresponding classes and annotations
    car_test_df = pd.merge(test_data, car_names, how = 'left', left_on='Class', r
ight_on='Class' )


    Data=ttk.Label(win,text="Training and Test images mapped to classes and annot
ations")
    Data.grid(row=1,column=1,sticky=tk.W)



# Map Data Button

databutton = Button(win, text="Map Training and Test Data", command=map_images, f
g='blue')
databutton.grid(row=1,column=0)
```

2) Functions to automate tasks for milestone 2

1. Design, train and test model to classify the car.
2. Pickle model to be used for future prediction

Output: Pickled model to be used for future prediction

➢ **Function to design and train classifier; and add required UI controls**

```
def TrainClassifier():
    global classification_model

    base_model = InceptionResNetV2(include_top=False, input_shape = INPUT_SIZE)

    classification_model = tf.keras.Sequential([
        base_model,
        tf.keras.layers.GlobalAveragePooling2D(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.BatchNormalization(),
        tf.keras.layers.Dropout(0.20),
        tf.keras.layers.Dense(197, activation='softmax')
    ])

    #Define optimizer
    lr=0.001
    classification_model.compile(loss='categorical_crossentropy', optimizer=Adam(
lr=lr), metrics=['accuracy'])
```

```
    #Early Stopping and Save best model
    patience = 1
    stop_patience = 3
    factor = 0.5

    callbacks = [
        tf.keras.callbacks.ModelCheckpoint("classify_model.h5", save_best_only=True, verbose = 0),
        tf.keras.callbacks.EarlyStopping(patience=stop_patience, monitor='val_accuracy', verbose=1),
        tf.keras.callbacks.ReduceLROnPlateau(monitor='val_accuracy', factor=factor, patience=patience, verbose=1)
    ]

    #Training [Forward pass and Backpropagation]
    epochs = 20
    history = classification_model.fit(train_generator, validation_data=validation_generator, epochs=epochs, callbacks=callbacks, verbose=1)



    Data=ttk.Label(win,text="Classifier designed and trained")
    Data.grid(row=3,column=1,sticky=tk.W)



# Train Classifier Button

databutton = Button(win, text="Design and Train Model", command=TrainClassifier, justify='right',fg='blue')
databutton.grid(row=3,column=0)
```

- **Function and UI to pickle model**

```
def pickle_model():

    #Pickle model for future use

    final_model = classification_model
    final_model.save('./final_model.h5')

    Data=ttk.Label(win,text="Model Pickled")
    Data.grid(row=4,column=1,sticky=tk.W)

# Pickle Model Button
databutton = Button(win, text="Pickle Model", command=pickle_model, justify='right',fg='blue')
databutton.grid(row=4,column=0)
```

3) Design a clickable UI based interface which can allow the user to browse & input the image, output the class and the bounding box of the input image

Install imageai library

```
!pip install imageai –upgrade
```

- **Define function to open dialog box to select image**

```
def openfilename():

    # open file dialog box to select image
    filename = filedialog.askopenfilename(title ='Car')
    return filename
```

- **Function to load image and display on UI**

```
def load_img():

    global img,x,img1
    # Select the Imagename  from a folder
    x = openfilename()

