To connect with drive

```
from google.colab import drive
drive.mount("/content/drive")
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

Import lib

```
import os
import cv2
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from google.colab.patches import cv2_imshow
import warnings
warnings.filterwarnings('ignore') # Hide all warnings
from tensorflow import keras
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, Flatten, MaxPooling2D, Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras import optimizers, losses
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.preprocessing import image
import pickle
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.layers import Dense,Dropout,BatchNormalization
from tensorflow.keras.preprocessing import image
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras import optimizers
from keras.layers.pooling import GlobalAveragePooling2D
import numpy as np
import seaborn as sns
import os
from tensorflow.keras import Model,layers
from tensorflow.keras.optimizers import Adam
from sklearn.metrics import confusion_matrix
```

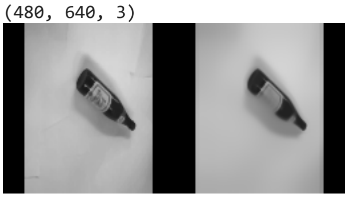
Dataset Description:

Our dataset consist of cans,glass_bottles,plastic_bottles

Read a image and resize & remove noise

```
image = cv2.imread('/content/drive/MyDrive/bottle/glass_bottles/glass_bottles/bdtmp.jpg')
print(image.shape)
input_size = 128
image_size = (input_size, input_size)
image = cv2.resize(image, image_size)
image1 = cv2.fastNlMeansDenoising(image,None,20,7,21)
Hori = np.concatenate((image, image1), axis=1)
cv2_imshow(Hori)
```

```
(480, 640, 3)
```



Preprocessing stage

Read each images and resize & remove noise each images of the dataset

```
data = []
labels = []
# Access the directory and sub-directories and so on
directory = "/content/drive/MyDrive/bottle"

# Extract all images file inside the folders and stored them into list
for sub_folder in os.listdir(directory):
    sub_folder_path = os.path.join(directory, sub_folder)
    for sub_sub_folder in os.listdir(sub_folder_path):
        sub_sub_folder_path = os.path.join(sub_folder_path, sub_sub_folder)
        for image_file in os.listdir(sub_sub_folder_path):
            if image_file.endswith(".jpg") or image_file.endswith(".png"): # Check if the file ends with either '.jped' or '.png'
                image_path = os.path.join(sub_sub_folder_path, image_file)
                # Read the image using OpenCV
                image = cv2.imread(image_path) #the decoded images stored in **B G R** order.
                # Resize the image to a standard size
                image = cv2.resize(image, image_size)
                image = cv2.fastNlMeansDenoising(image,None,20,7,21)
                # Append the image to the data list
                data.append(image)
                # Append the label to the labels list
                labels.append(sub_folder)

# Convert the data and labels lists into numpy arrays
data = np.array(data)
labels = np.array(labels)

# Print the dimension of dataset
print(f'data shape:{data.shape}')
print(f'labels shape:{labels.shape}')
```

```
data shape:(8215, 128, 128, 3)
labels shape:(8215,)
```

See how many numbers of each labels

```
df = pd.DataFrame({"label":labels})
df.value_counts()
```

```
label
plastic_bottles    7576
cans                495
glass_bottles       144
dtype: int64
```

```
df = pd.DataFrame({" bottle label":labels})
df.value_counts().plot(kind='bar')
plt.xticks(rotation = 0)
plt.show()
```

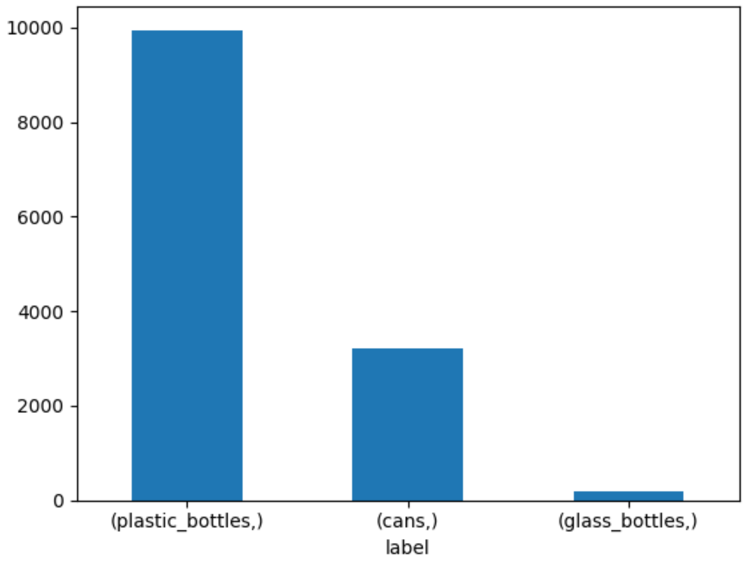
Generate augmented data

```
from keras.preprocessing.image import ImageDataGenerator

# Load the data
X = data # array of preprocessed data
y = labels # array of labels
n_gen = 40

# Create data generator
datagen = ImageDataGenerator(
        rotation_range=0, #0
        width_shift_range=0.2,
        height_shift_range=0.2,
        shear_range=0.2,
        zoom_range=0.2,
        horizontal_flip=True,
        fill_mode='nearest')

# Fit the data generator on the data
datagen.fit(X)

# Generate augmented data
X_augmented, y_augmented = [], []

# resampling with equaly labels ratio

# With resampling
for X_batch, y_batch in datagen.flow(X[:308], y[:308], batch_size=32):
    X_augmented.append(X_batch)
    y_augmented.append(y_batch)
    if len(X_augmented) >= n_gen: # Setting generated augmented data
        break

for X_batch, y_batch in datagen.flow(X[308:447], y[308:447], batch_size=32):
    X_augmented.append(X_batch)
    y_augmented.append(y_batch)
    if len(X_augmented) >= n_gen*2.3: # Setting generated augmented data
        break

for X_batch, y_batch in datagen.flow(X[447:], y[447:], batch_size=32):
    X_augmented.append(X_batch)
    y_augmented.append(y_batch)
    if len(X_augmented) >= n_gen*4.2: # Setting generated augmented data
        break

# Concatenate augmented data with original data
data = np.concatenate((X, np.concatenate(X_augmented)))
labels = np.concatenate((y, np.concatenate(y_augmented)))

print(f"data augmented shape : {data.shape}")
print(f"labels augmented shape : {labels.shape}")

import pandas as pd
df = pd.DataFrame({"label":labels})
df.value_counts()
```

```
data augmented shape : (13333, 128, 128, 3)
labels augmented shape : (13333,)
label
plastic_bottles    9948
cans               3196
glass_bottles       189
dtype: int64
```

See how many numbers of each labels after I regenerated data.

```
df = pd.DataFrame({"label":labels})
df.value_counts().plot(kind='bar')
plt.xticks(rotation = 0) # Rotates X-Axis Ticks by 45-degrees
plt.show()
```




```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, random_state=42)

data = X_train # Split training data
labels = y_train # Split training labels

X_test = X_test # Test data
y_test = y_test # Test labels
```

```
import pandas as pd

print(f'data shape:{data.shape}')
print(f'labels shape:{labels.shape}')
df = pd.DataFrame({"label":labels})
print(df.value_counts())
print("")
print(f'test_date shape:{X_test.shape}')
print(f'test_labels shape:{y_test.shape}')
df = pd.DataFrame({"test_labels":y_test})
print(df.value_counts())
```

```
data shape:(10666, 128, 128, 3)
labels shape:(10666,)
label
plastic_bottles    7997
cans               2520
glass_bottles       149
dtype: int64

test_date shape:(2667, 128, 128, 3)
test_labels shape:(2667,)
test_labels
plastic_bottles    1951
cans                676
glass_bottles        40
dtype: int64
```

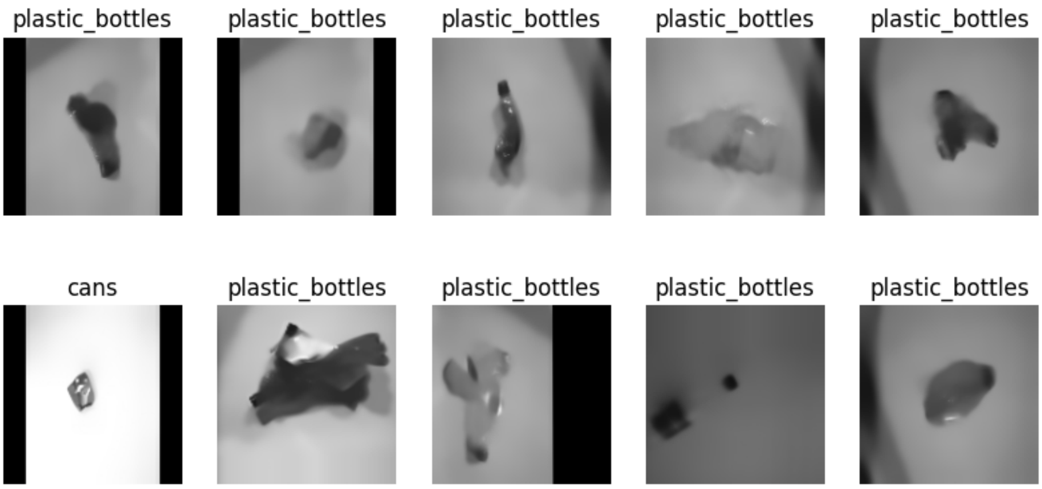
```
# Normalize the pixel values to a range between 0 and 1
data = data / 255.0
X_test = X_test / 225.0
```

```
labels = labels
# Convert the labels into one-hot encoded arrays
labels_one_hot = np.zeros((labels.shape[0], 3))

for i, label in enumerate(labels):
    if label == "plastic_bottles":
        labels_one_hot[i, 0] = 1
    elif label == "cans":
        labels_one_hot[i, 1] = 1
```

```
    else:
        labels_one_hot[i, 2] = 1
```

Show a sample of images from the dataset

```
data = data

# choose 20 random indices
indices = np.random.randint(0, len(data), 10)

# Get 20 sample images
sample_images = data[indices]

# Plot the images
fig = plt.figure(figsize=(10,10))
for i, img in enumerate(sample_images):
    plt.subplot(4, 5, i+1)
    plt.imshow(img)
    plt.axis('off')
    plt.title(labels[indices[i]])

plt.show()
```



Create my CNN model

```
def run_custom_model(batch_size, epochs):

    import tensorflow as tf
    from tensorflow import keras
    from tensorflow.keras import layers
    from tensorflow.keras.optimizers import Adam, SGD
    from tensorflow.keras.callbacks import ModelCheckpoint

    # set seed value for randomization
    # np.random.seed(42)
    tf.random.set_seed(42)

    # Build the model using a Convolutional Neural Network
    model = keras.Sequential([
        keras.layers.Conv2D(32, (3,3), activation='relu', input_shape=(input_size,input_size,3)),
        keras.layers.Conv2D(32, (3,3), activation='relu'),
        keras.layers.MaxPooling2D(2,2),
        keras.layers.Dropout(0.2),

        keras.layers.Conv2D(64, (3,3), activation='relu'),
        keras.layers.Conv2D(64, (3,3), activation='relu'),
        keras.layers.MaxPooling2D(2,2),
        keras.layers.Dropout(0.2),

        keras.layers.Conv2D(256, (3,3), activation='relu'),
        keras.layers.Conv2D(256, (3,3), activation='relu'),
        keras.layers.MaxPooling2D(2,2),
        keras.layers.Dropout(0.2),
```

```
        keras.layers.Flatten(),
        keras.layers.Dense(1024, activation='relu'),
        keras.layers.Dropout(0.5),
        keras.layers.Dense(3, activation='softmax')
    ])


    # Compile the model
    model.compile(optimizer=Adam(), loss='categorical_crossentropy', metrics=['accuracy'])

    # See an overview of the model architecture and to debug issues related to the model layers.
    model.summary()

###########################################################################################

    import time
    start_time = time.time() #To show the training time

    # Train the model

    # set an early stopping mechanism
    # set patience to be tolerant against random validation loss increases
    early_stopping = tf.keras.callbacks.EarlyStopping(patience=5)
    filepath = "/content/drive/MyDrive/cnnmodel_{epoch:02d}-{val_accuracy:.2f}.h5"

    # Using the ModelCheckpoint function to train and store all the best models
    checkpoint1 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

    callbacks_list = [checkpoint1]

    # history = model.fit(data, labels_one_hot, batch_size=32, epochs=10, validation_split=0.2)
    history = model.fit(x=data,
                        y=labels_one_hot,
                        batch_size=batch_size,
                        epochs=epochs,
                        validation_split=0.2,
                        callbacks=callbacks_list)


