#Automatic Port Operations

Author : Sundar Krishnamachari

Objectives

- Identify different classes of boats using a Covolutional Neural Network model
- Identify different classes of boats using the MobileNet_V2 model using transfer learning
- Compare the performanaces of both and draw inferences

Tasks performed on each model

1. Evaluate Model's performance by calculating accuracy and loss metrics
2. Make predictions on the test set
3. Print confusion matrix and display
4. Print classificaton report

##Common Functions

```
import os
import pathlib
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import numpy as np
import seaborn as sns
import pandas as pd

from sklearn.metrics import confusion_matrix,
ConfusionMatrixDisplay,classification_report

import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from  tensorflow.keras.models import Sequential

import warnings
warnings.filterwarnings('ignore')

train_accuracy = {}
test_accuracy = {}
train_loss = {}
test_loss = {}


def print_scores(model, val_ds, train_ds,name='Default'):
    """
    :param model:
    :param val_ds:
    :param train_ds:
```

```
    :return:
    """
    print(f'Displaying accuracy and loss for {name}')

    score_test = model.evaluate(val_ds, verbose=0)
    score_train = model.evaluate(train_ds, verbose=0)

    print('Test loss:%.4f' % score_test[0])
    print('Test accuracy:%.4f' % score_test[1])

    print('Train loss:%.4f' % score_train[0])
    print('train accuracy:%.4f' % score_train[1])

    if not name == 'Default':
      test_loss[name] = score_test[0]
      test_accuracy[name] = score_test[1]
      train_loss[name] = score_train[0]
      train_accuracy[name] = score_train[1]



def _plot(history, param_1,param_2,ax):

    """
    :param history:
    :param param_1:
    :param param_2:
    :return:
    """

    ax.plot(history.history[param_1])
    ax.plot(history.history[param_2])
    ax.set_title(f'model {param_1}')
    ax.set_ylabel(param_1)
    ax.set_xlabel('epoch')
    ax.legend(['train', 'test'], loc='best')


def plot_accuracy_loss_graphs(history, title=None, show=True):
    """
    :param history:
    :param title:
    :param show:
    :return:
    """
    print()
    fig = plt.figure(figsize=(10, 10))

    # accuracy chart
    ax = plt.subplot(1, 2, 1)
```

```
    _plot(history,'accuracy','val_accuracy',ax)
    # loss chart
    ax = plt.subplot(1, 2, 2)
    _plot(history,'loss','val_loss',ax)
    if show:
        plt.tight_layout()
        plt.show()
    else:
        plt.savefig(f'{title}.png')
        plt.close()


def get_predicted_labels(model,class_names):
    """
    :param model:
    :param class_names:
    :return: dict of image paths and image labels
    """
    images = os.listdir(test_dir)
    image_label_map = {}


    for i, image in enumerate(images):
        path = test_dir + '/' + image
        img = tf.keras.utils.load_img(path, target_size=(height,
width))
        img_array = tf.keras.utils.img_to_array(img)
        img_array = tf.expand_dims(img_array, 0)  # Create a batch
        predictions = model.predict(img_array, verbose=False)
        score = tf.nn.softmax(predictions[0])
        max_score = np.max(score)
        image_label_map[path] = class_names[np.argmax(score)]
    return image_label_map


def display_test_images(image_map, model_name):
    """

    :param image_map:
    :param model_name:
    :return:
    """
    print(f'Predictions for {model_name}\n')
    plt.figure(figsize=(25, 25))
    for i, image_path in enumerate(image_map.keys()):
        ax = plt.subplot(20, 15, i + 1)
        img = mpimg.imread(image_path)
        ax.imshow(img)
        ax.axis('off')
        label = image_map.get(image_path)
```

```
        # ax.text(0.5, .8, label, fontsize=10, fontweight=''
color='green', ha='center', va='center',transform=ax.transAxes)
        ax.set_title(label, fontsize=10)
    plt.show()


def display_cm(true_labels, predicted_labels):
    """
    :param true_labels:
    :param predicted_labels:
    :return:
    """
    cm = confusion_matrix(true_labels, predicted_labels)
    print('Confusion Matrix')
    print(cm)
    print()

    print('Confusion Matrix Displayed\n')
    fig = plt.figure(figsize=(10, 10))
    disp = ConfusionMatrixDisplay(confusion_matrix=cm,
display_labels=class_names)
    disp.plot()
    plt.title('Confusion Matrix')
    plt.xticks(rotation=90)
    plt.show()


def get_metrics(model, validation_ds, val_split=0.2):
    num_images = int(total_train_images * val_split)
    num_batches = int(num_images / 32)
    residuals = int(num_images % 32)
    true_labels = []
    predicted_labels = []
    for images, labels in validation_ds.take(num_batches):
        true_labels.extend(labels)
        for i in range(batch_size):  # prediction for each image in
the batch
            img_array = tf.keras.utils.img_to_array(images[i])
            img_array = tf.expand_dims(img_array, 0)  # Create a batch
            predictions = model.predict(img_array, verbose=False)
            score = tf.nn.softmax(predictions[0])
            predicted_labels.append(np.argmax(score))
    return true_labels, predicted_labels


def create_transfer_model(transfer_model, dropout=0.1,
trainable=False, data_augmentation=None):
    transfer_model.trainable = trainable
    model_t = Sequential()
    if not data_augmentation is None:
```

```
        for aug in data_augmentation:
            print(f'added {aug} to model')
            model_t.add(aug)
    if data_augmentation is None:
        # data augmentation will add an input shape in the first layer
        model_t.add(layers.Rescaling(1. / 255, input_shape=(height,
width, 3)))
    else:
        model_t.add(layers.Rescaling(1. / 255))
    model_t.add(transfer_model)
    model_t.add(layers.GlobalAveragePooling2D())
    model_t.add(layers.Dropout(dropout))


    # FCN
    model_t.add(layers.Flatten())
    model_t.add(layers.Dense(256, activation='relu'))
    model_t.add(layers.BatchNormalization())
    model_t.add(layers.Dropout(dropout))

    model_t.add(layers.Dense(128, activation='relu'))
    model_t.add(layers.BatchNormalization())
    model_t.add(layers.Dropout(dropout))

    model_t.add(layers.Dense(len(class_names), activation='softmax'))
    model_t.summary()
    return model_t
```

##CNN Model

##Load Datasets

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

```
Mounted at /content/drive
```

##Extract data

```
# from zipfile import ZipFile


# zip_ref = ZipFile('/content/drive/MyDrive/DL/BOATS.zip','r')
# zip_ref.extractall('/content/drive/MyDrive/DL')
# zip_ref.close()
```

##Define constants , test and train directories

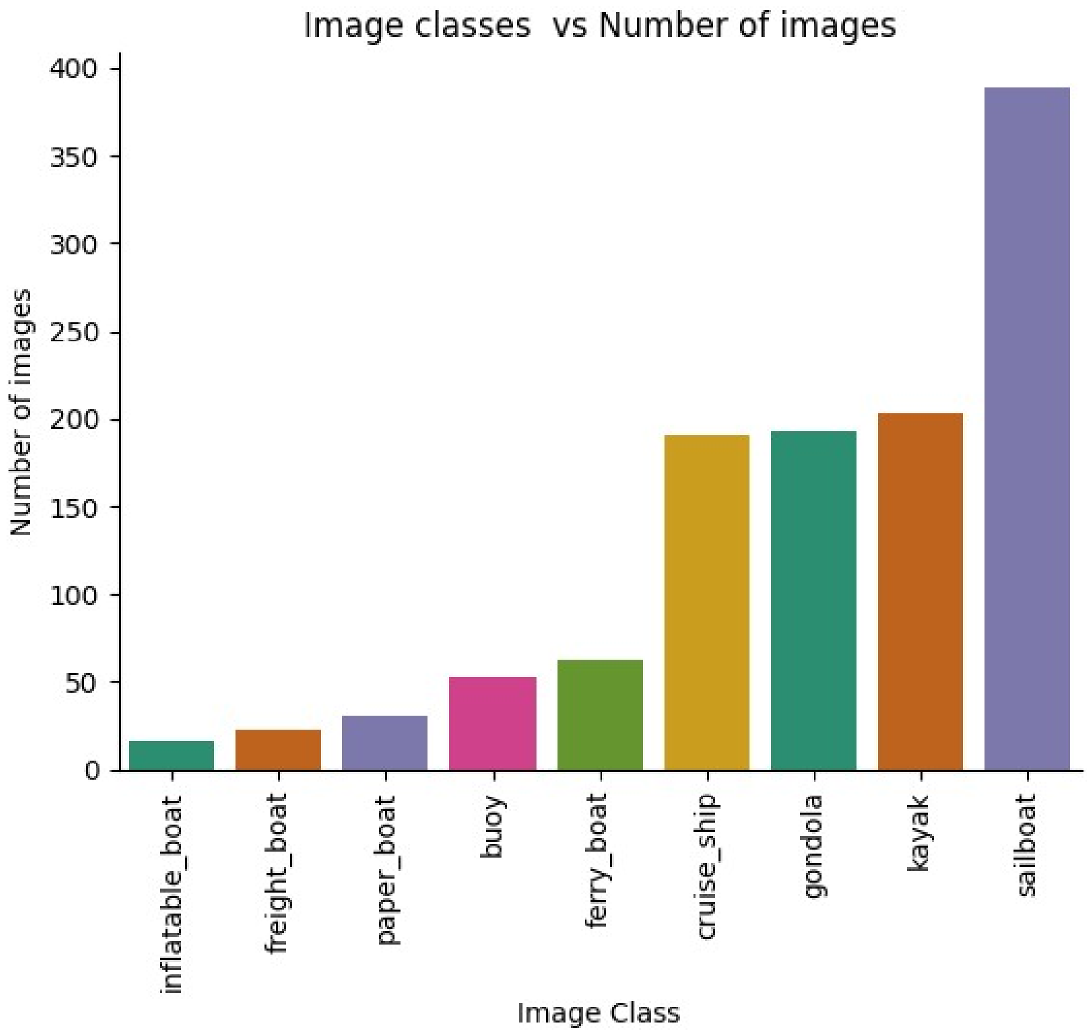
```
# lets define some constants
train_dir_kaggle = '/kaggle/input/boats-ds/TRAIN_BOATS'
test_dir_kaggle = '/kaggle/input/boats-ds/TEST_BOATS'


train_dir_colab = '/content/drive/MyDrive/DL/TRAIN_BOATS/'
test_dir_colab = '/content/drive/MyDrive/DL/TEST_BOATS/'

train_dir = train_dir_colab
test_dir = test_dir_colab

if os.path.exists(train_dir_kaggle):
  train_dir = train_dir_kaggle
  test_dir = test_dir_kaggle
  print('Using Kaggle data')
  print(f'train_dir = {train_dir}')
  print(f'test_dir = {test_dir}')
else:
  print('Using Colab data')
  print(f'train_dir = {train_dir}')
  print(f'test_dir = {test_dir}')


height = 224
width = 224
batch_size = 32
total_train_images = 0

total_test_images = len(os.listdir(test_dir))


for sub_dir in os.listdir(train_dir):
  total_train_images += len(os.listdir(train_dir+'/'+sub_dir))

print('total train images = ', total_train_images)
print('total test images =  ', total_test_images)
```

```
Using Colab data
train_dir = /content/drive/MyDrive/DL/TRAIN_BOATS/
test_dir = /content/drive/MyDrive/DL/TEST_BOATS/
total train images =  1162
total test images =   300
```