    # opens the image
    img = Image.open(x)

    # resize the image and apply a high-quality down sampling filter
    img = img.resize((250, 250), Image.ANTIALIAS)

    img1 = img

    # PhotoImage class is used to add image to widgets, icons etc
    img = ImageTk.PhotoImage(img)

    # create a label
    Data=ttk.Label(win, image = img)

    # set the image as img
    Data.grid(row = 7, column = 0)
    Data.image = img
```

```
# Upload Image Button

databutton = Button(win, text="Upload Image", command=load_img, justify='right
',fg='blue')
databutton.grid(row=5,column=0)
```

➢ **Function to detect car in image, draw bounding box around it and display on UI**

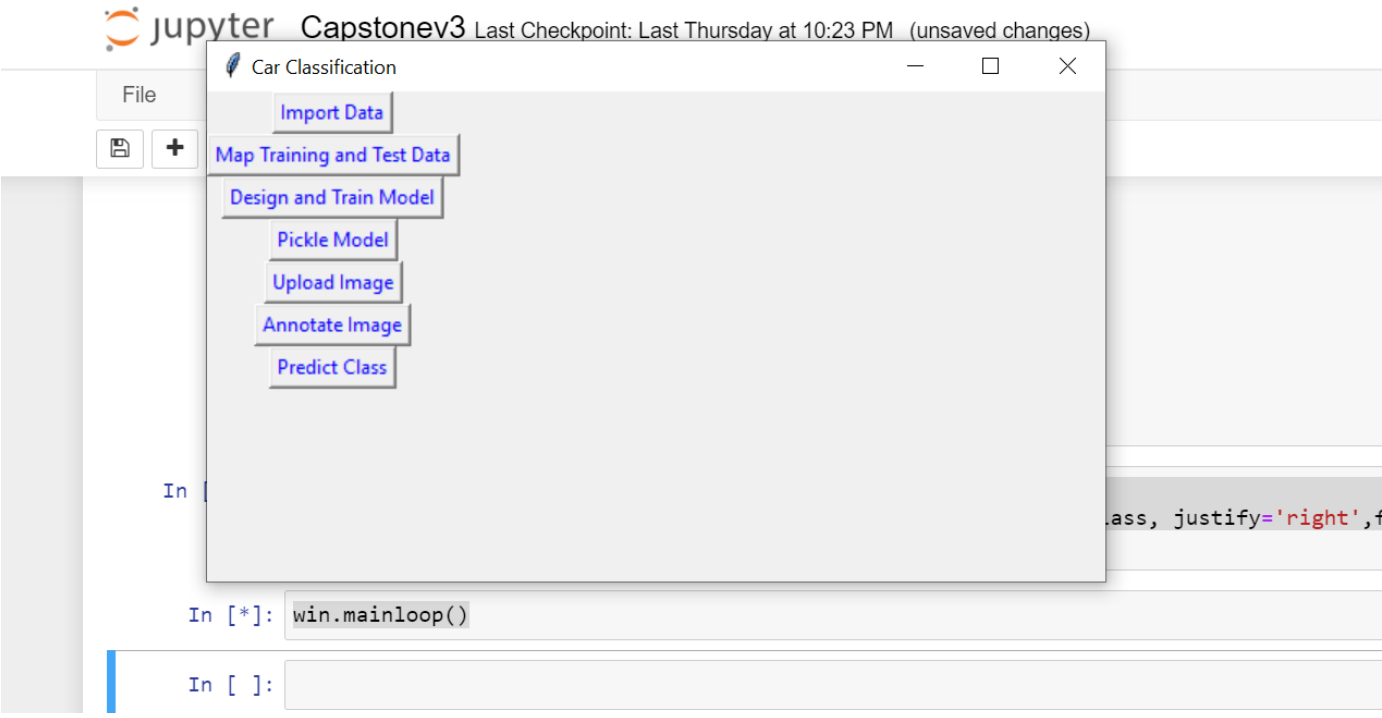
```
def detect_object():
    from imageai.Detection import ObjectDetection
    import os

    current_directory = os.getcwd()

    detector = ObjectDetection()
    detector.setModelTypeAsYOLOv3()

    detector.setModelPath(os.path.join(current_directory , "yolo.h5"))
    detector.loadModel()

    detections = detector.detectObjectsFromImage(input_image=x,output_image_pa
th=os.path.join(current_directory , "annoted.jpg"),minimum_percentage_probabil
ity=60)
    image_a = "annoted.jpg"