    # Evaluate the model
    print("Test accuracy: ", max(history.history['val_accuracy']))

    # Assign the trained model
    self_train_model = history

    end_time = time.time() # To show the training time
    training_time = end_time - start_time
    print("Training time:", training_time, "seconds")

    self_train_model_time = training_time

    return self_train_model, self_train_model_time
```

```
# Run CNN model
self_train_model, self_train_model_time = run_custom_model(batch_size = 256,epochs = 1)
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 126, 126, 32)      896       
                                                                 
 conv2d_1 (Conv2D)           (None, 124, 124, 32)      9248      
                                                                 
 max_pooling2d (MaxPooling2D  (None, 62, 62, 32)       0         
 )                                                               
                                                                 
 dropout (Dropout)           (None, 62, 62, 32)        0         
                                                                 
 conv2d_2 (Conv2D)           (None, 60, 60, 64)        18496     
                                                                 
 conv2d_3 (Conv2D)           (None, 58, 58, 64)        36928     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 29, 29, 64)       0         
 2D)                                                             
                                                                 
 dropout_1 (Dropout)         (None, 29, 29, 64)        0         
                                                                 
 conv2d_4 (Conv2D)           (None, 27, 27, 256)       147712    
                                                                 
 conv2d_5 (Conv2D)           (None, 25, 25, 256)       590080    
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 12, 12, 256)      0         
 2D)
```

```
 dropout_2 (Dropout)         (None, 12, 12, 256)       0

 flatten (Flatten)           (None, 36864)             0

 dense (Dense)               (None, 1024)              37749760

 dropout_3 (Dropout)         (None, 1024)              0

 dense_1 (Dense)             (None, 3)                 3075

=================================================================
Total params: 38,556,195
Trainable params: 38,556,195
Non-trainable params: 0
_________________________________________________________________
34/34 [==============================] - ETA: 0s - loss: 0.9178 - accuracy: 0.6997
Epoch 1: val_accuracy improved from -inf to 0.76523, saving model to /content/drive/MyDrive/cnnmodel_01-0.77.h5
34/34 [==============================] - 295s 8s/step - loss: 0.9178 - accuracy: 0.6997 - val_loss: 0.5700 - val_accuracy: 0.7652
Test accuracy:  0.7652296423912048
Training time: 326.83559703826904 seconds
```

```
# Check our folder and import the model with best validation accuracy
from tensorflow.keras.preprocessing import image
loaded_best_model = keras.models.load_model("/content/drive/MyDrive/cnnmodel_04-0.89.h5")

# Custom function to load and predict label for the image
def predict(img_rel_path):

    img = image.load_img(img_rel_path, target_size=(128, 128))

    # Convert Image to a numpy array
    img = image.img_to_array(img, dtype=np.uint8)

    # Scaling the Image Array values between 0 and 1
    img = np.array(img)/255.0

    # Plotting the Loaded Image
    plt.title("Loaded Image")
    plt.axis('off')
    plt.imshow(img.squeeze())
    plt.show()

    # Get the Predicted Label for the loaded Image
    p = loaded_best_model.predict(img[np.newaxis, ...])

    # Label array
    labels = {0: 'cans', 1: 'glass_bottles',2:'plastic_bottles'}

    print("\n\nMaximum Probability: ", np.max(p[0], axis=-1))
    predicted_class = labels[np.argmax(p[0], axis=-1)]
    print("Classified:", predicted_class, "\n\n")

    classes=[]
    prob=[]
    print("\n-------------------Individual Probability--------------------------------\n")

    for i,j in enumerate (p[0],0):
        print(labels[i].upper(),':',round(j*100,2),'%')
        classes.append(labels[i])
        prob.append(round(j*100,2))

    def plot_bar_x():
        # this is for plotting purpose
        index = np.arange(len(classes))
        plt.bar(index, prob)
        plt.xlabel('Labels', fontsize=8)
        plt.ylabel('Probability', fontsize=8)
        plt.xticks(index, classes, fontsize=8, rotation=20)
        plt.title('Probability for loaded image')
        plt.show()
    plot_bar_x()
```

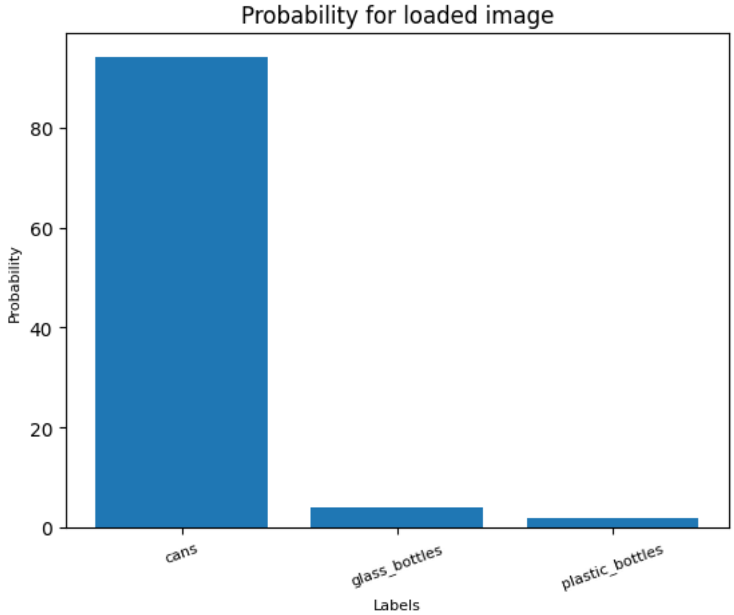
```
image = cv2.imread('/content/drive/MyDrive/bottle/glass_bottles/glass_bottles/bdtmp.jpg')
input_size = 128
image_size = (input_size, input_size)
image = cv2.resize(image, image_size)
cv2.imwrite('sample.jpg', image)
```

```
True
```

```
from tensorflow.keras.preprocessing import image
predict("/content/sample.jpg")
```

Loaded Image

```
1/1 [==============================] - 1s 1s/step


Maximum Probability:  0.94134986
Classified: cans



-------------------Individual Probability--------------------------------

CANS : 94.13 %
GLASS_BOTTLES : 4.08 %
PLASTIC_BOTTLES : 1.78 %
```




```
def output_converter(model_output):

    import numpy as np

    output = model_output

    # assume that 'output' is a numpy array of shape (n, 3)
    output_labels = ['gan', 'glass', 'plastci']
    predictions = np.argmax(output, axis=1)
    predicted_labels = [output_labels[p] for p in predictions]

    return predicted_labels
```

```
'''
Plot a Heatmap-Crosstab table out of predicted labels and True labels
'''
def plot_hm_ct(y_true, y_pred):
    import pandas as pd
```

```
    import seaborn as sns
    import matplotlib.pyplot as plt

    # create a DataFrame from y_true and y_pred
    df = pd.DataFrame({'y_true': y_true, 'y_pred': y_pred})

    # create cross-tabulation matrix
    ctab = pd.crosstab(df['y_true'], df['y_pred'])

    # create heatmap using seaborn
    sns.heatmap(ctab, annot=True, cmap='Blues', fmt='d')

    # add labels and title
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.title('Confusion Matrix')

    # show the plot
    plt.show()
```

```
def generate_cf(model, name):

    import pandas as pd
    import seaborn as sns
    import matplotlib.pyplot as plt

    # Assign model to variable 'history'
    history = model

    # Load output data
    y_pred = output_converter(history.model.predict(X_test))
    y_true = y_test

    # Plot the confusion matrix
    # create a DataFrame from y_true and y_pred
    df = pd.DataFrame({'y_true': y_true, 'y_pred': y_pred})

    # create cross-tabulation matrix
    ctab = pd.crosstab(df['y_true'], df['y_pred'])

    # create heatmap using seaborn
    sns.heatmap(ctab, annot=True, cmap='Blues', fmt='d')

    # add labels and title
    plt.xlabel('Predicted label')
    plt.ylabel('True label')
    plt.title('{} Confusion Matrix'.format(name))

    # show the plot
    plt.show()
    from sklearn.metrics import classification_report
    target_names = ['cans','glass_bottles','plastic_bottles']
    print(classification_report(y_test.classes, y_pred, target_names=target_names))

    # Calculate accuracy score
    from sklearn.metrics import accuracy_score
    accuracy = accuracy_score(y_true, y_pred)
    print("{} accuracy score: {}".format(name, accuracy))
```

```
generate_cf(self_train_model, 'Self Train CNNs')
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-3-3edb4c54a3e9> in <cell line: 1>()
----> 1 generate_cf(self_train_model, 'Self Train CNNs')

NameError: name 'self_train_model' is not defined
```

SEARCH STACK OVERFLOW

Hyperparameter tunning-my cnn model

```
from sklearn.model_selection import GridSearchCV
from keras.wrappers.scikit_learn import KerasClassifier
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
import tensorflow as tf
from keras.callbacks import EarlyStopping
```

```
from tensorflow.keras.callbacks import ModelCheckpoint


import warnings
warnings.filterwarnings('ignore') # Hide all warnings

import time
start_time = time.time() #To show the training time

tf.random.set_seed(42)
batch_size = [32,64,128 ,256]
epochs = [5,10]
optimizer = ['adam']
# optimizer = ['adam', 'rmsprop']
# cv = 5 # None mean default (K-fold=5)
cv = [(slice(None), slice(None))]


# Design Model Layers
def create_model(optimizer):
    model = Sequential()
    model.add(Conv2D(32, (3,3), activation='relu', input_shape=(input_size, input_size, 3)))
    model.add(Conv2D(32, (3, 3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(64, (3, 3), activation='relu',padding='same'))
    model.add(Conv2D(64, (3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Conv2D(256, (3, 3), activation='relu',padding='same'))
    model.add(Conv2D(256, (3, 3),activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    model.add(Dropout(0.2))

    model.add(Flatten())
    model.add(Dense(1024, activation='relu'))
    model.add(Dropout(0.5))
    model.add(Dense(3, activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer=optimizer, metrics=['accuracy'])
    return model


model = KerasClassifier(build_fn=create_model)
filepath = "/content/drive/MyDrive/Tune_cnnmodel_{epoch:02d}-{val_accuracy:.2f}.h5"

# Using the ModelCheckpoint function to train and store all the best models
checkpoint1 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

callbacks_list = [checkpoint1]

param_grid = {'batch_size': batch_size,
              'epochs': epochs,
              'optimizer': optimizer,
              'callbacks': callbacks_list}


grid = GridSearchCV(estimator=model, param_grid=param_grid, cv=cv)
grid_result = grid.fit(data, labels_one_hot, verbose=0)

print("Best: %f using %s" % (grid_result.best_score_, grid_result.best_params_))
means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))

end_time = time.time() # To show the training time
training_time = end_time - start_time
print("Training time:", training_time, "seconds")
grid_time = training_time
```

```
import pandas as pd
print(pd.DataFrame(grid_result.cv_results_))
```

```
output_labels = ['plastic_bottle', 'cans', 'glass_bottle']
result = grid.predict(X_test)

predicted_labels = list(map(lambda x: output_labels[x], result))
```

```
import seaborn as sns

# Load output data
y_pred = predicted_labels
y_true = y_test

# Plot the confusion matrix
# create a DataFrame from y_true and y_pred
df = pd.DataFrame({'y_true': y_true, 'y_pred': y_pred})

# create cross-tabulation matrix
ctab = pd.crosstab(df['y_true'], df['y_pred'])

# create heatmap using seaborn
sns.heatmap(ctab, annot=True, cmap='Blues', fmt='d')

# add labels and title
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('GridSerachCV result Confusion Matrix')

# show the plot
plt.show()

# Calculate accuracy score
from sklearn.metrics import accuracy_score
accuracy = accuracy_score(y_true, y_pred)
print("GridSerachCV accuracy score:{}".format(accuracy))
```

```
import matplotlib.pyplot as plt

# Load the data
X_test = X_test

# choose 20 random indices
indices = np.random.randint(0, len(X_test), 10)

# Get 20 sample images
sample_images = X_test[indices]

# Plot the images
fig = plt.figure(figsize=(10,10))
for i, img in enumerate(sample_images):
    plt.subplot(4, 5, i+1)
    plt.imshow(img)
    plt.axis('off')
    plt.title( y_true[indices[i]] + "\n" + "Predicted result: " + "\n"+ y_pred[indices[i]])

plt.show()
```

my customised InceptionV3 model

```
data_dir = '/content/drive/MyDrive/bottle'
datagenerator = {
    "train": ImageDataGenerator(horizontal_flip=True,
                                vertical_flip=True,
                                rescale=1. / 255,
                                validation_split=0.1,
                                shear_range=0.1,
                                zoom_range=0.1,
                                width_shift_range=0.1,
                                height_shift_range=0.1,
                                rotation_range=30,
                               ).flow_from_directory(directory=data_dir,
```

```
                                                     target_size=(300, 300),
                                                     subset='training',
                                                    ),

    "valid": ImageDataGenerator(rescale=1 / 255,
                                 validation_split=0.1,
                                ).flow_from_directory(directory=data_dir,
                                                     target_size=(300, 300),
                                                     subset='validation',
                                                    ),
}
```

```
base_model = InceptionV3(weights=None, include_top=False, input_shape=(300, 300, 3))