##Estimate number of images per class and plot

```
data_dir = pathlib.Path(train_dir)

def get_subdirectories(directory):
    with os.scandir(directory) as entries:
```

```
        return [entry.name for entry in entries if entry.is_dir()]
sub_dirs = get_subdirectories(data_dir)
images = {}
image_class = []
num_images = []
class_dict = {}

for d in sub_dirs:
    image_class.append(d)
    num_images.append(len(os.listdir(train_dir+'/'+d)))

df =
pd.DataFrame({'Image_Class':image_class,'Num_Images':num_images}).sort
_values(by='Num_Images',ascending=True).reset_index()

indices = sorted(df['index'].values.tolist())
class_names = df['Image_Class'].values
for i in indices:
  class_dict[i] = class_names[i]
sns.barplot(data=df,x='Image_Class',y='Num_Images',palette=sns.mpl_pal
ette('Dark2'))
plt.xlabel('Image Class')
plt.ylabel('Number of images')
plt.title(f'Image classes  vs Number of images' )
plt.xticks(rotation = 90)
plt.gca().spines[['top', 'right',]].set_visible(False)
plt.show()
df
```




```
{"summary":"{\n  \"name\": \"df\",\n  \"rows\": 9,\n  \"fields\": [\n    {\n      \"column\": \"index\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 2,\n        \"min\": 0,\n        \"max\": 8,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          6,\n          3,\n          1\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Image_Class\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 9,\n        \"samples\": [\n          \"kayak\",\n          \"freight_boat\",\n          \"cruise_ship\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Num_Images\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 124,\n        \"min\": 16,\n        \"max\": 389,\n        \"num_unique_values\": 9,\n        \"samples\": [\n          203,\n          23,\n          191\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"df"}
```

There is a predominence of the sailboat class in the training data. The bottom three image classes have less then 50 images per class. Ideally we can drop them. Having them in the training/testing datasets can skew the predictions. But, I am keeping them nonetheless.

##Define datasets for training and validatating the model

```
# Not using ImageDataGenerator as it is deprecated and it doesnt allow
for
# caching and prefetching. With image data, caching and prefetcing can
determine
# if training will be possible with limited gpu resources

# https://www.tensorflow.org/guide/data

import tensorflow as tf

def get_ds(data_dir,validation_split=0.2,subset='training',seed =
43,image_size=(255,255),batch_size=32):
    train_ds = tf.keras.utils.image_dataset_from_directory(
    data_dir,
    validation_split=validation_split,
    subset=subset,
    seed=seed,
    image_size=image_size,
    batch_size=batch_size)
    train_ds.class_names
    return train_ds

train_ds =
get_ds(data_dir,image_size=(height,width),batch_size=batch_size)
val_ds = get_ds(data_dir,validation_split=0.2,subset='validation',seed
= 43,image_size=(height,width),batch_size=batch_size)
class_names = train_ds.class_names
print(class_names)
```

```
Found 1162 files belonging to 9 classes.
Using 930 files for training.
Found 1162 files belonging to 9 classes.
Using 232 files for validation.
['buoy', 'cruise_ship', 'ferry_boat', 'freight_boat', 'gondola',
'inflatable_boat', 'kayak', 'paper_boat', 'sailboat']
```

##Plot some images from the training dataset

```
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 10))

for images, labels in train_ds.take(1):
  for i in range(9):
```

```
    ax = plt.subplot(3, 3, i + 1)
    plt.imshow(images[i].numpy().astype("uint8"))
    plt.title(class_names[labels[i]])
    plt.axis("off")
```

##Set up the datasets for caching and prefetching

```
# Using `cache`, `prfetch` and `AUTOTUNE` for efficient handling of
input data.
# https://www.tensorflow.org/guide/data_performance
AUTOTUNE = tf.data.AUTOTUNE
```

```
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

##CNN model exactly as specfied in the problem 2 Conv2d layers with Max 2d pooling

1 2D Global Average Pooling

2 FFN with 2 dense layers of 128 neurons each

1 outpt layer with 9 neurons and softmax activation

```
model = Sequential([

  # Image Scaling layer
  layers.Rescaling(1./255, input_shape=(height, width, 3)),

  # First convolution layer.convolution with 32 filters with a 3x3
kernel matrix,no padding
  layers.Conv2D(32, (3,3),  activation='relu'),
  layers.MaxPooling2D(),
  #layers.Dropout(0.1),

  # Second Convolution layer
  layers.Conv2D(32, (3,3),  activation='relu'),
  layers.MaxPooling2D(),

  #Global Average Pooling
  layers.GlobalAveragePooling2D(),

  # FFN
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(128, activation='relu'),
  layers.Dense(len(class_names),activation='softmax')

])
```

##Compile the model

```
model.compile(optimizer='adam',

#https://www.tensorflow.org/api_docs/python/tf/keras/losses/Categorica
lCrossentropy
              #It states:
```

```
              # "We expect labels to be provided in a one_hot
representation.
              #  If you want to provide labels as integers, please use
SparseCategoricalCrossentropy loss."
              loss = 'sparse_categorical_crossentropy',
              # keras has removed support for batch level precision
and recall metrics.Problem statement is obsolate
              metrics=['accuracy']
             )
model.summary()
```

```
Model: "sequential"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ rescaling (Rescaling)                │ (None, 224, 224, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d (Conv2D)                      │ (None, 222, 222, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d (MaxPooling2D)         │ (None, 111, 111, 32)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_1 (Conv2D)                    │ (None, 109, 109, 32)        │
9,248 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_1 (MaxPooling2D)       │ (None, 54, 54, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ global_average_pooling2d             │ (None, 32)                  │
0 │
│ (GlobalAveragePooling2D)             │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ flatten (Flatten)                    │ (None, 32)                  │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense (Dense)                        │ (None, 128)                 │
```

```
4,224 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_1 (Dense)                      │ (None, 128)                 │
16,512 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_2 (Dense)                      │ (None, 9)                   │
1,161 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 32,041 (125.16 KB)

 Trainable params: 32,041 (125.16 KB)

 Non-trainable params: 0 (0.00 B)
```

Train the model with Early Stopping and Checkpoint

```
from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint

#Optimizing for maximum accuracy instead of minimum loss.( min loss
does not always mean max accuracy)
early_stop=EarlyStopping(monitor='val_accuracy',mode='max',verbose=1,p
atience=10,min_delta=0.01)
checkpoint=ModelCheckpoint('best_model.keras',monitor='val_accuracy',m
ode='max',verbose=1,save_best_only=True)

epochs=100
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  batch_size = batch_size,
  # not specified. But i thought this was a good idea
  callbacks = [early_stop,checkpoint]
)

Epoch 1/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 79ms/step - accuracy: 0.2852 - loss:
2.0488
Epoch 1: val_accuracy improved from -inf to 0.30172, saving model to
best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 185s 1s/step - accuracy: 0.2881 - loss:
2.0421 - val_accuracy: 0.3017 - val_loss: 1.8125
Epoch 2/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 27ms/step - accuracy: 0.3327 - loss:
1.8598
Epoch 2: val_accuracy did not improve from 0.30172
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.3330 - loss:
1.8580 - val_accuracy: 0.3017 - val_loss: 1.7931
Epoch 3/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.3450 - loss:
1.7445
Epoch 3: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.3449 - loss:
1.7460 - val_accuracy: 0.3017 - val_loss: 1.7872
Epoch 4/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.3387 - loss:
1.7577
Epoch 4: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.3390 - loss:
1.7580 - val_accuracy: 0.3017 - val_loss: 1.7648
Epoch 5/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.3479 - loss:
1.8035
Epoch 5: val_accuracy improved from 0.30172 to 0.31034, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.3485 - loss:
1.8015 - val_accuracy: 0.3103 - val_loss: 1.7453
Epoch 6/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.3658 - loss:
1.6941
Epoch 6: val_accuracy improved from 0.31034 to 0.31897, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.3661 - loss:
1.6944 - val_accuracy: 0.3190 - val_loss: 1.7383
Epoch 7/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4214 - loss:
1.6582
Epoch 7: val_accuracy did not improve from 0.31897
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4206 - loss:
1.6593 - val_accuracy: 0.3060 - val_loss: 1.7430
Epoch 8/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.3546 - loss:
1.6897
Epoch 8: val_accuracy improved from 0.31897 to 0.32759, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.3574 - loss:
1.6887 - val_accuracy: 0.3276 - val_loss: 1.7422
Epoch 9/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4039 - loss:
1.6579
Epoch 9: val_accuracy did not improve from 0.32759
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4041 - loss:
1.6581 - val_accuracy: 0.3103 - val_loss: 1.7343
Epoch 10/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.3987 - loss:
```