        # opens the image
    img = Image.open(image_a)

    # resize the image and apply a high-quality down sampling filter
    img = img.resize((250, 250), Image.ANTIALIAS)

    # PhotoImage class is used to add image to widgets, icons etc
    img = ImageTk.PhotoImage(img)

    # create a label
    Data=ttk.Label(win, image = img)

    # set the image as img
    Data.grid(row = 7, column=1)
    Data.image = img


#Annotate image Button
databutton = Button(win, text="Annotate Image", command=detect_object, justify='r
ight',fg='blue')
databutton.grid(row=8,column=0)
```

➢ **Function to predict class**

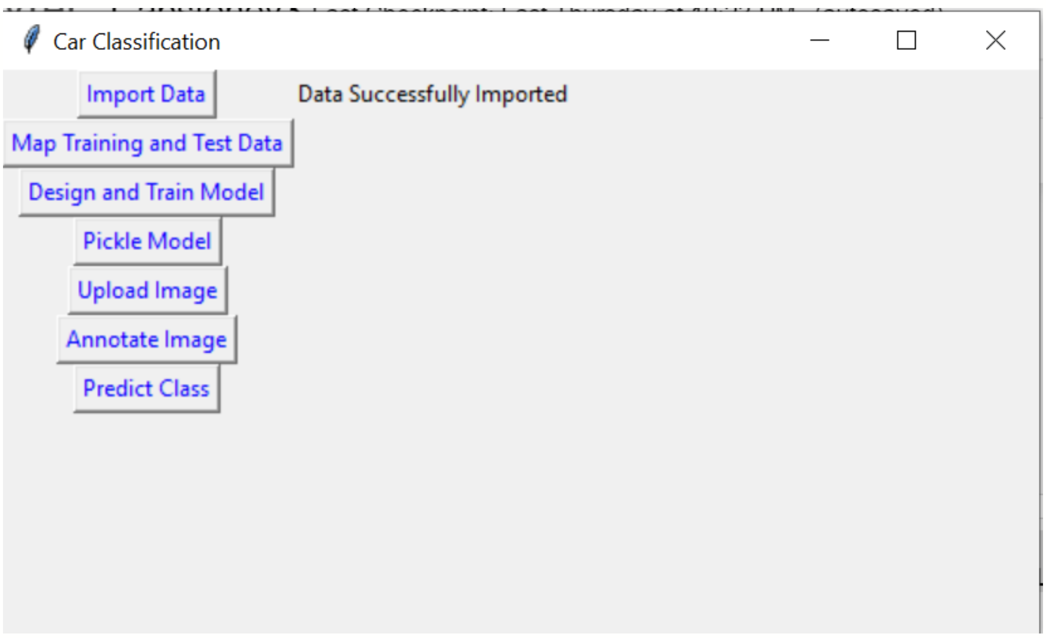
```
#function to predict class of image
def predict_class():

    path = /content/drive/MyDrive/Colab Notebooks/Capstone/annoted.jpg'

    image = cv2.imread(path)

    image = cv2.resize(image, (224,224),)
    pixels = image.astype('float32')
    pixels /= 255.0
    pixels = np.expand_dims(pixels, axis=0)
    prediction = final_model.predict(pixels)

    #Convert encoding to integer
    pred_int = np.argmax(prediction, axis=1)
    label = car_names[car_names['Class'] == pred_int[0]]


    Data=ttk.Label(win,text=label)
    Data.grid(row=9,column=1,sticky=tk.W)