# Load Weights for the InceptionV3 Model
base_model.load_weights('/content/drive/MyDrive/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5')

# Setting the Training of all layers of InceptionV3 model to false
base_model.trainable = False
```

```
# Adding some more layers at the end of the Model as per our requirement
model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dropout(0.15),
    Dense(1024, activation='relu'),
    Dense(3, activation='softmax')
])
```

```
opt = optimizers.Adam(learning_rate=0.0001)

# Compiling and setting the parameters we want our model to use
model.compile(loss="categorical_crossentropy", optimizer=opt, metrics=['accuracy'])
```

```
from keras.utils.vis_utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```

```
# Setting variables for the model
batch_size = 64
epochs = 5

# Seperating Training and Testing Data
train_generator = datagenerator["train"]
valid_generator = datagenerator["valid"]
```

```
# Calculating variables for the model
steps_per_epoch = train_generator.n // batch_size
validation_steps = valid_generator.n // batch_size

print("steps_per_epoch :", steps_per_epoch)
print("validation_steps :", validation_steps)
```

```
# File Path to store the trained models
filepath = "/content/drive/MyDrive/model_{epoch:02d}-{val_accuracy:.2f}.h5"

# Using the ModelCheckpoint function to train and store all the best models
checkpoint1 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

callbacks_list = [checkpoint1]
# Training the Model
history = model.fit_generator(generator=train_generator, epochs=epochs, steps_per_epoch=steps_per_epoch,
                              validation_data=valid_generator, validation_steps=validation_steps,
                              callbacks=callbacks_list)
```

```
acc = history.history['accuracy']
val_acc = history.history['val_accuracy']

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

# ________________ Graph 1 -------------------------

plt.figure(figsize=(8, 8))
plt.subplot(2, 1, 1)
plt.plot(acc, label='Training Accuracy')
plt.plot(val_acc, label='Validation Accuracy')
plt.legend(loc='lower right')
```

```
plt.ylabel('Accuracy')
plt.ylim([min(plt.ylim()),1])
plt.title('Training and Validation Accuracy')

# ________________ Graph 2 -------------------------

plt.subplot(2, 1, 2)
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.ylabel('Cross Entropy')
plt.ylim([0,max(plt.ylim())])
plt.title('Training and Validation Loss')
plt.show()
```

```
test_loss, test_acc = model.evaluate(valid_generator)
print('test accuracy : ', test_acc)
```

```
# Check our folder and import the model with best validation accuracy
loaded_best_model = keras.models.load_model("/content/drive/MyDrive/model_04-1.00.h5")

# Custom function to load and predict label for the image
def predict(img_rel_path):
    # Import Image from the path with size of (300, 300)
    img = image.load_img(img_rel_path, target_size=(300, 300))

    # Convert Image to a numpy array
    img = image.img_to_array(img, dtype=np.uint8)

    # Scaling the Image Array values between 0 and 1
    img = np.array(img)/255.0

    # Plotting the Loaded Image
    plt.title("Loaded Image")
    plt.axis('off')
    plt.imshow(img.squeeze())
    plt.show()

    # Get the Predicted Label for the loaded Image
    p = loaded_best_model.predict(img[np.newaxis, ...])

    # Label array
    labels = {0: 'cans', 1: 'glass_bottles',2:'plastic_bottles'}

    print("\n\nMaximum Probability: ", np.max(p[0], axis=-1))
    predicted_class = labels[np.argmax(p[0], axis=-1)]
    print("Classified:", predicted_class, "\n\n")

    classes=[]
    prob=[]
    print("\n-------------------Individual Probability--------------------------------\n")

    for i,j in enumerate (p[0],0):
        print(labels[i].upper(),':',round(j*100,2),'%')
        classes.append(labels[i])
        prob.append(round(j*100,2))

    def plot_bar_x():
        # this is for plotting purpose
        index = np.arange(len(classes))
        plt.bar(index, prob)
        plt.xlabel('Labels', fontsize=8)
        plt.ylabel('Probability', fontsize=8)
        plt.xticks(index, classes, fontsize=8, rotation=20)
        plt.title('Probability for loaded image')
        plt.show()
    plot_bar_x()
```

```
predict("/content/drive/MyDrive/bottle/cans/cans/agfie.jpg")
```

```
predict("/content/drive/MyDrive/bottle/glass_bottles/glass_bottles/ahnxy.jpg")
```

```
predict("/content/drive/MyDrive/bottle/plastic_bottles/plastic_bottles/abwiq.jpg")
```

```
y_pred=loaded_best_model.predict(valid_generator)
```