```
1.6922
Epoch 10: val_accuracy did not improve from 0.32759
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.3988 - loss:
1.6909 - val_accuracy: 0.3276 - val_loss: 1.7060
Epoch 11/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4005 - loss:
1.6570
Epoch 11: val_accuracy did not improve from 0.32759
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4006 - loss:
1.6571 - val_accuracy: 0.3190 - val_loss: 1.8027
Epoch 12/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4045 - loss:
1.6640
Epoch 12: val_accuracy did not improve from 0.32759
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4052 - loss:
1.6630 - val_accuracy: 0.3276 - val_loss: 1.7219
Epoch 13/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.3792 - loss:
1.6355
Epoch 13: val_accuracy did not improve from 0.32759
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.3802 - loss:
1.6355 - val_accuracy: 0.3190 - val_loss: 1.7777
Epoch 14/100
28/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4101 - loss:
1.6534
Epoch 14: val_accuracy improved from 0.32759 to 0.33190, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.4108 - loss:
1.6535 - val_accuracy: 0.3319 - val_loss: 1.7044
Epoch 15/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.4504 - loss:
1.5937
Epoch 15: val_accuracy improved from 0.33190 to 0.34052, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.4486 - loss:
1.5961 - val_accuracy: 0.3405 - val_loss: 1.6975
Epoch 16/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.4283 - loss:
1.6227
Epoch 16: val_accuracy did not improve from 0.34052
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.4281 - loss:
1.6224 - val_accuracy: 0.3319 - val_loss: 1.6979
Epoch 17/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4119 - loss:
1.6808
Epoch 17: val_accuracy did not improve from 0.34052
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.4122 - loss:
1.6790 - val_accuracy: 0.3233 - val_loss: 1.7019
Epoch 18/100
```

```
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4370 - loss:
1.6216
Epoch 18: val_accuracy did not improve from 0.34052
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4366 - loss:
1.6203 - val_accuracy: 0.3319 - val_loss: 1.7042
Epoch 19/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4183 - loss:
1.6161
Epoch 19: val_accuracy improved from 0.34052 to 0.34914, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.4184 - loss:
1.6155 - val_accuracy: 0.3491 - val_loss: 1.6773
Epoch 20/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4340 - loss:
1.5819
Epoch 20: val_accuracy did not improve from 0.34914
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4336 - loss:
1.5824 - val_accuracy: 0.3319 - val_loss: 1.6749
Epoch 21/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4088 - loss:
1.6157
Epoch 21: val_accuracy improved from 0.34914 to 0.36207, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.4096 - loss:
1.6151 - val_accuracy: 0.3621 - val_loss: 1.6463
Epoch 22/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4592 - loss:
1.5375
Epoch 22: val_accuracy did not improve from 0.36207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4584 - loss:
1.5387 - val_accuracy: 0.3405 - val_loss: 1.6630
Epoch 23/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4134 - loss:
1.5790
Epoch 23: val_accuracy did not improve from 0.36207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4141 - loss:
1.5786 - val_accuracy: 0.3578 - val_loss: 1.6266
Epoch 24/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4524 - loss:
1.5239
Epoch 24: val_accuracy did not improve from 0.36207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4519 - loss:
1.5251 - val_accuracy: 0.3578 - val_loss: 1.6500
Epoch 25/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4447 - loss:
1.5227
Epoch 25: val_accuracy did not improve from 0.36207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4447 - loss:
1.5235 - val_accuracy: 0.3362 - val_loss: 1.6190
```

```
Epoch 26/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4715 - loss:
1.4954
Epoch 26: val_accuracy improved from 0.36207 to 0.37500, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.4711 - loss:
1.4968 - val_accuracy: 0.3750 - val_loss: 1.5957
Epoch 27/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4502 - loss:
1.5242
Epoch 27: val_accuracy did not improve from 0.37500
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4502 - loss:
1.5248 - val_accuracy: 0.3578 - val_loss: 1.6842
Epoch 28/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4747 - loss:
1.5090
Epoch 28: val_accuracy improved from 0.37500 to 0.38362, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.4740 - loss:
1.5099 - val_accuracy: 0.3836 - val_loss: 1.5858
Epoch 29/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4512 - loss:
1.4877
Epoch 29: val_accuracy did not improve from 0.38362
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4513 - loss:
1.4898 - val_accuracy: 0.3534 - val_loss: 1.5949
Epoch 30/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4364 - loss:
1.5884
Epoch 30: val_accuracy did not improve from 0.38362
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.4374 - loss:
1.5866 - val_accuracy: 0.3750 - val_loss: 1.5709
Epoch 31/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4769 - loss:
1.4901
Epoch 31: val_accuracy improved from 0.38362 to 0.41379, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.4765 - loss:
1.4904 - val_accuracy: 0.4138 - val_loss: 1.6129
Epoch 32/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4549 - loss:
1.5267
Epoch 32: val_accuracy did not improve from 0.41379
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4559 - loss:
1.5256 - val_accuracy: 0.3750 - val_loss: 1.5965
Epoch 33/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4969 - loss:
1.4395
Epoch 33: val_accuracy did not improve from 0.41379
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4967 - loss:
1.4407 - val_accuracy: 0.4138 - val_loss: 1.5971
Epoch 34/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5139 - loss:
1.4200
Epoch 34: val_accuracy did not improve from 0.41379
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.5133 - loss:
1.4215 - val_accuracy: 0.3707 - val_loss: 1.5685
Epoch 35/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4459 - loss:
1.5455
Epoch 35: val_accuracy did not improve from 0.41379
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4466 - loss:
1.5435 - val_accuracy: 0.3966 - val_loss: 1.5519
Epoch 36/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4704 - loss:
1.4832
Epoch 36: val_accuracy improved from 0.41379 to 0.42241, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.4708 - loss:
1.4827 - val_accuracy: 0.4224 - val_loss: 1.5245
Epoch 37/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5102 - loss:
1.4282
Epoch 37: val_accuracy improved from 0.42241 to 0.45259, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.5096 - loss:
1.4295 - val_accuracy: 0.4526 - val_loss: 1.5232
Epoch 38/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4749 - loss:
1.4864
Epoch 38: val_accuracy did not improve from 0.45259
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4751 - loss:
1.4863 - val_accuracy: 0.4267 - val_loss: 1.5239
Epoch 39/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4659 - loss:
1.4905
Epoch 39: val_accuracy improved from 0.45259 to 0.45690, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.4669 - loss:
1.4891 - val_accuracy: 0.4569 - val_loss: 1.5199
Epoch 40/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5146 - loss:
1.4142
Epoch 40: val_accuracy did not improve from 0.45690
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.5141 - loss:
1.4149 - val_accuracy: 0.4095 - val_loss: 1.5277
Epoch 41/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5006 - loss:
```

```
1.4690
Epoch 41: val_accuracy did not improve from 0.45690
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.5006 - loss:
1.4683 - val_accuracy: 0.4138 - val_loss: 1.6197
Epoch 42/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.5218 - loss:
1.4158
Epoch 42: val_accuracy improved from 0.45690 to 0.46121, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.5211 - loss:
1.4166 - val_accuracy: 0.4612 - val_loss: 1.5268
Epoch 43/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.5019 - loss:
1.4511
Epoch 43: val_accuracy did not improve from 0.46121
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.5023 - loss:
1.4502 - val_accuracy: 0.4224 - val_loss: 1.5743
Epoch 44/100
28/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.4883 - loss:
1.4382
Epoch 44: val_accuracy did not improve from 0.46121
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.4898 - loss:
1.4359 - val_accuracy: 0.4440 - val_loss: 1.5133
Epoch 45/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5217 - loss:
1.4347
Epoch 45: val_accuracy did not improve from 0.46121
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.5215 - loss:
1.4346 - val_accuracy: 0.4052 - val_loss: 1.6003
Epoch 46/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5069 - loss:
1.4543
Epoch 46: val_accuracy improved from 0.46121 to 0.46983, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 29ms/step - accuracy: 0.5068 - loss:
1.4543 - val_accuracy: 0.4698 - val_loss: 1.5034
Epoch 47/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.5216 - loss:
1.4135
Epoch 47: val_accuracy improved from 0.46983 to 0.48276, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 30ms/step - accuracy: 0.5210 - loss:
1.4139 - val_accuracy: 0.4828 - val_loss: 1.4841
Epoch 48/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5428 - loss:
1.3631
Epoch 48: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.5413 - loss:
1.3651 - val_accuracy: 0.4784 - val_loss: 1.4935
```

```
Epoch 49/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5158 - loss:
1.3895
Epoch 49: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.5161 - loss:
1.3895 - val_accuracy: 0.4569 - val_loss: 1.5253
Epoch 50/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5049 - loss:
1.3991
Epoch 50: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.5050 - loss:
1.3996 - val_accuracy: 0.4784 - val_loss: 1.4883
Epoch 51/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5082 - loss:
1.4229
Epoch 51: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.5085 - loss:
1.4217 - val_accuracy: 0.4741 - val_loss: 1.4719
Epoch 52/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.5348 - loss:
1.3502
Epoch 52: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.5328 - loss:
1.3539 - val_accuracy: 0.4267 - val_loss: 1.5252
Epoch 53/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5221 - loss:
1.4030
Epoch 53: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.5218 - loss:
1.4028 - val_accuracy: 0.4483 - val_loss: 1.5108
Epoch 54/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5228 - loss:
1.4032
Epoch 54: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 31ms/step - accuracy: 0.5224 - loss:
1.4035 - val_accuracy: 0.4483 - val_loss: 1.5235
Epoch 55/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 26ms/step - accuracy: 0.5544 - loss:
1.3155
Epoch 55: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.5533 - loss:
1.3173 - val_accuracy: 0.4655 - val_loss: 1.4970
Epoch 56/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5232 - loss:
1.3861
Epoch 56: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 28ms/step - accuracy: 0.5229 - loss:
1.3856 - val_accuracy: 0.4655 - val_loss: 1.4840
Epoch 57/100
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 25ms/step - accuracy: 0.5530 - loss: 
1.3690
Epoch 57: val_accuracy did not improve from 0.48276
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 27ms/step - accuracy: 0.5529 - loss: 
1.3683 - val_accuracy: 0.4655 - val_loss: 1.4759
Epoch 57: early stopping
```

##Print the metrics

loss and accuracy graphs are plotted for all models in one graph later

```python
from tensorflow.keras.models import load_model
model = load_model('best_model.keras')
print_scores(model,val_ds,train_ds)
```

```
Displaying accuracy and loss for Default
Test loss:1.4841
Test accuracy:0.4828
Train loss:1.3976
train accuracy:0.5247
```

##Modified CNN Model

```python
model = Sequential([
  # Rescaling  images
  layers.Rescaling(1./255, input_shape=(height, width, 3)),

  # First convolution layer.convolution with 32 filters with a 3x3 
kernel matrix
  layers.Conv2D(32, (3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(), # Use max pooling
  layers.Dropout(0.1),

  # Second Convolution layer
  layers.Conv2D(32, (3,3), padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.1),

  # 3rd Convolution layer
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.1),

  # 4th Convolution layer
  layers.Conv2D(128, 3, padding='same', activation='relu'),
  layers.GlobalAveragePooling2D(),
  layers.Dropout(0.1),
```