#Predict class Button
databutton = Button(win, text="Predict Class", command=predict_class, justify=
'right',fg='blue')
databutton.grid(row=9,column=0)


win.mainloop()
```

User Interface

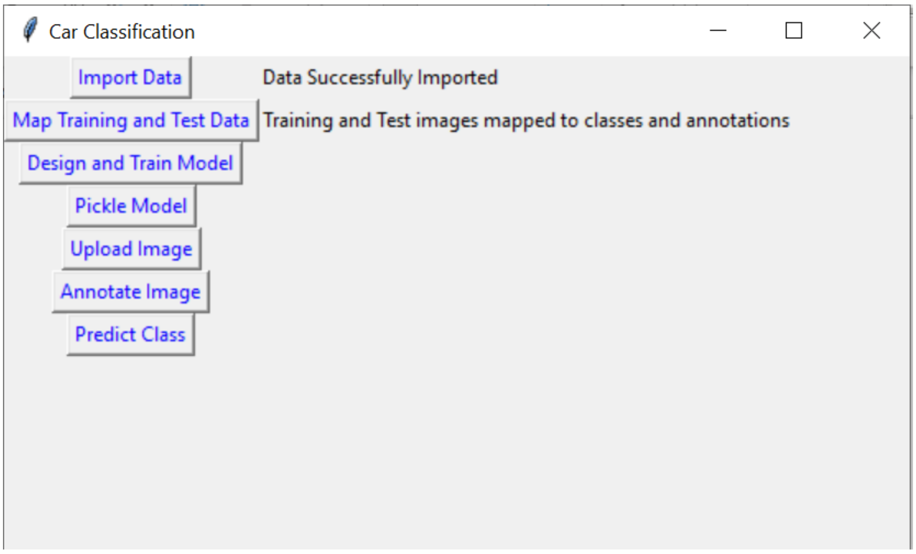
- **Launching User Interface**



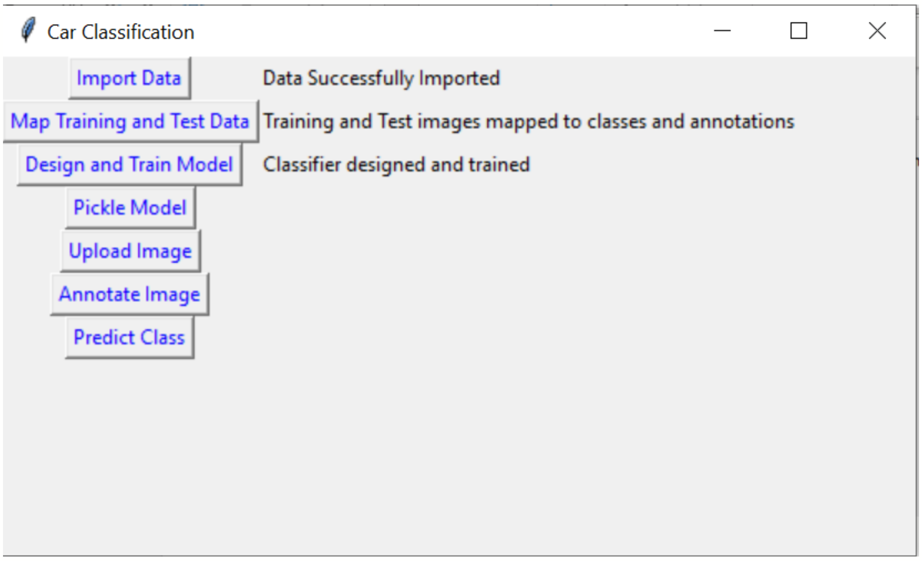