my customised model-resnet-50

```
# resnet50
from tensorflow.keras.applications.inception_v3 import InceptionV3
```

```
train_dir = '/content/drive/MyDrive/bottle'
os.path.exists(train_dir)
```

```
True
```

```
from keras.callbacks import EarlyStopping
Callback = EarlyStopping(monitor = 'val_loss',
                          min_delta = 0,
                          patience = 5,
                          verbose = 1,
                          restore_best_weights = True)
```

```
# augmentation train only
train_datagen = ImageDataGenerator(rescale = 1./255.,
                                   validation_split=0.15,
                                   rotation_range = 40,
                                   width_shift_range = 0.2,
                                   height_shift_range = 0.2,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   fill_mode = 'nearest'
                                  )

validation_datagen = ImageDataGenerator(rescale = 1./255., validation_split=0.15)
```

```
HYP = dict(
        seed = 77,
        img_size = (225,225)
        )
```

```
# flow_from_directory
train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=HYP['img_size'],
                                                    shuffle=True,
                                                    seed=HYP['seed'],
                                                    class_mode='categorical',
                                                    subset="training")

validation_generator = validation_datagen.flow_from_directory(train_dir,
                                                              target_size=HYP['img_size'],
                                                              shuffle=False,
                                                              seed=HYP['seed'],
                                                              class_mode='categorical',
                                                              subset="validation")
```

```
Found 6984 images belonging to 3 classes.
Found 1231 images belonging to 3 classes.
```

```
# load the pre-trained ResNet50 model
base_model = InceptionV3(
                        include_top = False,
                        weights = "imagenet",
                        input_shape = None
)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_orderir
87910968/87910968 [==============================] - 4s 0us/step
```

```
# New Construction of Fully Connected Layer
from keras import regularizers
x = base_model.output
x = GlobalAveragePooling2D()(x)
```

```
x = Dense(512, activation='relu',kernel_regularizer= regularizers.l1(0.001))(x)
predictions = Dense(3, activation='softmax')(x)
```

```
from keras import regularizers

# network definition
model = Model(inputs = base_model.input, outputs = predictions)

# Train layer 250 and above
for layer in model.layers[:249]:
    layer.trainable = False

    # Batch Normalization improves the generalization performance of the model by updating parameters during training.
    if layer.name.startswith('batch_normalization'):
        layer.trainable = True

for layer in model.layers[249:]:
    layer.trainable = True

# After setting layer.trainable, be sure to compile.
model.compile(
    optimizer = Adam(),
    loss = 'categorical_crossentropy',
    metrics = ["accuracy"]
)
filepath = "/content/drive/MyDrive/newmodel_{epoch:02d}-{val_accuracy:.2f}.h5"

# Using the ModelCheckpoint function to train and store all the best models
checkpoint1 = ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1, save_best_only=True, mode='max')

callbacks_list = [checkpoint1]
```

```
model.summary()
from keras.utils.vis_utils import plot_model
plot_model(model, show_shapes=True, show_layer_names=True)
```

```
Model: "model_1"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, None, None,  0           []
                                 3)]

 conv2d (Conv2D)                (None, None, None,   864         ['input_1[0][0]']
                                32)

 batch_normalization (BatchNorm  (None, None, None,   96         ['conv2d[0][0]']
 alization)                     32)

 activation (Activation)        (None, None, None,   0           ['batch_normalization[0][0]']
                                32)

 conv2d_1 (Conv2D)              (None, None, None,   9216        ['activation[0][0]']
                                32)

 batch_normalization_1 (BatchNo  (None, None, None,   96         ['conv2d_1[0][0]']
 rmalization)                   32)

 activation_1 (Activation)      (None, None, None,   0           ['batch_normalization_1[0][0]']
                                32)

 conv2d_2 (Conv2D)              (None, None, None,   18432       ['activation_1[0][0]']
                                64)

 batch_normalization_2 (BatchNo  (None, None, None,   192        ['conv2d_2[0][0]']
 rmalization)                   64)

 activation_2 (Activation)      (None, None, None,   0           ['batch_normalization_2[0][0]']
                                64)

 max_pooling2d (MaxPooling2D)   (None, None, None,   0           ['activation_2[0][0]']
                                64)

 conv2d_3 (Conv2D)              (None, None, None,   5120        ['max_pooling2d[0][0]']
                                80)

 batch_normalization_3 (BatchNo  (None, None, None,   240        ['conv2d_3[0][0]']
 rmalization)                   80)

 activation_3 (Activation)      (None, None, None,   0           ['batch_normalization_3[0][0]']
                                80)

 conv2d_4 (Conv2D)              (None, None, None,   138240      ['activation_3[0][0]']
                                192)

 batch_normalization_4 (BatchNo  (None, None, None,   576        ['conv2d_4[0][0]']
 rmalization)                   192)

 activation_4 (Activation)      (None, None, None,   0           ['batch_normalization_4[0][0]']
                                192)

 max_pooling2d_1 (MaxPooling2D)  (None, None, None,   0          ['activation_4[0][0]']
                                192)

 conv2d_8 (Conv2D)              (None, None, None,   12288       ['max_pooling2d_1[0][0]']
                                64)

 batch_normalization_8 (BatchNo  (None, None, None,   192        ['conv2d_8[0][0]']
 rmalization)                   64)

 activation_8 (Activation)      (None, None, None,   0           ['batch_normalization_8[0][0]']
                                64)

 conv2d_6 (Conv2D)              (None, None, None,   9216        ['max_pooling2d_1[0][0]']
                                48)

 conv2d_9 (Conv2D)              (None, None, None,   55296       ['activation_8[0][0]']
                                96)

 batch_normalization_6 (BatchNo  (None, None, None,   144        ['conv2d_6[0][0]']
 rmalization)                   48)

 batch_normalization_9 (BatchNo  (None, None, None,   288        ['conv2d_9[0][0]']
 rmalization)                   96)

 activation_6 (Activation)      (None, None, None,   0           ['batch_normalization_6[0][0]']
                                48)

 activation_9 (Activation)      (None, None, None,   0           ['batch_normalization_9[0][0]']
                                96)

 average_pooling2d (AveragePool  (None, None, None,   0          ['max_pooling2d_1[0][0]']
 ing2D)                         192)

 conv2d_5 (Conv2D)              (None, None, None,   12288       ['max_pooling2d_1[0][0]']
                                64)
```

```
 conv2d_7 (Conv2D)              (None, None, None,   76800       ['activation_6[0][0]']
                                64)

 conv2d_10 (Conv2D)             (None, None, None,   82944       ['activation_9[0][0]']
                                96)

 conv2d_11 (Conv2D)             (None, None, None,   6144        ['average_pooling2d[0][0]']
                                32)

 batch_normalization_5 (BatchNo  (None, None, None,   192        ['conv2d_5[0][0]']
 rmalization)                   64)

 batch_normalization_7 (BatchNo  (None, None, None,   192        ['conv2d_7[0][0]']
 rmalization)                   64)

 batch_normalization_10 (BatchN  (None, None, None,   288        ['conv2d_10[0][0]']
 ormalization)                  96)

 batch_normalization_11 (BatchN  (None, None, None,   96         ['conv2d_11[0][0]']
 ormalization)                  32)

 activation_5 (Activation)      (None, None, None,   0           ['batch_normalization_5[0][0]']
                                64)

 activation_7 (Activation)      (None, None, None,   0           ['batch_normalization_7[0][0]']
                                64)

 activation_10 (Activation)     (None, None, None,   0           ['batch_normalization_10[0][0]']
                                96)

 activation_11 (Activation)     (None, None, None,   0           ['batch_normalization_11[0][0]']
                                32)

 mixed0 (Concatenate)           (None, None, None,   0           ['activation_5[0][0]',
                                256)                              'activation_7[0][0]',
                                                                  'activation_10[0][0]',
                                                                  'activation_11[0][0]']

 conv2d_15 (Conv2D)             (None, None, None,   16384       ['mixed0[0][0]']
                                64)

 batch_normalization_15 (BatchN  (None, None, None,   192        ['conv2d_15[0][0]']
 ormalization)                  64)

 activation_15 (Activation)     (None, None, None,   0           ['batch_normalization_15[0][0]']
                                64)

 conv2d_13 (Conv2D)             (None, None, None,   12288       ['mixed0[0][0]']
                                48)

 conv2d_16 (Conv2D)             (None, None, None,   55296       ['activation_15[0][0]']
                                96)

 batch_normalization_13 (BatchN  (None, None, None,   144        ['conv2d_13[0][0]']
 ormalization)                  48)

 batch_normalization_16 (BatchN  (None, None, None,   288        ['conv2d_16[0][0]']
 ormalization)                  96)

 activation_13 (Activation)     (None, None, None,   0           ['batch_normalization_13[0][0]']
                                48)

 activation_16 (Activation)     (None, None, None,   0           ['batch_normalization_16[0][0]']
                                96)

 average_pooling2d_1 (AveragePo  (None, None, None,   0          ['mixed0[0][0]']
 oling2D)                       256)

 conv2d_12 (Conv2D)             (None, None, None,   16384       ['mixed0[0][0]']
                                64)

 conv2d_14 (Conv2D)             (None, None, None,   76800       ['activation_13[0][0]']
                                64)

 conv2d_17 (Conv2D)             (None, None, None,   82944       ['activation_16[0][0]']
                                96)

 conv2d_18 (Conv2D)             (None, None, None,   16384       ['average_pooling2d_1[0][0]']
                                64)

 batch_normalization_12 (BatchN  (None, None, None,   192        ['conv2d_12[0][0]']
 ormalization)                  64)

 batch_normalization_14 (BatchN  (None, None, None,   192        ['conv2d_14[0][0]']
 ormalization)                  64)

 batch_normalization_17 (BatchN  (None, None, None,   288        ['conv2d_17[0][0]']
 ormalization)                  96)
```

```
 batch_normalization_18 (BatchN  (None, None, None,   192       ['conv2d_18[0][0]']
 ormalization)                  64)

 activation_12 (Activation)     (None, None, None,  0           ['batch_normalization_12[0][0]']
                                64)

 activation_14 (Activation)     (None, None, None,  0           ['batch_normalization_14[0][0]']
                                64)

 activation_17 (Activation)     (None, None, None,  0           ['batch_normalization_17[0][0]']
                                96)

 activation_18 (Activation)     (None, None, None,  0           ['batch_normalization_18[0][0]']
                                64)

 mixed1 (Concatenate)           (None, None, None,  0           ['activation_12[0][0]',
                                288)                             'activation_14[0][0]',
                                                                 'activation_17[0][0]',
                                                                 'activation_18[0][0]']

 conv2d_22 (Conv2D)             (None, None, None,  18432       ['mixed1[0][0]']
                                64)

 batch_normalization_22 (BatchN  (None, None, None,   192       ['conv2d_22[0][0]']
 ormalization)                  64)

 activation_22 (Activation)     (None, None, None,  0           ['batch_normalization_22[0][0]']
                                64)

 conv2d_20 (Conv2D)             (None, None, None,  13824       ['mixed1[0][0]']
                                48)

 conv2d_23 (Conv2D)             (None, None, None,  55296       ['activation_22[0][0]']
                                96)

 batch_normalization_20 (BatchN  (None, None, None,   144       ['conv2d_20[0][0]']
 ormalization)                  48)

 batch_normalization_23 (BatchN  (None, None, None,   288       ['conv2d_23[0][0]']
 ormalization)                  96)

 activation_20 (Activation)     (None, None, None,  0           ['batch_normalization_20[0][0]']
                                48)

 activation_23 (Activation)     (None, None, None,  0           ['batch_normalization_23[0][0]']
                                96)

 average_pooling2d_2 (AveragePo  (None, None, None,   0         ['mixed1[0][0]']
 oling2D)                       288)

 conv2d_19 (Conv2D)             (None, None, None,  18432       ['mixed1[0][0]']
                                64)

 conv2d_21 (Conv2D)             (None, None, None,  76800       ['activation_20[0][0]']
                                64)

 conv2d_24 (Conv2D)             (None, None, None,  82944       ['activation_23[0][0]']
                                96)

 conv2d_25 (Conv2D)             (None, None, None,  18432       ['average_pooling2d_2[0][0]']
                                64)

 batch_normalization_19 (BatchN  (None, None, None,   192       ['conv2d_19[0][0]']
 ormalization)                  64)

 batch_normalization_21 (BatchN  (None, None, None,   192       ['conv2d_21[0][0]']
 ormalization)                  64)

 batch_normalization_24 (BatchN  (None, None, None,   288       ['conv2d_24[0][0]']
 ormalization)                  96)

 batch_normalization_25 (BatchN  (None, None, None,   192       ['conv2d_25[0][0]']
 ormalization)                  64)

 activation_19 (Activation)     (None, None, None,  0           ['batch_normalization_19[0][0]']
                                64)

 activation_21 (Activation)     (None, None, None,  0           ['batch_normalization_21[0][0]']
                                64)

 activation_24 (Activation)     (None, None, None,  0           ['batch_normalization_24[0][0]']
                                96)

 activation_25 (Activation)     (None, None, None,  0           ['batch_normalization_25[0][0]']
                                64)

 mixed2 (Concatenate)           (None, None, None,  0           ['activation_19[0][0]',
                                288)                             'activation_21[0][0]',
                                                                 'activation_24[0][0]',
                                                                 'activation_25[0][0]']
```

```
 conv2d_27 (Conv2D)             (None, None, None,   18432       ['mixed2[0][0]']
                                64)

 batch_normalization_27 (BatchN  (None, None, None,   192        ['conv2d_27[0][0]']
 ormalization)                  64)

 activation_27 (Activation)     (None, None, None,   0           ['batch_normalization_27[0][0]']
                                64)

 conv2d_28 (Conv2D)             (None, None, None,   55296       ['activation_27[0][0]']
                                96)

 batch_normalization_28 (BatchN  (None, None, None,   288        ['conv2d_28[0][0]']
 ormalization)                  96)

 activation_28 (Activation)     (None, None, None,   0           ['batch_normalization_28[0][0]']
                                96)

 conv2d_26 (Conv2D)             (None, None, None,   995328      ['mixed2[0][0]']
                                384)

 conv2d_29 (Conv2D)             (None, None, None,   82944       ['activation_28[0][0]']
                                96)

 batch_normalization_26 (BatchN  (None, None, None,   1152       ['conv2d_26[0][0]']
 ormalization)                  384)

 batch_normalization_29 (BatchN  (None, None, None,   288        ['conv2d_29[0][0]']
 ormalization)                  96)

 activation_26 (Activation)     (None, None, None,   0           ['batch_normalization_26[0][0]']
                                384)

 activation_29 (Activation)     (None, None, None,   0           ['batch_normalization_29[0][0]']
                                96)

 max_pooling2d_2 (MaxPooling2D)  (None, None, None,   0          ['mixed2[0][0]']
                                288)

 mixed3 (Concatenate)           (None, None, None,   0           ['activation_26[0][0]',
                                768)                              'activation_29[0][0]',
                                                                  'max_pooling2d_2[0][0]']

 conv2d_34 (Conv2D)             (None, None, None,   98304       ['mixed3[0][0]']
                                128)

 batch_normalization_34 (BatchN  (None, None, None,   384        ['conv2d_34[0][0]']
 ormalization)                  128)

 activation_34 (Activation)     (None, None, None,   0           ['batch_normalization_34[0][0]']
                                128)

 conv2d_35 (Conv2D)             (None, None, None,   114688      ['activation_34[0][0]']
                                128)

 batch_normalization_35 (BatchN  (None, None, None,   384        ['conv2d_35[0][0]']
 ormalization)                  128)

 activation_35 (Activation)     (None, None, None,   0           ['batch_normalization_35[0][0]']
                                128)

 conv2d_31 (Conv2D)             (None, None, None,   98304       ['mixed3[0][0]']
                                128)

 conv2d_36 (Conv2D)             (None, None, None,   114688      ['activation_35[0][0]']
                                128)

 batch_normalization_31 (BatchN  (None, None, None,   384        ['conv2d_31[0][0]']
 ormalization)                  128)

 batch_normalization_36 (BatchN  (None, None, None,   384        ['conv2d_36[0][0]']
 ormalization)                  128)

 activation_31 (Activation)     (None, None, None,   0           ['batch_normalization_31[0][0]']
                                128)

 activation_36 (Activation)     (None, None, None,   0           ['batch_normalization_36[0][0]']
                                128)

 conv2d_32 (Conv2D)             (None, None, None,   114688      ['activation_31[0][0]']
                                128)

 conv2d_37 (Conv2D)             (None, None, None,   114688      ['activation_36[0][0]']
                                128)

 batch_normalization_32 (BatchN  (None, None, None,   384        ['conv2d_32[0][0]']
 ormalization)                  128)

 batch_normalization_37 (BatchN  (None, None, None,   384        ['conv2d_37[0][0]']
 ormalization)                  128)
```

```
 activation_32 (Activation)     (None, None, None,   0           ['batch_normalization_32[0][0]']
                                128)

 activation_37 (Activation)     (None, None, None,   0           ['batch_normalization_37[0][0]']
                                128)

 average_pooling2d_3 (AveragePo  (None, None, None,   0          ['mixed3[0][0]']
 oling2D)                       768)

 conv2d_30 (Conv2D)             (None, None, None,   147456      ['mixed3[0][0]']
                                192)

 conv2d_33 (Conv2D)             (None, None, None,   172032      ['activation_32[0][0]']
                                192)

 conv2d_38 (Conv2D)             (None, None, None,   172032      ['activation_37[0][0]']
                                192)

 conv2d_39 (Conv2D)             (None, None, None,   147456      ['average_pooling2d_3[0][0]']
                                192)

 batch_normalization_30 (BatchN  (None, None, None,   576        ['conv2d_30[0][0]']
 ormalization)                  192)

 batch_normalization_33 (BatchN  (None, None, None,   576        ['conv2d_33[0][0]']
 ormalization)                  192)

 batch_normalization_38 (BatchN  (None, None, None,   576        ['conv2d_38[0][0]']
 ormalization)                  192)

 batch_normalization_39 (BatchN  (None, None, None,   576        ['conv2d_39[0][0]']
 ormalization)                  192)

 activation_30 (Activation)     (None, None, None,   0           ['batch_normalization_30[0][0]']
                                192)

 activation_33 (Activation)     (None, None, None,   0           ['batch_normalization_33[0][0]']
                                192)

 activation_38 (Activation)     (None, None, None,   0           ['batch_normalization_38[0][0]']
                                192)

 activation_39 (Activation)     (None, None, None,   0           ['batch_normalization_39[0][0]']
                                192)

 mixed4 (Concatenate)           (None, None, None,   0           ['activation_30[0][0]',
                                768)                              'activation_33[0][0]',
                                                                  'activation_38[0][0]',
                                                                  'activation_39[0][0]']

 conv2d_44 (Conv2D)             (None, None, None,   122880      ['mixed4[0][0]']
                                160)

 batch_normalization_44 (BatchN  (None, None, None,   480        ['conv2d_44[0][0]']
 ormalization)                  160)

 activation_44 (Activation)     (None, None, None,   0           ['batch_normalization_44[0][0]']
                                160)

 conv2d_45 (Conv2D)             (None, None, None,   179200      ['activation_44[0][0]']
                                160)

 batch_normalization_45 (BatchN  (None, None, None,   480        ['conv2d_45[0][0]']
 ormalization)                  160)

 activation_45 (Activation)     (None, None, None,   0           ['batch_normalization_45[0][0]']
                                160)

 conv2d_41 (Conv2D)             (None, None, None,   122880      ['mixed4[0][0]']
                                160)

 conv2d_46 (Conv2D)             (None, None, None,   179200      ['activation_45[0][0]']
                                160)

 batch_normalization_41 (BatchN  (None, None, None,   480        ['conv2d_41[0][0]']
 ormalization)                  160)

 batch_normalization_46 (BatchN  (None, None, None,   480        ['conv2d_46[0][0]']
 ormalization)                  160)

 activation_41 (Activation)     (None, None, None,   0           ['batch_normalization_41[0][0]']
                                160)

 activation_46 (Activation)     (None, None, None,   0           ['batch_normalization_46[0][0]']
                                160)

 conv2d_42 (Conv2D)             (None, None, None,   179200      ['activation_41[0][0]']
                                160)
```

```
 conv2d_47 (Conv2D)             (None, None, None,   179200      ['activation_46[0][0]']
                                160)

 batch_normalization_42 (BatchN  (None, None, None,   480        ['conv2d_42[0][0]']
 ormalization)                  160)

 batch_normalization_47 (BatchN  (None, None, None,   480        ['conv2d_47[0][0]']
 ormalization)                  160)

 activation_42 (Activation)     (None, None, None,   0           ['batch_normalization_42[0][0]']
                                160)

 activation_47 (Activation)     (None, None, None,   0           ['batch_normalization_47[0][0]']
                                160)

 average_pooling2d_4 (AveragePo  (None, None, None,   0          ['mixed4[0][0]']
 oling2D)                       768)

 conv2d_40 (Conv2D)             (None, None, None,   147456      ['mixed4[0][0]']
                                192)

 conv2d_43 (Conv2D)             (None, None, None,   215040      ['activation_42[0][0]']
                                192)

 conv2d_48 (Conv2D)             (None, None, None,   215040      ['activation_47[0][0]']
                                192)

 conv2d_49 (Conv2D)             (None, None, None,   147456      ['average_pooling2d_4[0][0]']
                                192)

 batch_normalization_40 (BatchN  (None, None, None,   576        ['conv2d_40[0][0]']
 ormalization)                  192)

 batch_normalization_43 (BatchN  (None, None, None,   576        ['conv2d_43[0][0]']
 ormalization)                  192)

 batch_normalization_48 (BatchN  (None, None, None,   576        ['conv2d_48[0][0]']
 ormalization)                  192)

 batch_normalization_49 (BatchN  (None, None, None,   576        ['conv2d_49[0][0]']
 ormalization)                  192)

 activation_40 (Activation)     (None, None, None,   0           ['batch_normalization_40[0][0]']
                                192)

 activation_43 (Activation)     (None, None, None,   0           ['batch_normalization_43[0][0]']
                                192)

 activation_48 (Activation)     (None, None, None,   0           ['batch_normalization_48[0][0]']
                                192)

 activation_49 (Activation)     (None, None, None,   0           ['batch_normalization_49[0][0]']
                                192)

 mixed5 (Concatenate)           (None, None, None,   0           ['activation_40[0][0]',
                                768)                              'activation_43[0][0]',
                                                                  'activation_48[0][0]',
                                                                  'activation_49[0][0]']

 conv2d_54 (Conv2D)             (None, None, None,   122880      ['mixed5[0][0]']
                                160)

 batch_normalization_54 (BatchN  (None, None, None,   480        ['conv2d_54[0][0]']
 ormalization)                  160)

 activation_54 (Activation)     (None, None, None,   0           ['batch_normalization_54[0][0]']
                                160)

 conv2d_55 (Conv2D)             (None, None, None,   179200      ['activation_54[0][0]']
                                160)

 batch_normalization_55 (BatchN  (None, None, None,   480        ['conv2d_55[0][0]']
 ormalization)                  160)

 activation_55 (Activation)     (None, None, None,   0           ['batch_normalization_55[0][0]']
                                160)

 conv2d_51 (Conv2D)             (None, None, None,   122880      ['mixed5[0][0]']
                                160)

 conv2d_56 (Conv2D)             (None, None, None,   179200      ['activation_55[0][0]']
                                160)

 batch_normalization_51 (BatchN  (None, None, None,   480        ['conv2d_51[0][0]']
 ormalization)                  160)

 batch_normalization_56 (BatchN  (None, None, None,   480        ['conv2d_56[0][0]']
 ormalization)                  160)

 activation_51 (Activation)     (None, None, None,   0           ['batch_normalization_51[0][0]']
                                160)
```