```
    #FCN
    layers.Flatten(),
    layers.Dense(128, activation='relu'),
    layers.Dense(128, activation='relu'),
    layers.Dropout(0.2),
    layers.Dense(len(class_names),activation='softmax')

])
```

##Compile the model

```
model.compile(optimizer='adam',

#https://www.tensorflow.org/api_docs/python/tf/keras/losses/Categorica
lCrossentropy
              #It states:
              # "We expect labels to be provided in a one_hot
representation.
              #  If you want to provide labels as integers, please use
SparseCategoricalCrossentropy loss."
              loss = 'sparse_categorical_crossentropy',
              # keras has removed support for batch level precision
and recall metrics.Problem statement is obsolate
              metrics=['accuracy']
             )
model.summary()
```

```
Model: "sequential_1"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ rescaling_1 (Rescaling)              │ (None, 224, 224, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_2 (Conv2D)                    │ (None, 224, 224, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_2 (MaxPooling2D)       │ (None, 112, 112, 32)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout (Dropout)                    │ (None, 112, 112, 32)        │
0 │
```

```
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv2d_3 (Conv2D)                    │ (None, 112, 112, 32)        │
9,248 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 56, 56, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dropout_1 (Dropout)                  │ (None, 56, 56, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv2d_4 (Conv2D)                    │ (None, 56, 56, 64)          │
18,496 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ max_pooling2d_4 (MaxPooling2D)       │ (None, 28, 28, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dropout_2 (Dropout)                  │ (None, 28, 28, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv2d_5 (Conv2D)                    │ (None, 28, 28, 128)         │
73,856 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ global_average_pooling2d_1           │ (None, 128)                 │
0 │
│ (GlobalAveragePooling2D)             │                             │
│
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dropout_3 (Dropout)                  │ (None, 128)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ flatten_1 (Flatten)                  │ (None, 128)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dense_3 (Dense)                      │ (None, 128)                 │
16,512 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ dense_4 (Dense)                      │ (None, 128)                 │
```

```
16,512 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_4 (Dropout)                  │ (None, 128)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_5 (Dense)                      │ (None, 9)                   │
1,161 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 136,681 (533.91 KB)

 Trainable params: 136,681 (533.91 KB)

 Non-trainable params: 0 (0.00 B)
```

##Train the model

```
histories = []

from tensorflow.keras.callbacks import EarlyStopping,ModelCheckpoint

early_stop=EarlyStopping(monitor='val_accuracy',mode='max',verbose=1,p
atience=10,min_delta=0.01)
checkpoint=ModelCheckpoint('best_model.keras',monitor='val_accuracy',m
ode='max',verbose=1,save_best_only=True)

epochs=100
history_c = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  batch_size = batch_size,
  callbacks = [early_stop,checkpoint]
)

histories.append(history_c)

Epoch 1/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 178ms/step - accuracy: 0.3032 - loss:
2.0114
Epoch 1: val_accuracy improved from -inf to 0.30172, saving model to
best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 16s 242ms/step - accuracy: 0.3037 - loss:
2.0085 - val_accuracy: 0.3017 - val_loss: 1.8247
Epoch 2/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.3535 - loss:
1.8137
```

```
Epoch 2: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.3530 - loss:
1.8145 - val_accuracy: 0.3017 - val_loss: 1.8503
Epoch 3/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.3515 - loss:
1.8248
Epoch 3: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.3511 - loss:
1.8244 - val_accuracy: 0.3017 - val_loss: 1.7972
Epoch 4/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.3525 - loss:
1.7807
Epoch 4: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.3522 - loss:
1.7817 - val_accuracy: 0.3017 - val_loss: 1.8086
Epoch 5/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.3359 - loss:
1.8012
Epoch 5: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.3361 - loss:
1.8012 - val_accuracy: 0.3017 - val_loss: 1.8066
Epoch 6/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.3538 - loss:
1.7908
Epoch 6: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.3534 - loss:
1.7909 - val_accuracy: 0.3017 - val_loss: 1.7699
Epoch 7/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.3090 - loss:
1.8624
Epoch 7: val_accuracy did not improve from 0.30172
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.3101 - loss:
1.8594 - val_accuracy: 0.3017 - val_loss: 1.7858
Epoch 8/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.3516 - loss:
1.7776
Epoch 8: val_accuracy improved from 0.30172 to 0.31897, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.3514 - loss:
1.7770 - val_accuracy: 0.3190 - val_loss: 1.7395
Epoch 9/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.3797 - loss:
1.7487
Epoch 9: val_accuracy improved from 0.31897 to 0.32328, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.3798 - loss:
1.7481 - val_accuracy: 0.3233 - val_loss: 1.7551
Epoch 10/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4095 - loss:
```

```
1.7087
Epoch 10: val_accuracy improved from 0.32328 to 0.33621, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.4089 - loss:
1.7095 - val_accuracy: 0.3362 - val_loss: 1.7165
Epoch 11/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4004 - loss:
1.6759
Epoch 11: val_accuracy did not improve from 0.33621
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.3998 - loss:
1.6761 - val_accuracy: 0.3190 - val_loss: 1.6731
Epoch 12/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.3972 - loss:
1.6420
Epoch 12: val_accuracy did not improve from 0.33621
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.3971 - loss:
1.6446 - val_accuracy: 0.3276 - val_loss: 1.7359
Epoch 13/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4046 - loss:
1.6948
Epoch 13: val_accuracy improved from 0.33621 to 0.35345, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.4049 - loss:
1.6936 - val_accuracy: 0.3534 - val_loss: 1.6605
Epoch 14/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4402 - loss:
1.6305
Epoch 14: val_accuracy did not improve from 0.35345
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.4405 - loss:
1.6287 - val_accuracy: 0.3534 - val_loss: 1.6316
Epoch 15/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 35ms/step - accuracy: 0.4422 - loss:
1.6274
Epoch 15: val_accuracy improved from 0.35345 to 0.40948, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.4418 - loss:
1.6249 - val_accuracy: 0.4095 - val_loss: 1.5927
Epoch 16/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.4701 - loss:
1.5659
Epoch 16: val_accuracy did not improve from 0.40948
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.4700 - loss:
1.5657 - val_accuracy: 0.3922 - val_loss: 1.5756
Epoch 17/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.4523 - loss:
1.5171
Epoch 17: val_accuracy improved from 0.40948 to 0.43103, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.4523 - loss:
```

```
1.5182 - val_accuracy: 0.4310 - val_loss: 1.5797
Epoch 18/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4745 - loss:
1.4854
Epoch 18: val_accuracy did not improve from 0.43103
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.4746 - loss:
1.4863 - val_accuracy: 0.4267 - val_loss: 1.5424
Epoch 19/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5131 - loss:
1.4194
Epoch 19: val_accuracy improved from 0.43103 to 0.45259, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.5125 - loss:
1.4218 - val_accuracy: 0.4526 - val_loss: 1.5570
Epoch 20/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5114 - loss:
1.4332
Epoch 20: val_accuracy did not improve from 0.45259
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.5108 - loss:
1.4347 - val_accuracy: 0.4224 - val_loss: 1.5428
Epoch 21/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4983 - loss:
1.4524
Epoch 21: val_accuracy did not improve from 0.45259
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.4983 - loss:
1.4529 - val_accuracy: 0.3922 - val_loss: 1.5954
Epoch 22/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4794 - loss:
1.4627
Epoch 22: val_accuracy did not improve from 0.45259
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.4798 - loss:
1.4627 - val_accuracy: 0.4267 - val_loss: 1.5085
Epoch 23/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5051 - loss:
1.4491
Epoch 23: val_accuracy improved from 0.45259 to 0.47845, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 40ms/step - accuracy: 0.5049 - loss:
1.4500 - val_accuracy: 0.4784 - val_loss: 1.5350
Epoch 24/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.4906 - loss:
1.4845
Epoch 24: val_accuracy did not improve from 0.47845
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.4908 - loss:
1.4839 - val_accuracy: 0.4353 - val_loss: 1.4958
Epoch 25/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5208 - loss:
1.4296
Epoch 25: val_accuracy did not improve from 0.47845
```

```
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.5212 - loss:
1.4287 - val_accuracy: 0.4698 - val_loss: 1.5100
Epoch 26/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.5185 - loss:
1.4756
Epoch 26: val_accuracy improved from 0.47845 to 0.50431, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - accuracy: 0.5187 - loss:
1.4736 - val_accuracy: 0.5043 - val_loss: 1.4486
Epoch 27/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.5326 - loss:
1.3288
Epoch 27: val_accuracy did not improve from 0.50431
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.5326 - loss:
1.3305 - val_accuracy: 0.4741 - val_loss: 1.4511
Epoch 28/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5674 - loss:
1.3374
Epoch 28: val_accuracy did not improve from 0.50431
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.5662 - loss:
1.3401 - val_accuracy: 0.4612 - val_loss: 1.4757
Epoch 29/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5563 - loss:
1.3446
Epoch 29: val_accuracy improved from 0.50431 to 0.55603, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.5562 - loss:
1.3455 - val_accuracy: 0.5560 - val_loss: 1.4172
Epoch 30/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5409 - loss:
1.3657
Epoch 30: val_accuracy improved from 0.55603 to 0.58190, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.5409 - loss:
1.3661 - val_accuracy: 0.5819 - val_loss: 1.4059
Epoch 31/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.5608 - loss:
1.3830
Epoch 31: val_accuracy did not improve from 0.58190
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.5608 - loss:
1.3820 - val_accuracy: 0.5172 - val_loss: 1.4210
Epoch 32/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5864 - loss:
1.2669
Epoch 32: val_accuracy did not improve from 0.58190
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.5866 - loss:
1.2666 - val_accuracy: 0.5517 - val_loss: 1.4315
Epoch 33/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5997 - loss:
```