- **Import Data**



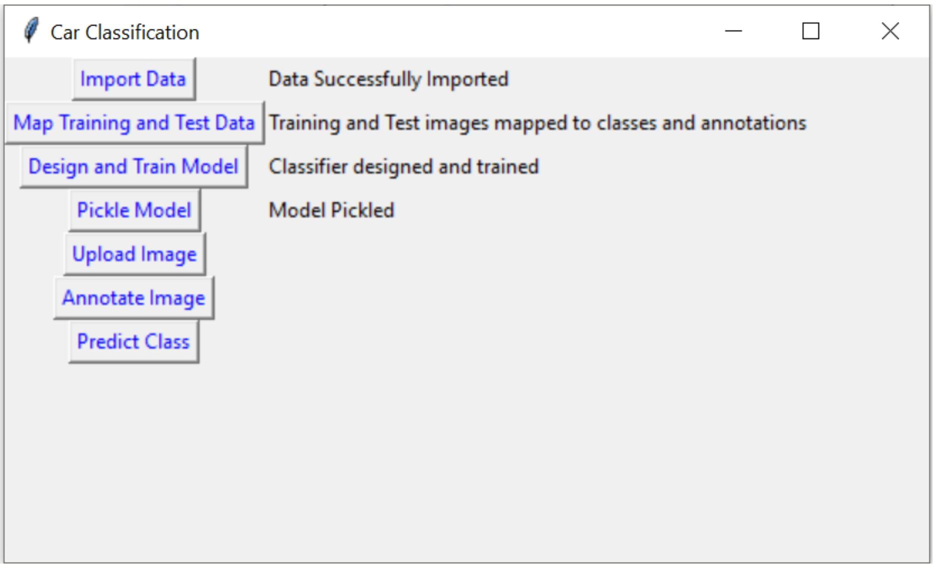
- **Map Training and test data to classes and annotations**



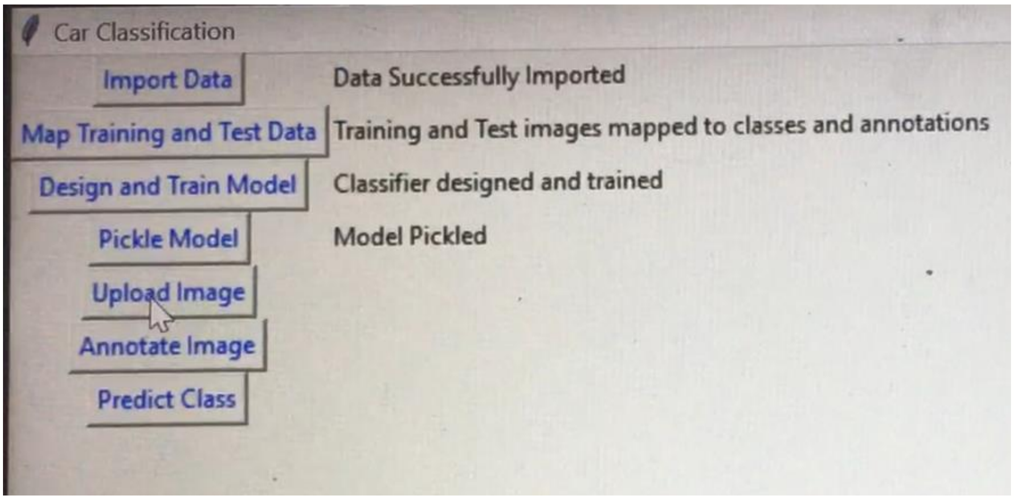
- **Design and Train the classifier**

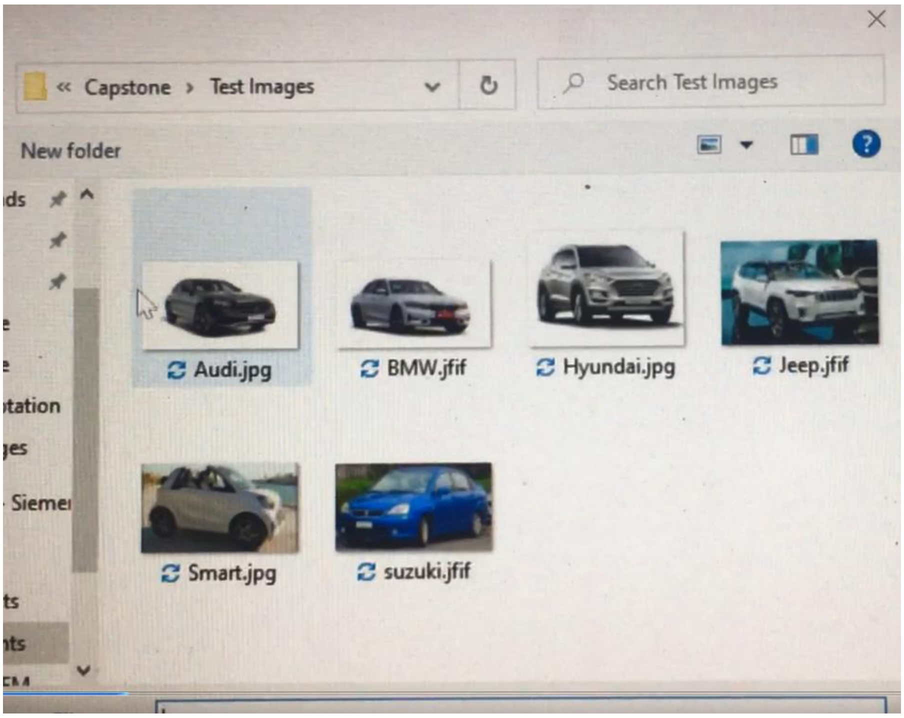


- **Pickle the model**

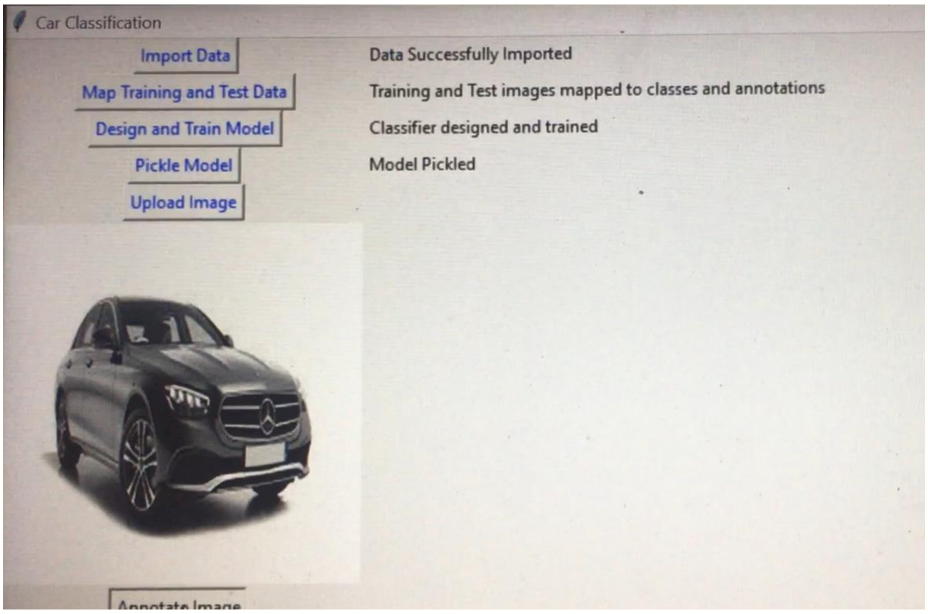