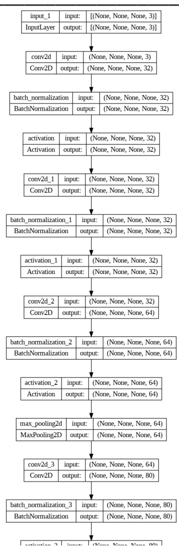
```
                                160)

 activation_56 (Activation)     (None, None, None,   0           ['batch_normalization_56[0][0]']
                                160)

 conv2d_52 (Conv2D)             (None, None, None,   179200      ['activation_51[0][0]']
                                160)

 conv2d_57 (Conv2D)             (None, None, None,   179200      ['activation_56[0][0]']
                                160)

 batch_normalization_52 (BatchN  (None, None, None,   480        ['conv2d_52[0][0]']
 ormalization)                  160)

 batch_normalization_57 (BatchN  (None, None, None,   480        ['conv2d_57[0][0]']
 ormalization)                  160)

 activation_52 (Activation)     (None, None, None,   0           ['batch_normalization_52[0][0]']
                                160)

 activation_57 (Activation)     (None, None, None,   0           ['batch_normalization_57[0][0]']
                                160)

 average_pooling2d_5 (AveragePo  (None, None, None,   0          ['mixed5[0][0]']
 oling2D)                       768)

 conv2d_50 (Conv2D)             (None, None, None,   147456      ['mixed5[0][0]']
                                192)

 conv2d_53 (Conv2D)             (None, None, None,   215040      ['activation_52[0][0]']
                                192)

 conv2d_58 (Conv2D)             (None, None, None,   215040      ['activation_57[0][0]']
                                192)

 conv2d_59 (Conv2D)             (None, None, None,   147456      ['average_pooling2d_5[0][0]']
                                192)

 batch_normalization_50 (BatchN  (None, None, None,   576        ['conv2d_50[0][0]']
 ormalization)                  192)

 batch_normalization_53 (BatchN  (None, None, None,   576        ['conv2d_53[0][0]']
 ormalization)                  192)

 batch_normalization_58 (BatchN  (None, None, None,   576        ['conv2d_58[0][0]']
 ormalization)                  192)

 batch_normalization_59 (BatchN  (None, None, None,   576        ['conv2d_59[0][0]']
 ormalization)                  192)

 activation_50 (Activation)     (None, None, None,   0           ['batch_normalization_50[0][0]']
                                192)

 activation_53 (Activation)     (None, None, None,   0           ['batch_normalization_53[0][0]']
                                192)

 activation_58 (Activation)     (None, None, None,   0           ['batch_normalization_58[0][0]']
                                192)

 activation_59 (Activation)     (None, None, None,   0           ['batch_normalization_59[0][0]']
                                192)

 mixed6 (Concatenate)           (None, None, None,   0           ['activation_50[0][0]',
                                768)                              'activation_53[0][0]',
                                                                  'activation_58[0][0]',
                                                                  'activation_59[0][0]']

 conv2d_64 (Conv2D)             (None, None, None,   147456      ['mixed6[0][0]']
                                192)

 batch_normalization_64 (BatchN  (None, None, None,   576        ['conv2d_64[0][0]']
 ormalization)                  192)

 activation_64 (Activation)     (None, None, None,   0           ['batch_normalization_64[0][0]']
                                192)

 conv2d_65 (Conv2D)             (None, None, None,   258048      ['activation_64[0][0]']
                                192)

 batch_normalization_65 (BatchN  (None, None, None,   576        ['conv2d_65[0][0]']
 ormalization)                  192)

 activation_65 (Activation)     (None, None, None,   0           ['batch_normalization_65[0][0]']
                                192)

 conv2d_61 (Conv2D)             (None, None, None,   147456      ['mixed6[0][0]']
                                192)

 conv2d_66 (Conv2D)             (None, None, None,   258048      ['activation_65[0][0]']
                                192)
```

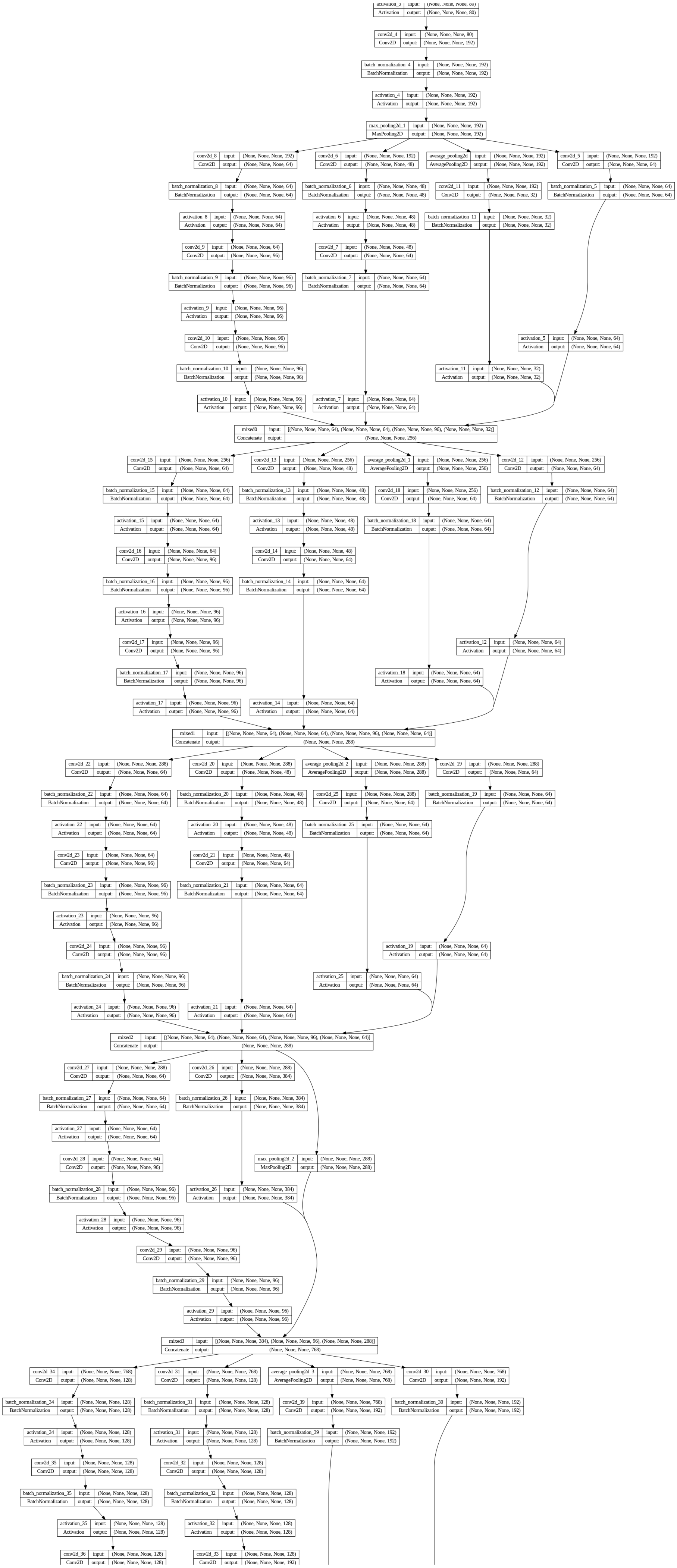
```
batch_normalization_61 (BatchN  (None, None, None,   576        ['conv2d_61[0][0]']
ormalization)                  192)

batch_normalization_66 (BatchN  (None, None, None,   576        ['conv2d_66[0][0]']
ormalization)                  192)

activation_61 (Activation)     (None, None, None,   0          ['batch_normalization_61[0][0]']
                               192)

activation_66 (Activation)     (None, None, None,   0          ['batch_normalization_66[0][0]']
                               192)

conv2d_62 (Conv2D)             (None, None, None,   258048     ['activation_61[0][0]']
                               192)

conv2d_67 (Conv2D)             (None, None, None,   258048     ['activation_66[0][0]']
                               192)

batch_normalization_62 (BatchN  (None, None, None,   576        ['conv2d_62[0][0]']
ormalization)                  192)

batch_normalization_67 (BatchN  (None, None, None,   576        ['conv2d_67[0][0]']
ormalization)                  192)

activation_62 (Activation)     (None, None, None,   0          ['batch_normalization_62[0][0]']
                               192)

activation_67 (Activation)     (None, None, None,   0          ['batch_normalization_67[0][0]']
                               192)

average_pooling2d_6 (AveragePo  (None, None, None,   0          ['mixed6[0][0]']
oling2D)                       768)

conv2d_60 (Conv2D)             (None, None, None,   147456     ['mixed6[0][0]']
                               192)

conv2d_63 (Conv2D)             (None, None, None,   258048     ['activation_62[0][0]']
                               192)

conv2d_68 (Conv2D)             (None, None, None,   258048     ['activation_67[0][0]']
                               192)

conv2d_69 (Conv2D)             (None, None, None,   147456     ['average_pooling2d_6[0][0]']
                               192)

batch_normalization_60 (BatchN  (None, None, None,   576        ['conv2d_60[0][0]']
ormalization)                  192)

batch_normalization_63 (BatchN  (None, None, None,   576        ['conv2d_63[0][0]']
ormalization)                  192)

batch_normalization_68 (BatchN  (None, None, None,   576        ['conv2d_68[0][0]']
ormalization)                  192)

batch_normalization_69 (BatchN  (None, None, None,   576        ['conv2d_69[0][0]']
ormalization)                  192)

activation_60 (Activation)     (None, None, None,   0          ['batch_normalization_60[0][0]']
                               192)

activation_63 (Activation)     (None, None, None,   0          ['batch_normalization_63[0][0]']
                               192)

activation_68 (Activation)     (None, None, None,   0          ['batch_normalization_68[0][0]']
                               192)

activation_69 (Activation)     (None, None, None,   0          ['batch_normalization_69[0][0]']
                               192)

mixed7 (Concatenate)           (None, None, None,   0          ['activation_60[0][0]',
                               768)                             'activation_63[0][0]',
                                                                'activation_68[0][0]',
                                                                'activation_69[0][0]']

conv2d_72 (Conv2D)             (None, None, None,   147456     ['mixed7[0][0]']
                               192)

batch_normalization_72 (BatchN  (None, None, None,   576        ['conv2d_72[0][0]']
ormalization)                  192)

activation_72 (Activation)     (None, None, None,   0          ['batch_normalization_72[0][0]']
                               192)

conv2d_73 (Conv2D)             (None, None, None,   258048     ['activation_72[0][0]']
                               192)

batch_normalization_73 (BatchN  (None, None, None,   576        ['conv2d_73[0][0]']
ormalization)                  192)

activation_73 (Activation)     (None, None, None,   0          ['batch_normalization_73[0][0]']
```