```
1.2963
Epoch 33: val_accuracy did not improve from 0.58190
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.5997 - loss:
1.2944 - val_accuracy: 0.5603 - val_loss: 1.3752
Epoch 34/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6008 - loss:
1.2259
Epoch 34: val_accuracy did not improve from 0.58190
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.6007 - loss:
1.2260 - val_accuracy: 0.5086 - val_loss: 1.4549
Epoch 35/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5908 - loss:
1.2610
Epoch 35: val_accuracy did not improve from 0.58190
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.5900 - loss:
1.2624 - val_accuracy: 0.5776 - val_loss: 1.3698
Epoch 36/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6070 - loss:
1.2713
Epoch 36: val_accuracy improved from 0.58190 to 0.61207, saving model
to best_model.keras
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.6068 - loss:
1.2711 - val_accuracy: 0.6121 - val_loss: 1.2901
Epoch 37/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.5647 - loss:
1.2750
Epoch 37: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.5655 - loss:
1.2737 - val_accuracy: 0.5819 - val_loss: 1.3531
Epoch 38/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6387 - loss:
1.2129
Epoch 38: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - accuracy: 0.6383 - loss:
1.2117 - val_accuracy: 0.5991 - val_loss: 1.3431
Epoch 39/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6625 - loss:
1.1206
Epoch 39: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.6619 - loss:
1.1210 - val_accuracy: 0.5560 - val_loss: 1.4739
Epoch 40/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6515 - loss:
1.1457
Epoch 40: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.6503 - loss:
1.1476 - val_accuracy: 0.5819 - val_loss: 1.2794
Epoch 41/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6806 - loss:
```

```
1.0863
Epoch 41: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.6798 - loss:
1.0882 - val_accuracy: 0.5991 - val_loss: 1.3043
Epoch 42/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.6728 - loss:
1.0701
Epoch 42: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.6722 - loss:
1.0718 - val_accuracy: 0.6034 - val_loss: 1.3370
Epoch 43/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 32ms/step - accuracy: 0.6360 - loss:
1.1083
Epoch 43: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 37ms/step - accuracy: 0.6363 - loss:
1.1085 - val_accuracy: 0.6034 - val_loss: 1.2843
Epoch 44/100
30/30 ━━━━━━━━━━━━━━━━━━━━ 0s 33ms/step - accuracy: 0.6622 - loss:
1.0919
Epoch 44: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - accuracy: 0.6617 - loss:
1.0917 - val_accuracy: 0.5690 - val_loss: 1.3548
Epoch 45/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.6657 - loss:
1.0585
Epoch 45: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 35ms/step - accuracy: 0.6655 - loss:
1.0596 - val_accuracy: 0.5905 - val_loss: 1.2406
Epoch 46/100
29/30 ━━━━━━━━━━━━━━━━━━━━ 0s 34ms/step - accuracy: 0.6441 - loss:
1.1389
Epoch 46: val_accuracy did not improve from 0.61207
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 0.6436 - loss:
1.1382 - val_accuracy: 0.6078 - val_loss: 1.2634
Epoch 46: early stopping
```

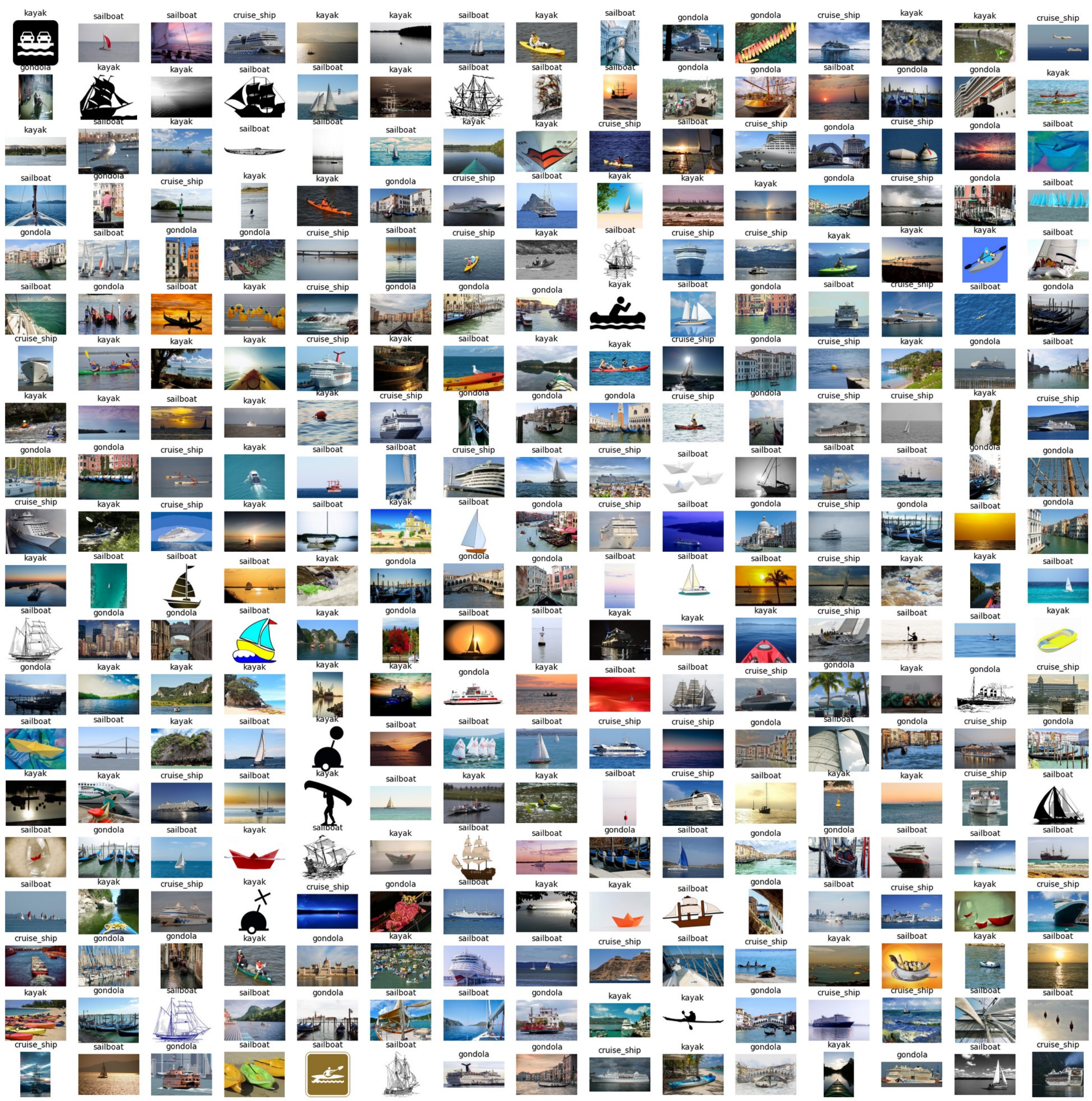
##Print Metrics

```
from tensorflow.keras.models import load_model
model = load_model('best_model.keras')
print_scores(model,val_ds,train_ds,'CNN Model')

Displaying accuracy and loss for CNN Model
Test loss:1.2901
Test accuracy:0.6121
Train loss:1.1262
train accuracy:0.6409
```

##Display the predictions

```
image_map = get_predicted_labels(model,class_names)
display_test_images(image_map,'CNN Model. Title is the predicted
label')
```

```
Predictions for CNN Model. Title is the predicted label
```



Notice how classes with very few images in training set tend to get misclassified. sailboat , which has the largest number of images is the best predicted class

##Confusion Matrix

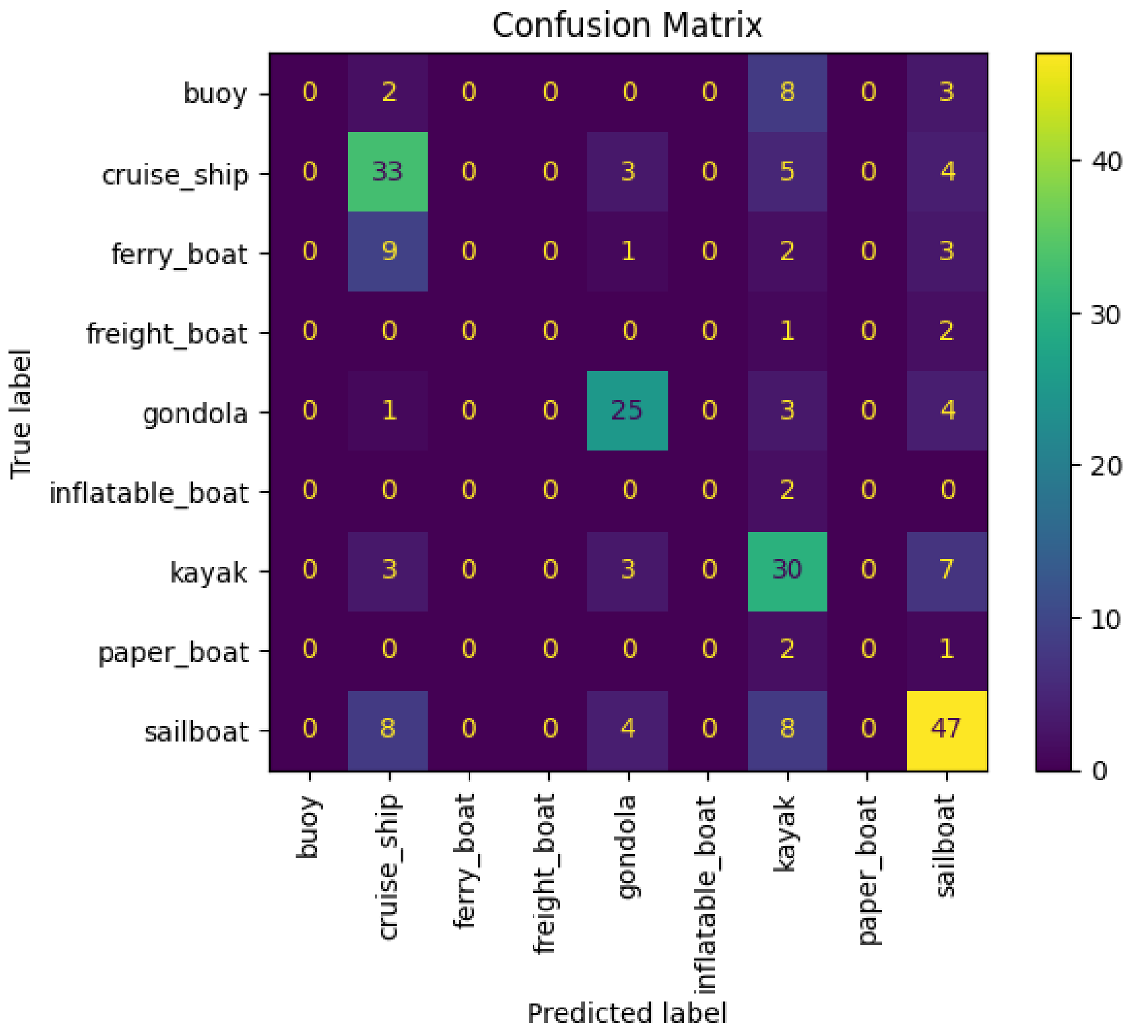
##Get the predicted and True labels