- **Upload Image**



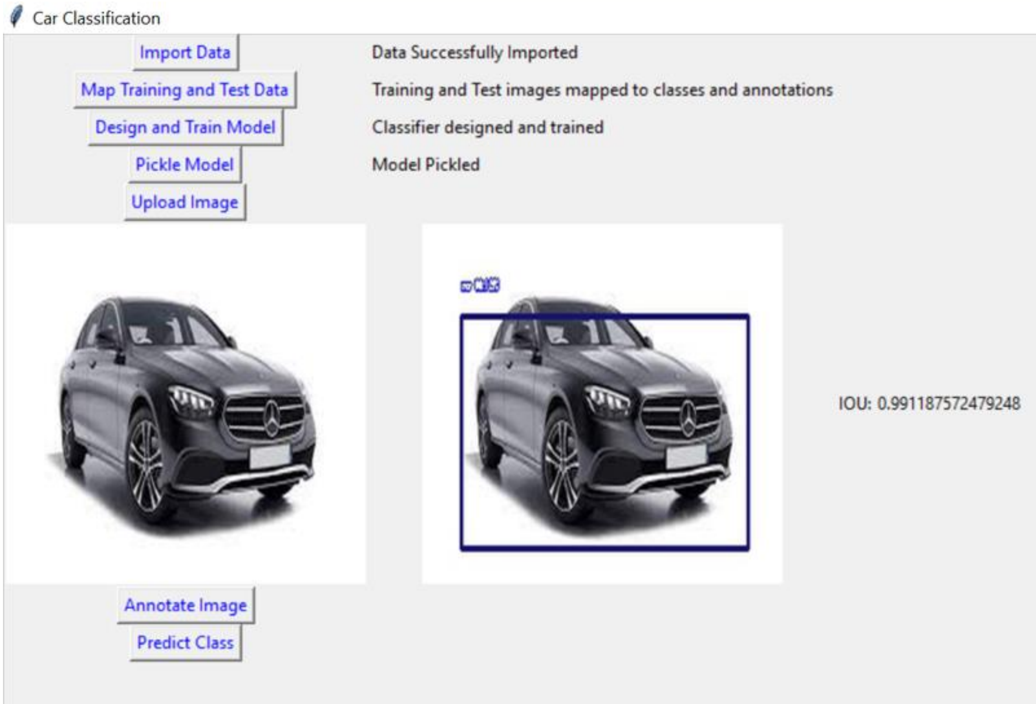


« Capstone › Test Images
Search Test Images
New folder
Audi.jpg
BMW.jfif
Hyundai.jpg
Jeep.jfif
Smart.jpg
suzuki.jfif

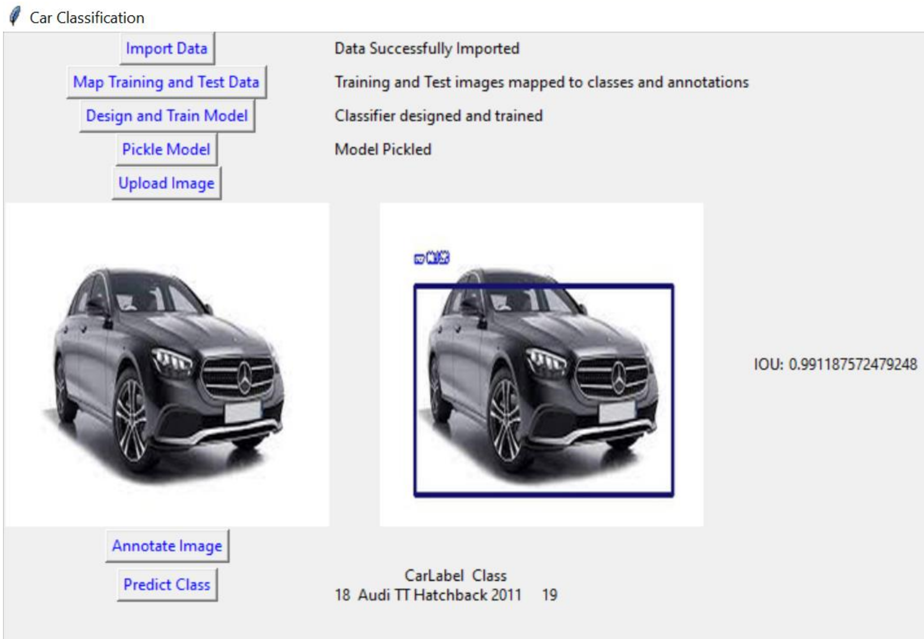


Car Classification
Import Data
Data Successfully Imported
Map Training and Test Data
Training and Test images mapped to classes and annotations
Design and Train Model
Classifier designed and trained
Pickle Model
Model Pickled
Upload Image

- **Annotate Image and Display IOU**



- **Predict Class**



- **Upload and Annotate one other image and predict class**