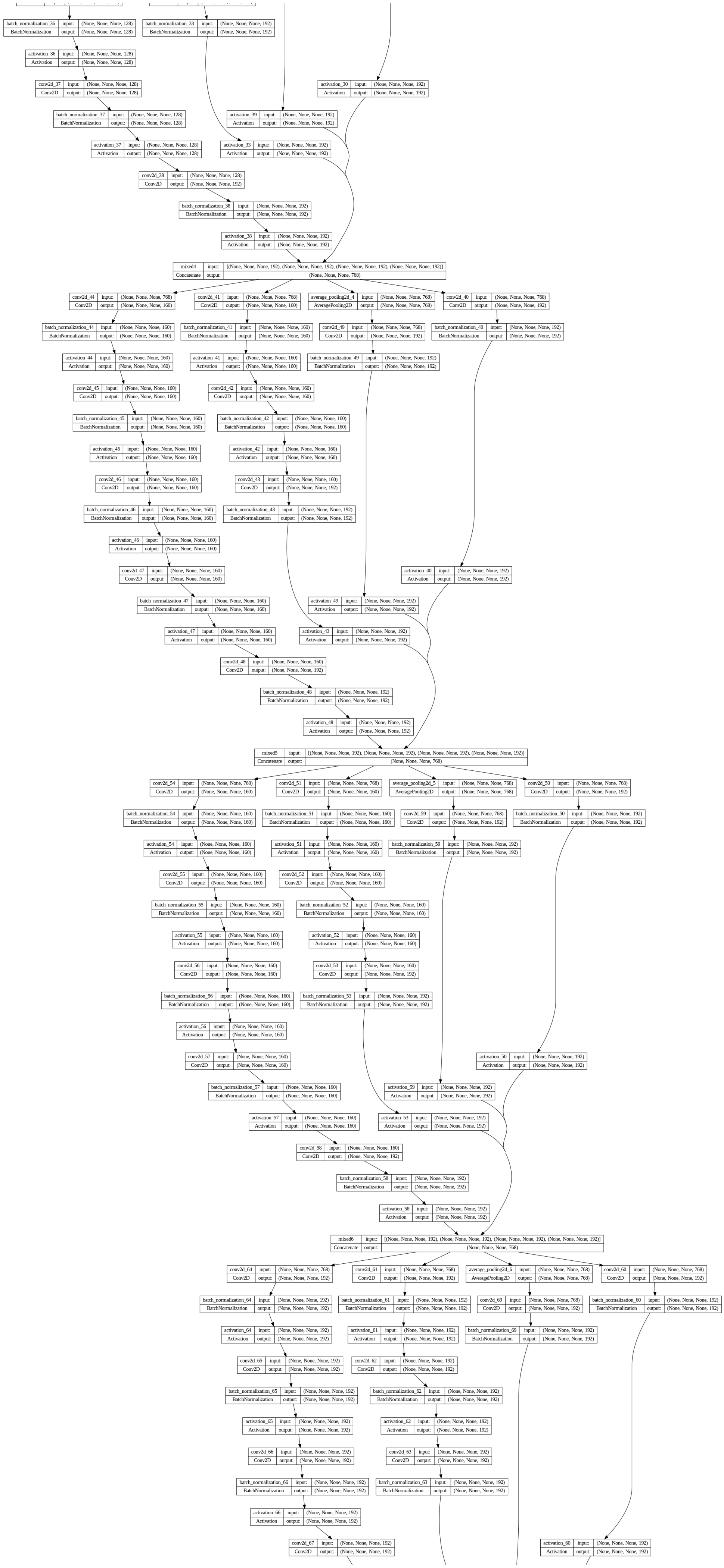
```
                                192)

 conv2d_70 (Conv2D)             (None, None, None,   147456      ['mixed7[0][0]']
                                192)

 conv2d_74 (Conv2D)             (None, None, None,   258048      ['activation_73[0][0]']
                                192)

 batch_normalization_70 (BatchN  (None, None, None,   576        ['conv2d_70[0][0]']
 ormalization)                  192)

 batch_normalization_74 (BatchN  (None, None, None,   576        ['conv2d_74[0][0]']
 ormalization)                  192)

 activation_70 (Activation)     (None, None, None,   0           ['batch_normalization_70[0][0]']
                                192)

 activation_74 (Activation)     (None, None, None,   0           ['batch_normalization_74[0][0]']
                                192)

 conv2d_71 (Conv2D)             (None, None, None,   552960      ['activation_70[0][0]']
                                320)

 conv2d_75 (Conv2D)             (None, None, None,   331776      ['activation_74[0][0]']
                                192)

 batch_normalization_71 (BatchN  (None, None, None,   960        ['conv2d_71[0][0]']
 ormalization)                  320)

 batch_normalization_75 (BatchN  (None, None, None,   576        ['conv2d_75[0][0]']
 ormalization)                  192)

 activation_71 (Activation)     (None, None, None,   0           ['batch_normalization_71[0][0]']
                                320)

 activation_75 (Activation)     (None, None, None,   0           ['batch_normalization_75[0][0]']
                                192)

 max_pooling2d_3 (MaxPooling2D)  (None, None, None,   0          ['mixed7[0][0]']
                                768)

 mixed8 (Concatenate)           (None, None, None,   0           ['activation_71[0][0]',
                                1280)                             'activation_75[0][0]',
                                                                  'max_pooling2d_3[0][0]']

 conv2d_80 (Conv2D)             (None, None, None,   573440      ['mixed8[0][0]']
                                448)

 batch_normalization_80 (BatchN  (None, None, None,   1344       ['conv2d_80[0][0]']
 ormalization)                  448)

 activation_80 (Activation)     (None, None, None,   0           ['batch_normalization_80[0][0]']
                                448)

 conv2d_77 (Conv2D)             (None, None, None,   491520      ['mixed8[0][0]']
                                384)

 conv2d_81 (Conv2D)             (None, None, None,   1548288     ['activation_80[0][0]']
                                384)

 batch_normalization_77 (BatchN  (None, None, None,   1152       ['conv2d_77[0][0]']
 ormalization)                  384)

 batch_normalization_81 (BatchN  (None, None, None,   1152       ['conv2d_81[0][0]']
 ormalization)                  384)

 activation_77 (Activation)     (None, None, None,   0           ['batch_normalization_77[0][0]']
                                384)

 activation_81 (Activation)     (None, None, None,   0           ['batch_normalization_81[0][0]']
                                384)

 conv2d_78 (Conv2D)             (None, None, None,   442368      ['activation_77[0][0]']
                                384)

 conv2d_79 (Conv2D)             (None, None, None,   442368      ['activation_77[0][0]']
                                384)

 conv2d_82 (Conv2D)             (None, None, None,   442368      ['activation_81[0][0]']
                                384)

 conv2d_83 (Conv2D)             (None, None, None,   442368      ['activation_81[0][0]']
                                384)

 average_pooling2d_7 (AveragePo  (None, None, None,   0          ['mixed8[0][0]']
 oling2D)                       1280)

 conv2d_76 (Conv2D)             (None, None, None,   409600      ['mixed8[0][0]']
                                320)

 batch_normalization_78 (BatchN  (None, None, None,   1152       ['conv2d_78[0][0]']
```

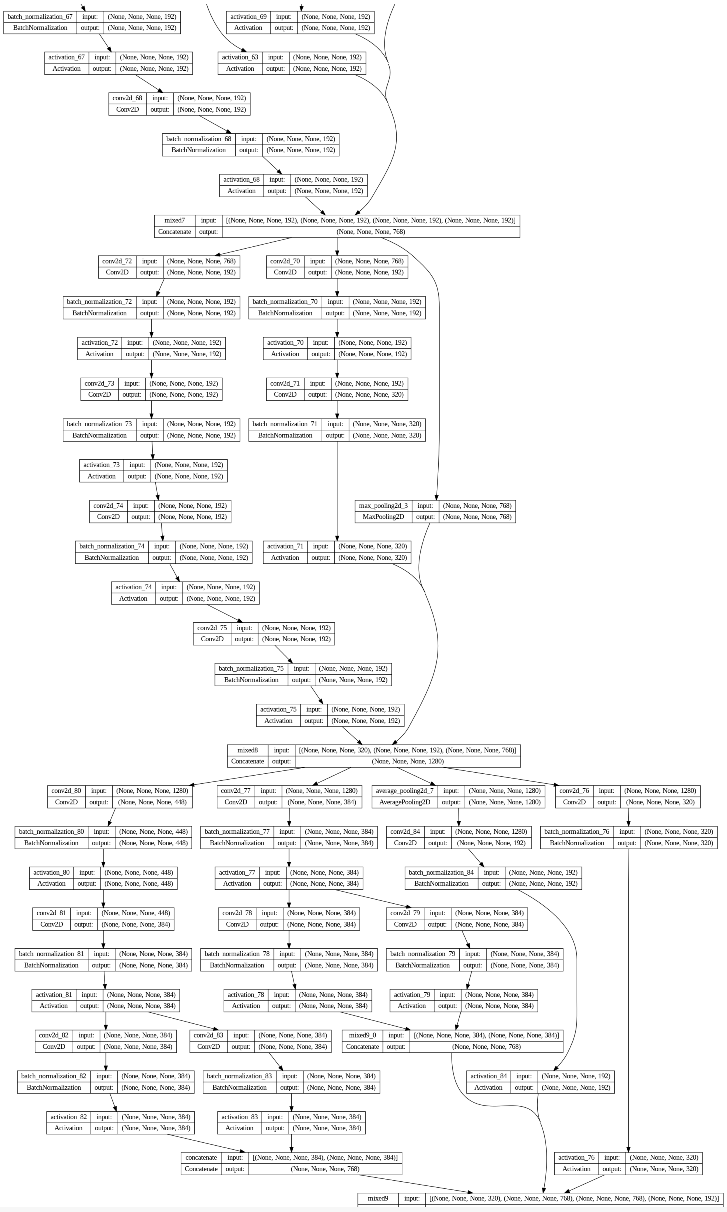
```
 batch_normalization_78 (BatchN  (None, None, None,   1152       ['conv2d_78[0][0]']
 ormalization)                  384)

 batch_normalization_79 (BatchN  (None, None, None,   1152       ['conv2d_79[0][0]']
 ormalization)                  384)

 batch_normalization_82 (BatchN  (None, None, None,   1152       ['conv2d_82[0][0]']
 ormalization)                  384)

 batch_normalization_83 (BatchN  (None, None, None,   1152       ['conv2d_83[0][0]']
 ormalization)                  384)

 conv2d_84 (Conv2D)             (None, None, None,   245760      ['average_pooling2d_7[0][0]']
                                192)

 batch_normalization_76 (BatchN  (None, None, None,   960        ['conv2d_76[0][0]']
 ormalization)                  320)

 activation_78 (Activation)     (None, None, None,   0           ['batch_normalization_78[0][0]']
                                384)

 activation_79 (Activation)     (None, None, None,   0           ['batch_normalization_79[0][0]']
                                384)

 activation_82 (Activation)     (None, None, None,   0           ['batch_normalization_82[0][0]']
                                384)

 activation_83 (Activation)     (None, None, None,   0           ['batch_normalization_83[0][0]']
                                384)

 batch_normalization_84 (BatchN  (None, None, None,   576        ['conv2d_84[0][0]']
 ormalization)                  192)

 activation_76 (Activation)     (None, None, None,   0           ['batch_normalization_76[0][0]']
                                320)

 mixed9_0 (Concatenate)         (None, None, None,   0           ['activation_78[0][0]',
                                768)                              'activation_79[0][0]']

 concatenate (Concatenate)      (None, None, None,   0           ['activation_82[0][0]',
                                768)                              'activation_83[0][0]']

 activation_84 (Activation)     (None, None, None,   0           ['batch_normalization_84[0][0]']
                                192)

 mixed9 (Concatenate)           (None, None, None,   0           ['activation_76[0][0]',
                                2048)                             'mixed9_0[0][0]',
                                                                  'concatenate[0][0]',
                                                                  'activation_84[0][0]']

 conv2d_89 (Conv2D)             (None, None, None,   917504      ['mixed9[0][0]']
                                448)

 batch_normalization_89 (BatchN  (None, None, None,   1344       ['conv2d_89[0][0]']
 ormalization)                  448)

 activation_89 (Activation)     (None, None, None,   0           ['batch_normalization_89[0][0]']
                                448)

 conv2d_86 (Conv2D)             (None, None, None,   786432      ['mixed9[0][0]']
                                384)

 conv2d_90 (Conv2D)             (None, None, None,   1548288     ['activation_89[0][0]']
                                384)

 batch_normalization_86 (BatchN  (None, None, None,   1152       ['conv2d_86[0][0]']
 ormalization)                  384)

 batch_normalization_90 (BatchN  (None, None, None,   1152       ['conv2d_90[0][0]']
 ormalization)                  384)

 activation_86 (Activation)     (None, None, None,   0           ['batch_normalization_86[0][0]']
                                384)

 activation_90 (Activation)     (None, None, None,   0           ['batch_normalization_90[0][0]']
                                384)

 conv2d_87 (Conv2D)             (None, None, None,   442368      ['activation_86[0][0]']
                                384)

 conv2d_88 (Conv2D)             (None, None, None,   442368      ['activation_86[0][0]']
                                384)

 conv2d_91 (Conv2D)             (None, None, None,   442368      ['activation_90[0][0]']
                                384)

 conv2d_92 (Conv2D)             (None, None, None,   442368      ['activation_90[0][0]']
                                384)

 average_pooling2d_8 (AveragePo  (None, None, None,   0          ['mixed9[0][0]']
 oling2D)                       2048)
```

```
 conv2d_85 (Conv2D)             (None, None, None,   655360      ['mixed9[0][0]']
                                320)

 batch_normalization_87 (BatchN  (None, None, None,   1152       ['conv2d_87[0][0]']
 ormalization)                  384)

 batch_normalization_88 (BatchN  (None, None, None,   1152       ['conv2d_88[0][0]']
 ormalization)                  384)

 batch_normalization_91 (BatchN  (None, None, None,   1152       ['conv2d_91[0][0]']
 ormalization)                  384)

 batch_normalization_92 (BatchN  (None, None, None,   1152       ['conv2d_92[0][0]']
 ormalization)                  384)

 conv2d_93 (Conv2D)             (None, None, None,   393216      ['average_pooling2d_8[0][0]']
                                192)

 batch_normalization_85 (BatchN  (None, None, None,   960        ['conv2d_85[0][0]']
 ormalization)                  320)

 activation_87 (Activation)     (None, None, None,   0           ['batch_normalization_87[0][0]']
                                384)

 activation_88 (Activation)     (None, None, None,   0           ['batch_normalization_88[0][0]']
                                384)

 activation_91 (Activation)     (None, None, None,   0           ['batch_normalization_91[0][0]']
                                384)

 activation_92 (Activation)     (None, None, None,   0           ['batch_normalization_92[0][0]']
                                384)

 batch_normalization_93 (BatchN  (None, None, None,   576        ['conv2d_93[0][0]']
 ormalization)                  192)

 activation_85 (Activation)     (None, None, None,   0           ['batch_normalization_85[0][0]']
                                320)

 mixed9_1 (Concatenate)         (None, None, None,   0           ['activation_87[0][0]',
                                768)                              'activation_88[0][0]']

 concatenate_1 (Concatenate)    (None, None, None,   0           ['activation_91[0][0]',
                                768)                              'activation_92[0][0]']

 activation_93 (Activation)     (None, None, None,   0           ['batch_normalization_93[0][0]']
                                192)

 mixed10 (Concatenate)          (None, None, None,   0           ['activation_85[0][0]',
                                2048)                             'mixed9_1[0][0]',
                                                                  'concatenate_1[0][0]',
                                                                  'activation_93[0][0]']

 global_average_pooling2d (Glob  (None, 2048)        0           ['mixed10[0][0]']
 alAveragePooling2D)

 dense (Dense)                  (None, 512)          1049088     ['global_average_pooling2d[0][0]'
                                                                 ]

 dense_1 (Dense)                (None, 3)            1539        ['dense[0][0]']

==================================================================================================
Total params: 22,853,411
Trainable params: 12,176,195
Non-trainable params: 10,677,216
__________________________________________________________________________________________________
```