```
true_labels, predicted_labels = get_metrics(model,val_ds)
```

##Display Confusion Matrix

```
display_cm(true_labels,predicted_labels)
```

```
Confusion Matrix
[[ 0  2  0  0  0  0  8  0  3]
 [ 0 33  0  0  3  0  5  0  4]
 [ 0  9  0  0  1  0  2  0  3]
 [ 0  0  0  0  0  0  1  0  2]
 [ 0  1  0  0 25  0  3  0  4]
 [ 0  0  0  0  0  0  2  0  0]
 [ 0  3  0  0  3  0 30  0  7]
 [ 0  0  0  0  0  0  2  0  1]
 [ 0  8  0  0  4  0  8  0 47]]

Confusion Matrix Displayed


<Figure size 1000x1000 with 0 Axes>
```



As expected Sailboat,Kayak,Gondola and Cruise ship has high true positive values because of large number of images in the training data. Most mis-classifications are seen in the other image classes

##Classification Report

```
cl_report = classification_report(true_labels,predicted_labels)
print(cl_report)

print('Index to name mapping:\n')

for k,v in class_dict.items():
    print(f'{k} - > {v}')
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        13
           1       0.59      0.73      0.65        45
           2       0.00      0.00      0.00        15
```

```
           3       0.00      0.00      0.00         3
           4       0.69      0.76      0.72        33
           5       0.00      0.00      0.00         2
           6       0.49      0.70      0.58        43
           7       0.00      0.00      0.00         3
           8       0.66      0.70      0.68        67

    accuracy                           0.60       224
   macro avg       0.27      0.32      0.29       224
weighted avg       0.51      0.60      0.55       224

Index to name mapping:

0 - > inflatable_boat
1 - > freight_boat
2 - > paper_boat
3 - > buoy
4 - > ferry_boat
5 - > cruise_ship
6 - > gondola
7 - > kayak
8 - > sailboat
```

As expected the classes with higher support have the better overall scores

#Transfer Learning with MobileNet application

##Import mobilenet_v2 and freeze the weights

```
from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
mob_model = tf.keras.applications.MobileNet(
    input_shape=(height,width,3),
    include_top=False,
    weights="imagenet",
    input_tensor=None,
    pooling=None,
    classes=1000,
    classifier_activation="softmax",
    name='MobileNet',

)
mob_model.trainable=False
mob_model.summary()

Downloading data from https://storage.googleapis.com/tensorflow/keras-
applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5
17225924/17225924 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step

Model: "MobileNet"
```

Layer (type)	Output Shape	Param #
input_layer_2 (InputLayer)	(None, 224, 224, 3)	0
conv1 (Conv2D)	(None, 112, 112, 32)	864
conv1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv1_relu (ReLU)	(None, 112, 112, 32)	0
conv_dw_1 (DepthwiseConv2D)	(None, 112, 112, 32)	288
conv_dw_1_bn (BatchNormalization)	(None, 112, 112, 32)	128
conv_dw_1_relu (ReLU)	(None, 112, 112, 32)	0
conv_pw_1 (Conv2D)	(None, 112, 112, 64)	2,048
conv_pw_1_bn (BatchNormalization)	(None, 112, 112, 64)	256
conv_pw_1_relu (ReLU)	(None, 112, 112, 64)	0
conv_pad_2 (ZeroPadding2D)	(None, 113, 113, 64)	0

```
│ conv_dw_2 (DepthwiseConv2D)          │ (None, 56, 56, 64)          │
576 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_dw_2_bn (BatchNormalization)    │ (None, 56, 56, 64)          │
256 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_dw_2_relu (ReLU)                │ (None, 56, 56, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_pw_2 (Conv2D)                   │ (None, 56, 56, 128)         │
8,192 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_pw_2_bn (BatchNormalization)    │ (None, 56, 56, 128)         │
512 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_pw_2_relu (ReLU)                │ (None, 56, 56, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_dw_3 (DepthwiseConv2D)          │ (None, 56, 56, 128)         │
1,152 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_dw_3_bn (BatchNormalization)    │ (None, 56, 56, 128)         │
512 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_dw_3_relu (ReLU)                │ (None, 56, 56, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_pw_3 (Conv2D)                   │ (None, 56, 56, 128)         │
16,384 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_pw_3_bn (BatchNormalization)    │ (None, 56, 56, 128)         │
512 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_pw_3_relu (ReLU)                │ (None, 56, 56, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┤
─────────────────┤
│ conv_pad_4 (ZeroPadding2D)           │ (None, 57, 57, 128)         │
0 │
```

```
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_4 (DepthwiseConv2D)          │ (None, 28, 28, 128)         │
1,152 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_4_bn (BatchNormalization)    │ (None, 28, 28, 128)         │
512 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_4_relu (ReLU)                │ (None, 28, 28, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_4 (Conv2D)                   │ (None, 28, 28, 256)         │
32,768 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_4_bn (BatchNormalization)    │ (None, 28, 28, 256)         │
1,024 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_4_relu (ReLU)                │ (None, 28, 28, 256)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_5 (DepthwiseConv2D)          │ (None, 28, 28, 256)         │
2,304 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_5_bn (BatchNormalization)    │ (None, 28, 28, 256)         │
1,024 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_5_relu (ReLU)                │ (None, 28, 28, 256)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_5 (Conv2D)                   │ (None, 28, 28, 256)         │
65,536 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_5_bn (BatchNormalization)    │ (None, 28, 28, 256)         │
1,024 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_5_relu (ReLU)                │ (None, 28, 28, 256)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
```

```
│ conv_pad_6 (ZeroPadding2D)          │ (None, 29, 29, 256)          │
0 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_dw_6 (DepthwiseConv2D)         │ (None, 14, 14, 256)          │
2,304 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_dw_6_bn (BatchNormalization)   │ (None, 14, 14, 256)          │
1,024 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_dw_6_relu (ReLU)               │ (None, 14, 14, 256)          │
0 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_pw_6 (Conv2D)                  │ (None, 14, 14, 512)          │
131,072 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_pw_6_bn (BatchNormalization)   │ (None, 14, 14, 512)          │
2,048 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_pw_6_relu (ReLU)               │ (None, 14, 14, 512)          │
0 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_dw_7 (DepthwiseConv2D)         │ (None, 14, 14, 512)          │
4,608 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_dw_7_bn (BatchNormalization)   │ (None, 14, 14, 512)          │
2,048 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_dw_7_relu (ReLU)               │ (None, 14, 14, 512)          │
0 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_pw_7 (Conv2D)                  │ (None, 14, 14, 512)          │
262,144 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_pw_7_bn (BatchNormalization)   │ (None, 14, 14, 512)          │
2,048 │
├─────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ conv_pw_7_relu (ReLU)               │ (None, 14, 14, 512)          │
0 │
```

```
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_dw_8 (DepthwiseConv2D)         │ (None, 14, 14, 512)           │
4,608 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_dw_8_bn (BatchNormalization)   │ (None, 14, 14, 512)           │
2,048 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_dw_8_relu (ReLU)               │ (None, 14, 14, 512)           │
0 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_pw_8 (Conv2D)                  │ (None, 14, 14, 512)           │
262,144 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_pw_8_bn (BatchNormalization)   │ (None, 14, 14, 512)           │
2,048 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_pw_8_relu (ReLU)               │ (None, 14, 14, 512)           │
0 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_dw_9 (DepthwiseConv2D)         │ (None, 14, 14, 512)           │
4,608 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_dw_9_bn (BatchNormalization)   │ (None, 14, 14, 512)           │
2,048 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_dw_9_relu (ReLU)               │ (None, 14, 14, 512)           │
0 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_pw_9 (Conv2D)                  │ (None, 14, 14, 512)           │
262,144 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_pw_9_bn (BatchNormalization)   │ (None, 14, 14, 512)           │
2,048 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
│ conv_pw_9_relu (ReLU)               │ (None, 14, 14, 512)           │
0 │
├─────────────────────────────────────┼───────────────────────────────┼
─────────────────┤
```

```
│ conv_dw_10 (DepthwiseConv2D)           │ (None, 14, 14, 512)            │
4,608 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_dw_10_bn (BatchNormalization)     │ (None, 14, 14, 512)            │
2,048 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_dw_10_relu (ReLU)                 │ (None, 14, 14, 512)            │
0 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_pw_10 (Conv2D)                    │ (None, 14, 14, 512)            │
262,144 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_pw_10_bn (BatchNormalization)     │ (None, 14, 14, 512)            │
2,048 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_pw_10_relu (ReLU)                 │ (None, 14, 14, 512)            │
0 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_dw_11 (DepthwiseConv2D)           │ (None, 14, 14, 512)            │
4,608 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_dw_11_bn (BatchNormalization)     │ (None, 14, 14, 512)            │
2,048 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_dw_11_relu (ReLU)                 │ (None, 14, 14, 512)            │
0 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_pw_11 (Conv2D)                    │ (None, 14, 14, 512)            │
262,144 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_pw_11_bn (BatchNormalization)     │ (None, 14, 14, 512)            │
2,048 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_pw_11_relu (ReLU)                 │ (None, 14, 14, 512)            │
0 │
├────────────────────────────────────────┼────────────────────────────────┼
─────────────────┤
│ conv_pad_12 (ZeroPadding2D)            │ (None, 15, 15, 512)            │
0 │
```

```
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_12 (DepthwiseConv2D)         │ (None, 7, 7, 512)           │
4,608 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_12_bn (BatchNormalization)   │ (None, 7, 7, 512)           │
2,048 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_12_relu (ReLU)               │ (None, 7, 7, 512)           │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_12 (Conv2D)                  │ (None, 7, 7, 1024)          │
524,288 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_12_bn (BatchNormalization)   │ (None, 7, 7, 1024)          │
4,096 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_12_relu (ReLU)               │ (None, 7, 7, 1024)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_13 (DepthwiseConv2D)         │ (None, 7, 7, 1024)          │
9,216 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_13_bn (BatchNormalization)   │ (None, 7, 7, 1024)          │
4,096 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_dw_13_relu (ReLU)               │ (None, 7, 7, 1024)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_13 (Conv2D)                  │ (None, 7, 7, 1024)          │
1,048,576 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_13_bn (BatchNormalization)   │ (None, 7, 7, 1024)          │
4,096 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv_pw_13_relu (ReLU)               │ (None, 7, 7, 1024)          │
0 │
```