activation_3 Activation input: (None, None, None, 80) output: (None, None, None, 80)
conv2d_4 Conv2D input: (None, None, None, 80) output: (None, None, None, 192)
batch_normalization_4 BatchNormalization input: (None, None, None, 192) output: (None, None, None, 192)
activation_4 Activation input: (None, None, None, 192) output: (None, None, None, 192)
max_pooling2d_1 MaxPooling2D input: (None, None, None, 192) output: (None, None, None, 192)
conv2d_8 Conv2D input: (None, None, None, 192) output: (None, None, None, 64)
conv2d_6 Conv2D input: (None, None, None, 192) output: (None, None, None, 48)
average_pooling2d AveragePooling2D input: (None, None, None, 192) output: (None, None, None, 192)
conv2d_5 Conv2D input: (None, None, None, 192) output: (None, None, None, 64)
batch_normalization_8 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
batch_normalization_6 BatchNormalization input: (None, None, None, 48) output: (None, None, None, 48)
conv2d_11 Conv2D input: (None, None, None, 192) output: (None, None, None, 32)
batch_normalization_5 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
activation_8 Activation input: (None, None, None, 64) output: (None, None, None, 64)
activation_6 Activation input: (None, None, None, 48) output: (None, None, None, 48)
batch_normalization_11 BatchNormalization input: (None, None, None, 32) output: (None, None, None, 32)
conv2d_9 Conv2D input: (None, None, None, 64) output: (None, None, None, 96)
conv2d_7 Conv2D input: (None, None, None, 48) output: (None, None, None, 64)
batch_normalization_9 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
batch_normalization_7 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
activation_9 Activation input: (None, None, None, 96) output: (None, None, None, 96)
conv2d_10 Conv2D input: (None, None, None, 96) output: (None, None, None, 96)
activation_5 Activation input: (None, None, None, 64) output: (None, None, None, 64)
batch_normalization_10 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
activation_11 Activation input: (None, None, None, 32) output: (None, None, None, 32)
activation_10 Activation input: (None, None, None, 96) output: (None, None, None, 96)
activation_7 Activation input: (None, None, None, 64) output: (None, None, None, 64)
mixed0 Concatenate input: [(None, None, None, 64), (None, None, None, 64), (None, None, None, 96), (None, None, None, 32)] output: (None, None, None, 256)
conv2d_15 Conv2D input: (None, None, None, 256) output: (None, None, None, 64)
conv2d_13 Conv2D input: (None, None, None, 256) output: (None, None, None, 48)
average_pooling2d_1 AveragePooling2D input: (None, None, None, 256) output: (None, None, None, 256)
conv2d_12 Conv2D input: (None, None, None, 256) output: (None, None, None, 64)
batch_normalization_15 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
batch_normalization_13 BatchNormalization input: (None, None, None, 48) output: (None, None, None, 48)
conv2d_18 Conv2D input: (None, None, None, 256) output: (None, None, None, 64)
batch_normalization_12 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
activation_15 Activation input: (None, None, None, 64) output: (None, None, None, 64)
activation_13 Activation input: (None, None, None, 48) output: (None, None, None, 48)
batch_normalization_18 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
conv2d_16 Conv2D input: (None, None, None, 64) output: (None, None, None, 96)
conv2d_14 Conv2D input: (None, None, None, 48) output: (None, None, None, 64)
batch_normalization_16 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
batch_normalization_14 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
activation_16 Activation input: (None, None, None, 96) output: (None, None, None, 96)
conv2d_17 Conv2D input: (None, None, None, 96) output: (None, None, None, 96)
activation_12 Activation input: (None, None, None, 64) output: (None, None, None, 64)
batch_normalization_17 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
activation_18 Activation input: (None, None, None, 64) output: (None, None, None, 64)
activation_17 Activation input: (None, None, None, 96) output: (None, None, None, 96)
activation_14 Activation input: (None, None, None, 64) output: (None, None, None, 64)
mixed1 Concatenate input: [(None, None, None, 64), (None, None, None, 64), (None, None, None, 96), (None, None, None, 64)] output: (None, None, None, 288)
conv2d_22 Conv2D input: (None, None, None, 288) output: (None, None, None, 64)
conv2d_20 Conv2D input: (None, None, None, 288) output: (None, None, None, 48)
average_pooling2d_2 AveragePooling2D input: (None, None, None, 288) output: (None, None, None, 288)
conv2d_19 Conv2D input: (None, None, None, 288) output: (None, None, None, 64)
batch_normalization_22 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
batch_normalization_20 BatchNormalization input: (None, None, None, 48) output: (None, None, None, 48)
conv2d_25 Conv2D input: (None, None, None, 288) output: (None, None, None, 64)
batch_normalization_19 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
activation_22 Activation input: (None, None, None, 64) output: (None, None, None, 64)
activation_20 Activation input: (None, None, None, 48) output: (None, None, None, 48)
batch_normalization_25 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
conv2d_23 Conv2D input: (None, None, None, 64) output: (None, None, None, 96)
conv2d_21 Conv2D input: (None, None, None, 48) output: (None, None, None, 64)
batch_normalization_23 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
batch_normalization_21 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
activation_23 Activation input: (None, None, None, 96) output: (None, None, None, 96)
conv2d_24 Conv2D input: (None, None, None, 96) output: (None, None, None, 96)
activation_19 Activation input: (None, None, None, 64) output: (None, None, None, 64)
batch_normalization_24 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
activation_25 Activation input: (None, None, None, 64) output: (None, None, None, 64)
activation_24 Activation input: (None, None, None, 96) output: (None, None, None, 96)
activation_21 Activation input: (None, None, None, 64) output: (None, None, None, 64)
mixed2 Concatenate input: [(None, None, None, 64), (None, None, None, 64), (None, None, None, 96), (None, None, None, 64)] output: (None, None, None, 288)
conv2d_27 Conv2D input: (None, None, None, 288) output: (None, None, None, 64)
conv2d_26 Conv2D input: (None, None, None, 288) output: (None, None, None, 384)
batch_normalization_27 BatchNormalization input: (None, None, None, 64) output: (None, None, None, 64)
batch_normalization_26 BatchNormalization input: (None, None, None, 384) output: (None, None, None, 384)
activation_27 Activation input: (None, None, None, 64) output: (None, None, None, 64)
conv2d_28 Conv2D input: (None, None, None, 64) output: (None, None, None, 96)
max_pooling2d_2 MaxPooling2D input: (None, None, None, 288) output: (None, None, None, 288)
batch_normalization_28 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
activation_26 Activation input: (None, None, None, 384) output: (None, None, None, 384)
activation_28 Activation input: (None, None, None, 96) output: (None, None, None, 96)
conv2d_29 Conv2D input: (None, None, None, 96) output: (None, None, None, 96)
batch_normalization_29 BatchNormalization input: (None, None, None, 96) output: (None, None, None, 96)
activation_29 Activation input: (None, None, None, 96) output: (None, None, None, 96)
mixed3 Concatenate input: [(None, None, None, 384), (None, None, None, 96), (None, None, None, 288)] output: (None, None, None, 768)
conv2d_34 Conv2D input: (None, None, None, 768) output: (None, None, None, 128)
conv2d_31 Conv2D input: (None, None, None, 768) output: (None, None, None, 128)
average_pooling2d_3 AveragePooling2D input: (None, None, None, 768) output: (None, None, None, 768)
conv2d_30 Conv2D input: (None, None, None, 768) output: (None, None, None, 192)
batch_normalization_34 BatchNormalization input: (None, None, None, 128) output: (None, None, None, 128)
batch_normalization_31 BatchNormalization input: (None, None, None, 128) output: (None, None, None, 128)
conv2d_39 Conv2D input: (None, None, None, 768) output: (None, None, None, 192)
batch_normalization_30 BatchNormalization input: (None, None, None, 192) output: (None, None, None, 192)
activation_34 Activation input: (None, None, None, 128) output: (None, None, None, 128)
activation_31 Activation input: (None, None, None, 128) output: (None, None, None, 128)
batch_normalization_39 BatchNormalization input: (None, None, None, 192) output: (None, None, None, 192)
conv2d_35 Conv2D input: (None, None, None, 128) output: (None, None, None, 128)
conv2d_32 Conv2D input: (None, None, None, 128) output: (None, None, None, 128)
batch_normalization_35 BatchNormalization input: (None, None, None, 128) output: (None, None, None, 128)
batch_normalization_32 BatchNormalization input: (None, None, None, 128) output: (None, None, None, 128)
activation_35 Activation input: (None, None, None, 128) output: (None, None, None, 128)
activation_32 Activation input: (None, None, None, 128) output: (None, None, None, 128)
conv2d_36 Conv2D input: (None, None, None, 128) output: (None, None, None, 128)
conv2d_33 Conv2D input: (None, None, None, 128) output: (None, None, None, 192)



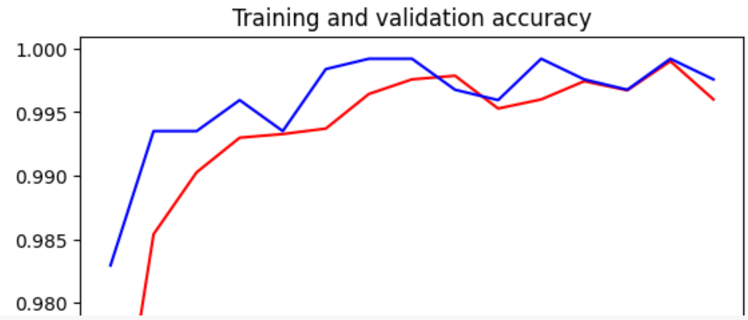


```
fit_history = model.fit(
                        train_generator,
                        validation_data=validation_generator,
                        callbacks=callbacks_list,
                        epochs=15,
                        verbose=1
                        )
```

```
Epoch 1/15
219/219 [==============================] - ETA: 0s - loss: 4.2821 - accuracy: 0.9639
Epoch 1: val_accuracy improved from -inf to 0.98294, saving model to /content/drive/MyDrive/newmodel_01-0.98.h5
219/219 [==============================] - 210s 748ms/step - loss: 4.2821 - accuracy: 0.9639 - val_loss: 0.2729 - val_accuracy:
Epoch 2/15
219/219 [==============================] - ETA: 0s - loss: 0.2457 - accuracy: 0.9854
Epoch 2: val_accuracy improved from 0.98294 to 0.99350, saving model to /content/drive/MyDrive/newmodel_02-0.99.h5
219/219 [==============================] - 158s 719ms/step - loss: 0.2457 - accuracy: 0.9854 - val_loss: 0.1952 - val_accuracy:
Epoch 3/15
219/219 [==============================] - ETA: 0s - loss: 0.2005 - accuracy: 0.9903
Epoch 3: val_accuracy did not improve from 0.99350
219/219 [==============================] - 152s 693ms/step - loss: 0.2005 - accuracy: 0.9903 - val_loss: 0.1632 - val_accuracy:
Epoch 4/15
219/219 [==============================] - ETA: 0s - loss: 0.1820 - accuracy: 0.9930
Epoch 4: val_accuracy improved from 0.99350 to 0.99594, saving model to /content/drive/MyDrive/newmodel_04-1.00.h5
219/219 [==============================] - 153s 697ms/step - loss: 0.1820 - accuracy: 0.9930 - val_loss: 0.1648 - val_accuracy:
Epoch 5/15
219/219 [==============================] - ETA: 0s - loss: 0.1764 - accuracy: 0.9933
Epoch 5: val_accuracy did not improve from 0.99594
219/219 [==============================] - 142s 648ms/step - loss: 0.1764 - accuracy: 0.9933 - val_loss: 0.1658 - val_accuracy:
Epoch 6/15
219/219 [==============================] - ETA: 0s - loss: 0.1720 - accuracy: 0.9937
Epoch 6: val_accuracy improved from 0.99594 to 0.99838, saving model to /content/drive/MyDrive/newmodel_06-1.00.h5
219/219 [==============================] - 166s 757ms/step - loss: 0.1720 - accuracy: 0.9937 - val_loss: 0.1528 - val_accuracy:
Epoch 7/15
219/219 [==============================] - ETA: 0s - loss: 0.1610 - accuracy: 0.9964
Epoch 7: val_accuracy improved from 0.99838 to 0.99919, saving model to /content/drive/MyDrive/newmodel_07-1.00.h5
219/219 [==============================] - 170s 776ms/step - loss: 0.1610 - accuracy: 0.9964 - val_loss: 0.1591 - val_accuracy:
Epoch 8/15
```

```
219/219 [==============================] - ETA: 0s - loss: 0.1522 - accuracy: 0.9976
Epoch 8: val_accuracy did not improve from 0.99919
219/219 [==============================] - 168s 765ms/step - loss: 0.1522 - accuracy: 0.9976 - val_loss: 0.1449 - val_accuracy:
Epoch 9/15
219/219 [==============================] - ETA: 0s - loss: 0.1524 - accuracy: 0.9979
Epoch 9: val_accuracy did not improve from 0.99919
219/219 [==============================] - 166s 757ms/step - loss: 0.1524 - accuracy: 0.9979 - val_loss: 0.1544 - val_accuracy:
Epoch 10/15
219/219 [==============================] - ETA: 0s - loss: 0.1898 - accuracy: 0.9953
Epoch 10: val_accuracy did not improve from 0.99919
219/219 [==============================] - 165s 752ms/step - loss: 0.1898 - accuracy: 0.9953 - val_loss: 0.1482 - val_accuracy:
Epoch 11/15
219/219 [==============================] - ETA: 0s - loss: 0.1529 - accuracy: 0.9960
Epoch 11: val_accuracy did not improve from 0.99919
219/219 [==============================] - 167s 761ms/step - loss: 0.1529 - accuracy: 0.9960 - val_loss: 0.1455 - val_accuracy:
Epoch 12/15
219/219 [==============================] - ETA: 0s - loss: 0.1409 - accuracy: 0.9974
Epoch 12: val_accuracy did not improve from 0.99919
219/219 [==============================] - 166s 756ms/step - loss: 0.1409 - accuracy: 0.9974 - val_loss: 0.1383 - val_accuracy:
Epoch 13/15
219/219 [==============================] - ETA: 0s - loss: 0.1427 - accuracy: 0.9967
Epoch 13: val_accuracy did not improve from 0.99919
219/219 [==============================] - 162s 740ms/step - loss: 0.1427 - accuracy: 0.9967 - val_loss: 0.1389 - val_accuracy:
Epoch 14/15
219/219 [==============================] - ETA: 0s - loss: 0.1342 - accuracy: 0.9990
Epoch 14: val_accuracy did not improve from 0.99919
219/219 [==============================] - 163s 742ms/step - loss: 0.1342 - accuracy: 0.9990 - val_loss: 0.1356 - val_accuracy:
Epoch 15/15
```

```
acc = fit_history.history['accuracy']
val_acc = fit_history.history['val_accuracy']
loss = fit_history.history['loss']
val_loss = fit_history.history['val_loss']

epochs = range(len(acc))

plt.plot(epochs, acc, 'r', label='Training accuracy')
plt.plot(epochs, val_acc, 'b', label='Validation accuracy')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'r', label='Training Loss')
plt.plot(epochs, val_loss, 'b', label='Validation Loss')
plt.title('Training and validation loss')

plt.legend()

plt.show()
```




```
scores = model.evaluate(validation_generator)
```

```
39/39 [==============================] - 10s 261ms/step - loss: 0.1371 - accuracy: 0.9976
```

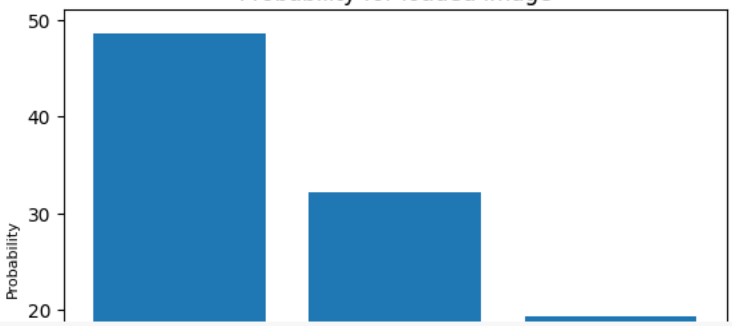
```
# Check our folder and import the model with best validation accuracy
loaded_best_model = keras.models.load_model("/content/drive/MyDrive/model_04-1.00.h5")

# Custom function to load and predict label for the image
def predict(img_rel_path):
    # Import Image from the path with size of (300, 300)
    img = image.load_img(img_rel_path, target_size=(300, 300))

    # Convert Image to a numpy array
    img = image.img_to_array(img, dtype=np.uint8)

    # Scaling the Image Array values between 0 and 1
    img = np.array(img)/255.0

    # Plotting the Loaded Image
    plt.title("Loaded Image")
    plt.axis('off')
    plt.imshow(img.squeeze())
    plt.show()

    # Get the Predicted Label for the loaded Image
    p = loaded_best_model.predict(img[np.newaxis, ...])

    # Label array
    labels = {0: 'cans', 1: 'glass_bottles',2:'plastic_bottles'}

    print("\n\nMaximum Probability: ", np.max(p[0], axis=-1))
    predicted_class = labels[np.argmax(p[0], axis=-1)]
    print("Classified:", predicted_class, "\n\n")

    classes=[]
    prob=[]
    print("\n-------------------Individual Probability--------------------------------\n")

    for i,j in enumerate (p[0],0):
        print(labels[i].upper(),':',round(j*100,2),'%')
        classes.append(labels[i])
        prob.append(round(j*100,2))

    def plot_bar_x():
        # this is for plotting purpose
        index = np.arange(len(classes))
        plt.bar(index, prob)
        plt.xlabel('Labels', fontsize=8)
        plt.ylabel('Probability', fontsize=8)
        plt.xticks(index, classes, fontsize=8, rotation=20)
        plt.title('Probability for loaded image')
        plt.show()
    plot_bar_x()
```

```
predict("/content/drive/MyDrive/bottle/cans/cans/agfie.jpg")
```

Loaded Image

```
1/1 [==============================] - 3s 3s/step


Maximum Probability:  0.48555177
Classified: cans



-------------------Individual Probability--------------------------------

CANS : 48.56 %
GLASS_BOTTLES : 32.11 %
PLASTIC_BOTTLES : 19.34 %
```

Probability for loaded image




```
predict("/content/drive/MyDrive/bottle/glass_bottles/glass_bottles/ahnxy.jpg")
```

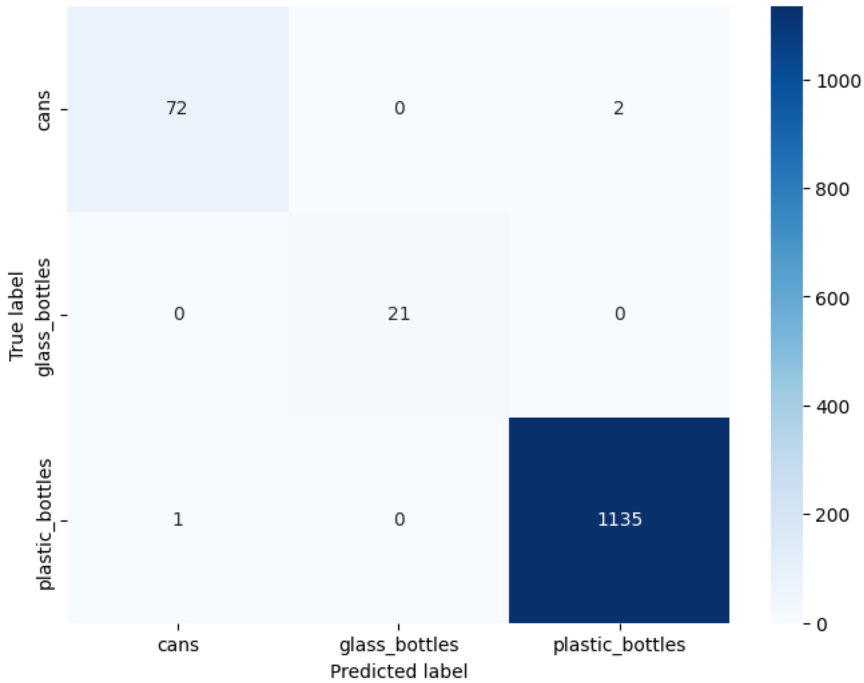
Loaded Image

```
1/1 [==============================] - 0s 32ms/step


Maximum Probability:  0.9811832
Classified: glass_bottles
```

```
predict("/content/drive/MyDrive/bottle/plastic_bottles/plastic_bottles/abwiq.jpg")
```


```
classes = train_generator.class_indices.keys()

from sklearn.metrics import confusion_matrix

y_pred = np.argmax(model.predict(validation_generator), axis=1)
cm = confusion_matrix(validation_generator.classes, y_pred)

# Heatmap
plt.figure(figsize=(8,6))
sns.heatmap(cm, annot=True, fmt='d', cbar=True, cmap='Blues',xticklabels=classes, yticklabels=classes)
plt.xlabel('Predicted label')
plt.ylabel('True label')
plt.title('Confusion matrix')
plt.show()
```

```
39/39 [==============================] - 12s 260ms/step
```




```
from sklearn.metrics import classification_report
target_names = ['cans','glass_bottles','plastic_bottles']
print(classification_report(validation_generator.classes, y_pred, target_names=target_names))
```

```
                 precision    recall  f1-score   support

           cans       0.99      0.97      0.98        74
  glass_bottles       1.00      1.00      1.00        21
plastic_bottles       1.00      1.00      1.00      1136

       accuracy                           1.00      1231
      macro avg       0.99      0.99      0.99      1231
   weighted avg       1.00      1.00      1.00      1231
```

```
import numpy as np
import matplotlib.pyplot as plt

# set width of bar
barWidth = 0.10
fig = plt.subplots(figsize =(15, 10))

# set height of bar
a= [72.4]
b= [78.2]
c= [79.6]
```