```
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 3,228,864 (12.32 MB)

 Trainable params: 0 (0.00 B)

 Non-trainable params: 3,228,864 (12.32 MB)
```

##Build an FCN

##Get training and validation set at 70:30 ratio and seed 1

```
train_ds = get_ds(data_dir,validation_split=0.3,subset='training',seed
= 1,image_size=(height,width),batch_size=batch_size)
val_ds = get_ds(data_dir,validation_split=0.3,subset='validation',seed
= 1,image_size=(height,width),batch_size=batch_size)

Found 1162 files belonging to 9 classes.
Using 814 files for training.
Found 1162 files belonging to 9 classes.
Using 348 files for validation.
```

##Set up dataset with caching and prefetch

```
AUTOTUNE = tf.data.AUTOTUNE

train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
```

##Add a FCN to the mobile net model(no data augmentation)

```
model_t = create_transfer_model(mob_model)
model_t.name = 'Transfer_Model'

Model: "sequential_2"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ rescaling_2 (Rescaling)              │ (None, 224, 224, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ MobileNet (Functional)               │ (None, 7, 7, 1024)          │
```

```
3,228,864 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ global_average_pooling2d_2           │ (None, 1024)                 │
0 │
│ (GlobalAveragePooling2D)             │                              │
│
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dropout_5 (Dropout)                  │ (None, 1024)                 │
0 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ flatten_2 (Flatten)                  │ (None, 1024)                 │
0 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dense_6 (Dense)                      │ (None, 256)                  │
262,400 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ batch_normalization                  │ (None, 256)                  │
1,024 │
│ (BatchNormalization)                 │                              │
│
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dropout_6 (Dropout)                  │ (None, 256)                  │
0 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dense_7 (Dense)                      │ (None, 128)                  │
32,896 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ batch_normalization_1                │ (None, 128)                  │
512 │
│ (BatchNormalization)                 │                              │
│
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dropout_7 (Dropout)                  │ (None, 128)                  │
0 │
├──────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ dense_8 (Dense)                      │ (None, 9)                    │
1,161 │
└──────────────────────────────────────┴──────────────────────────────┴
─────────────────┘
```

```
 Total params: 3,526,857 (13.45 MB)

 Trainable params: 297,225 (1.13 MB)

 Non-trainable params: 3,229,632 (12.32 MB)
```

##Complile the model

```
model_t.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
print('model_t compiled')

model_t compiled
```

##Configure for early stopping and Checkpoint and train the model

```
early_stop=EarlyStopping(monitor='val_accuracy',mode='max',verbose=1,p
atience=10,min_delta=0.01)
checkpoint=ModelCheckpoint('best_model_t.keras',monitor='val_accuracy'
,mode='max',verbose=1,save_best_only=True)

epochs=100
history_t = model_t.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  batch_size = batch_size,
  callbacks = [early_stop,checkpoint]
)

histories.append(history_t)

Epoch 1/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 220ms/step - accuracy: 0.4264 - loss:
1.8711
Epoch 1: val_accuracy improved from -inf to 0.67529, saving model to
best_model_t.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 28s 513ms/step - accuracy: 0.4328 - loss:
1.8516 - val_accuracy: 0.6753 - val_loss: 1.0619
Epoch 2/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8425 - loss:
0.5144
Epoch 2: val_accuracy improved from 0.67529 to 0.77874, saving model
to best_model_t.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.8447 - loss:
0.5084 - val_accuracy: 0.7787 - val_loss: 0.7892
Epoch 3/100
24/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9499 - loss:
0.2008
```

```
Epoch 3: val_accuracy improved from 0.77874 to 0.80172, saving model
to best_model_t.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9506 - loss:
0.2012 - val_accuracy: 0.8017 - val_loss: 0.6515
Epoch 4/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9720 - loss:
0.1307
Epoch 4: val_accuracy improved from 0.80172 to 0.81897, saving model
to best_model_t.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 52ms/step - accuracy: 0.9725 - loss:
0.1305 - val_accuracy: 0.8190 - val_loss: 0.5712
Epoch 5/100
24/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9926 - loss:
0.0737
Epoch 5: val_accuracy improved from 0.81897 to 0.83046, saving model
to best_model_t.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.9921 - loss:
0.0745 - val_accuracy: 0.8305 - val_loss: 0.5724
Epoch 6/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9859 - loss:
0.0759
Epoch 6: val_accuracy improved from 0.83046 to 0.85057, saving model
to best_model_t.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.9863 - loss:
0.0754 - val_accuracy: 0.8506 - val_loss: 0.5604
Epoch 7/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9990 - loss:
0.0547
Epoch 7: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9988 - loss:
0.0545 - val_accuracy: 0.8391 - val_loss: 0.5507
Epoch 8/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 1.0000 - loss:
0.0334
Epoch 8: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 1.0000 - loss:
0.0336 - val_accuracy: 0.8506 - val_loss: 0.5217
Epoch 9/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9998 - loss:
0.0245
Epoch 9: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9997 - loss:
0.0247 - val_accuracy: 0.8506 - val_loss: 0.5314
Epoch 10/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9995 - loss:
0.0188
Epoch 10: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9993 - loss:
0.0192 - val_accuracy: 0.8477 - val_loss: 0.5483
```

```
Epoch 11/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9997 - loss:
0.0218
Epoch 11: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9994 - loss:
0.0222 - val_accuracy: 0.8420 - val_loss: 0.5499
Epoch 12/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9951 - loss:
0.0248
Epoch 12: val_accuracy improved from 0.85057 to 0.86207, saving model
to best_model_t.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.9952 - loss:
0.0249 - val_accuracy: 0.8621 - val_loss: 0.5267
Epoch 13/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 1.0000 - loss:
0.0149
Epoch 13: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 1.0000 - loss:
0.0150 - val_accuracy: 0.8506 - val_loss: 0.5698
Epoch 14/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 1.0000 - loss:
0.0154
Epoch 14: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 1.0000 - loss:
0.0154 - val_accuracy: 0.8621 - val_loss: 0.5414
Epoch 15/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 1.0000 - loss:
0.0099
Epoch 15: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 1.0000 - loss:
0.0100 - val_accuracy: 0.8534 - val_loss: 0.5469
Epoch 16/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9983 - loss:
0.0114
Epoch 16: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9983 - loss:
0.0114 - val_accuracy: 0.8506 - val_loss: 0.5624
Epoch 17/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 1.0000 - loss:
0.0088
Epoch 17: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 36ms/step - accuracy: 1.0000 - loss:
0.0089 - val_accuracy: 0.8563 - val_loss: 0.5694
Epoch 18/100
24/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 1.0000 - loss:
0.0099
Epoch 18: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 1.0000 - loss:
0.0097 - val_accuracy: 0.8362 - val_loss: 0.6015
```

```
Epoch 19/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9988 - loss:
0.0097
Epoch 19: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9988 - loss:
0.0097 - val_accuracy: 0.8362 - val_loss: 0.5923
Epoch 20/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9975 - loss:
0.0217
Epoch 20: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9976 - loss:
0.0212 - val_accuracy: 0.8477 - val_loss: 0.5721
Epoch 21/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 1.0000 - loss:
0.0094
Epoch 21: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 1.0000 - loss:
0.0093 - val_accuracy: 0.8534 - val_loss: 0.5480
Epoch 22/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9971 - loss:
0.0094
Epoch 22: val_accuracy did not improve from 0.86207
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9972 - loss:
0.0092 - val_accuracy: 0.8563 - val_loss: 0.5547
Epoch 22: early stopping
```

##Print metrics

```python
from tensorflow.keras.models import load_model
model_t = load_model('best_model_t.keras')
print_scores(model_t,val_ds,train_ds,'Transfer Model')
```

```
Displaying accuracy and loss for Transfer Model
Test loss:0.5267
Test accuracy:0.8621
Train loss:0.0102
train accuracy:0.9988
```

Severe overfitting can be observed. We will modify this model by increasing dropouts. The below model uses a droput that i have arrived at after trial and error which produced lesser overfitting

##Dropout adjusted Transfer Model(no augmentation)

```python
model_t1 = create_transfer_model(mob_model,dropout=0.4)
```

```
Model: "sequential_3"
```

Layer (type)	Output Shape	Param #
rescaling_3 (Rescaling)	(None, 224, 224, 3)	0
MobileNet (Functional)	(None, 7, 7, 1024)	3,228,864
global_average_pooling2d_3 (GlobalAveragePooling2D)	(None, 1024)	0
dropout_8 (Dropout)	(None, 1024)	0
flatten_3 (Flatten)	(None, 1024)	0
dense_9 (Dense)	(None, 256)	262,400
batch_normalization_2 (BatchNormalization)	(None, 256)	1,024
dropout_9 (Dropout)	(None, 256)	0
dense_10 (Dense)	(None, 128)	32,896
batch_normalization_3 (BatchNormalization)	(None, 128)	512

```
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_10 (Dropout)                 │ (None, 128)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_11 (Dense)                     │ (None, 9)                   │
1,161 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 3,526,857 (13.45 MB)

 Trainable params: 297,225 (1.13 MB)

 Non-trainable params: 3,229,632 (12.32 MB)
```

##Compile the model

```
model_t1.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
print('model_t1 compiled')

model_t1 compiled
```

##Train the model

```
early_stop=EarlyStopping(monitor='val_accuracy',mode='max',verbose=1,p
atience=10,min_delta=0.01)
checkpoint=ModelCheckpoint('best_model_t1.keras',monitor='val_accuracy
',mode='max',verbose=1,save_best_only=True)

epochs=100
history_t1 = model_t1.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  batch_size = batch_size,
  callbacks = [early_stop,checkpoint]
)

histories.append(history_t1)

Epoch 1/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 149ms/step - accuracy: 0.2050 - loss:
2.7388
Epoch 1: val_accuracy improved from -inf to 0.52874, saving model to
best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 15s 313ms/step - accuracy: 0.2133 - loss:
2.7025 - val_accuracy: 0.5287 - val_loss: 1.4464
```

```
Epoch 2/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.5586 - loss:
1.4237
Epoch 2: val_accuracy improved from 0.52874 to 0.70977, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.5644 - loss:
1.4091 - val_accuracy: 0.7098 - val_loss: 1.0327
Epoch 3/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.7462 - loss:
0.8408
Epoch 3: val_accuracy improved from 0.70977 to 0.76724, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.7441 - loss:
0.8477 - val_accuracy: 0.7672 - val_loss: 0.8145
Epoch 4/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8037 - loss:
0.6875
Epoch 4: val_accuracy improved from 0.76724 to 0.80172, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 44ms/step - accuracy: 0.8027 - loss:
0.6895 - val_accuracy: 0.8017 - val_loss: 0.6673
Epoch 5/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8155 - loss:
0.6046
Epoch 5: val_accuracy did not improve from 0.80172
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8143 - loss:
0.6059 - val_accuracy: 0.7902 - val_loss: 0.6345
Epoch 6/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.7889 - loss:
0.6165
Epoch 6: val_accuracy improved from 0.80172 to 0.82184, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 43ms/step - accuracy: 0.7899 - loss:
0.6152 - val_accuracy: 0.8218 - val_loss: 0.6081
Epoch 7/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8097 - loss:
0.6110
Epoch 7: val_accuracy improved from 0.82184 to 0.83046, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 46ms/step - accuracy: 0.8105 - loss:
0.6066 - val_accuracy: 0.8305 - val_loss: 0.5926
Epoch 8/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8504 - loss:
0.4538
Epoch 8: val_accuracy did not improve from 0.83046
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8499 - loss:
0.4551 - val_accuracy: 0.8276 - val_loss: 0.5940
Epoch 9/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8656 - loss:
```

```
0.4162
Epoch 9: val_accuracy did not improve from 0.83046
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.8653 - loss:
0.4172 - val_accuracy: 0.8276 - val_loss: 0.5682
Epoch 10/100
24/26 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.8694 - loss:
0.3948
Epoch 10: val_accuracy improved from 0.83046 to 0.83908, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 51ms/step - accuracy: 0.8682 - loss:
0.3973 - val_accuracy: 0.8391 - val_loss: 0.5561
Epoch 11/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8981 - loss:
0.3393
Epoch 11: val_accuracy did not improve from 0.83908
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 32ms/step - accuracy: 0.8967 - loss:
0.3436 - val_accuracy: 0.8362 - val_loss: 0.5802
Epoch 12/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.8543 - loss:
0.4477
Epoch 12: val_accuracy did not improve from 0.83908
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.8558 - loss:
0.4441 - val_accuracy: 0.8391 - val_loss: 0.5603
Epoch 13/100
24/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.8836 - loss:
0.3455
Epoch 13: val_accuracy improved from 0.83908 to 0.84483, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 44ms/step - accuracy: 0.8836 - loss:
0.3455 - val_accuracy: 0.8448 - val_loss: 0.5482
Epoch 14/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8809 - loss:
0.3696
Epoch 14: val_accuracy did not improve from 0.84483
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8827 - loss:
0.3661 - val_accuracy: 0.8362 - val_loss: 0.5369
Epoch 15/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.8954 - loss:
0.3131
Epoch 15: val_accuracy did not improve from 0.84483
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.8961 - loss:
0.3114 - val_accuracy: 0.8391 - val_loss: 0.5517
Epoch 16/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9285 - loss:
0.2531
Epoch 16: val_accuracy did not improve from 0.84483
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9278 - loss:
0.2533 - val_accuracy: 0.8391 - val_loss: 0.5259
Epoch 17/100
```

```
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9241 - loss:
0.2381
Epoch 17: val_accuracy did not improve from 0.84483
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9240 - loss:
0.2376 - val_accuracy: 0.8276 - val_loss: 0.5429
Epoch 18/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9418 - loss:
0.2094
Epoch 18: val_accuracy improved from 0.84483 to 0.84770, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 46ms/step - accuracy: 0.9413 - loss:
0.2100 - val_accuracy: 0.8477 - val_loss: 0.5082
Epoch 19/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9219 - loss:
0.2738
Epoch 19: val_accuracy did not improve from 0.84770
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9218 - loss:
0.2722 - val_accuracy: 0.8477 - val_loss: 0.5157
Epoch 20/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9297 - loss:
0.2062
Epoch 20: val_accuracy improved from 0.84770 to 0.85920, saving model
to best_model_t1.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 49ms/step - accuracy: 0.9297 - loss:
0.2063 - val_accuracy: 0.8592 - val_loss: 0.5220
Epoch 21/100
24/26 ━━━━━━━━━━━━━━━━━━━━ 0s 24ms/step - accuracy: 0.9489 - loss:
0.1606
Epoch 21: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 34ms/step - accuracy: 0.9475 - loss:
0.1644 - val_accuracy: 0.8506 - val_loss: 0.5491
Epoch 22/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9496 - loss:
0.1773
Epoch 22: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9495 - loss:
0.1768 - val_accuracy: 0.8534 - val_loss: 0.5619
Epoch 23/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9426 - loss:
0.1617
Epoch 23: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 33ms/step - accuracy: 0.9423 - loss:
0.1627 - val_accuracy: 0.8506 - val_loss: 0.5470
Epoch 24/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9480 - loss:
0.1690
Epoch 24: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9478 - loss:
0.1691 - val_accuracy: 0.8477 - val_loss: 0.5443
```

```
Epoch 25/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9381 - loss:
0.2156
Epoch 25: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9390 - loss:
0.2122 - val_accuracy: 0.8420 - val_loss: 0.5659
Epoch 26/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9519 - loss:
0.1638
Epoch 26: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9515 - loss:
0.1641 - val_accuracy: 0.8563 - val_loss: 0.5769
Epoch 27/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9451 - loss:
0.1543
Epoch 27: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9457 - loss:
0.1533 - val_accuracy: 0.8448 - val_loss: 0.5686
Epoch 28/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9573 - loss:
0.1228
Epoch 28: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9574 - loss:
0.1234 - val_accuracy: 0.8534 - val_loss: 0.5494
Epoch 29/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 22ms/step - accuracy: 0.9319 - loss:
0.1751
Epoch 29: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9328 - loss:
0.1737 - val_accuracy: 0.8506 - val_loss: 0.5280
Epoch 30/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 23ms/step - accuracy: 0.9540 - loss:
0.1456
Epoch 30: val_accuracy did not improve from 0.85920
26/26 ━━━━━━━━━━━━━━━━━━━━ 1s 32ms/step - accuracy: 0.9538 - loss:
0.1462 - val_accuracy: 0.8592 - val_loss: 0.5285
Epoch 30: early stopping
```

##Print metrics

```
from tensorflow.keras.models import load_model
model_t1 = load_model('best_model_t1.keras')
print_scores(model_t1,val_ds,train_ds,'Dropout Adjusted Transfer
Model')
# plot_accuracy_loss_graphs(history_t1,'Dropout adjusted transfer
model')

Displaying accuracy and loss for Dropout Adjusted Transfer Model
Test loss:0.5220
```

```
Test accuracy:0.8592
Train loss:0.0325
train accuracy:0.9963
```

Increaing dropout to 0.4 has reduced overfitting a little

##Augmented Transfer Model

```
data_augmentation =
[layers.RandomFlip("horizontal",input_shape=(height,width,3)),layers.R
andomRotation(0.1),layers.RandomZoom(0.1)]
model_augmented =
create_transfer_model(mob_model,dropout=0.4,data_augmentation=data_aug
mentation)
```

```
added <RandomFlip name=random_flip, built=False> to model
added <RandomRotation name=random_rotation, built=False> to model
added <RandomZoom name=random_zoom, built=False> to model
```

```
Model: "sequential_4"
```

```
┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                          ┃ Output Shape                 ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇
━━━━━━━━━━━━━━━━━┩
│ random_flip (RandomFlip)              │ (None, 224, 224, 3)          │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ random_rotation (RandomRotation)      │ (None, 224, 224, 3)          │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ random_zoom (RandomZoom)              │ (None, 224, 224, 3)          │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ rescaling_4 (Rescaling)               │ (None, 224, 224, 3)          │
0 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ MobileNet (Functional)                │ (None, 7, 7, 1024)           │
3,228,864 │
├───────────────────────────────────────┼──────────────────────────────┼
─────────────────┤
│ global_average_pooling2d_4            │ (None, 1024)                 │
0 │
│ (GlobalAveragePooling2D)              │                              │
```

```
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_11 (Dropout)                 │ (None, 1024)                │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ flatten_4 (Flatten)                  │ (None, 1024)                │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_12 (Dense)                     │ (None, 256)                 │
262,400 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_4                │ (None, 256)                 │
1,024 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_12 (Dropout)                 │ (None, 256)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_13 (Dense)                     │ (None, 128)                 │
32,896 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_5                │ (None, 128)                 │
512 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_13 (Dropout)                 │ (None, 128)                 │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_14 (Dense)                     │ (None, 9)                   │
1,161 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 3,526,857 (13.45 MB)

 Trainable params: 297,225 (1.13 MB)

 Non-trainable params: 3,229,632 (12.32 MB)
```

##Compile the model

```
model_augmented.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])
print('model_augmented compiled')
```

```
model_augmented compiled
```

##Train the model

```
early_stop=EarlyStopping(monitor='val_accuracy',mode='max',verbose=1,p
atience=10,min_delta=0.01)
checkpoint=ModelCheckpoint('best_model_a.keras',monitor='val_accuracy'
,mode='max',verbose=1,save_best_only=True)

epochs=100
history_a = model_augmented.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs,
  batch_size = batch_size,
  callbacks = [early_stop,checkpoint]
)

histories.append(history_a)
```

```
Epoch 1/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 59ms/step - accuracy: 0.1796 - loss:
2.9263
Epoch 1: val_accuracy improved from -inf to 0.49425, saving model to
best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 11s 143ms/step - accuracy: 0.1832 - loss:
2.9111 - val_accuracy: 0.4943 - val_loss: 1.5626
Epoch 2/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.4971 - loss:
1.6661
Epoch 2: val_accuracy improved from 0.49425 to 0.67529, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 80ms/step - accuracy: 0.4999 - loss:
1.6569 - val_accuracy: 0.6753 - val_loss: 1.2297
Epoch 3/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.6437 - loss:
1.1335
Epoch 3: val_accuracy improved from 0.67529 to 0.72126, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 80ms/step - accuracy: 0.6434 - loss:
1.1357 - val_accuracy: 0.7213 - val_loss: 0.9651
Epoch 4/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 43ms/step - accuracy: 0.6995 - loss:
1.0396
```

```
Epoch 4: val_accuracy improved from 0.72126 to 0.75862, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 72ms/step - accuracy: 0.6987 - loss:
1.0411 - val_accuracy: 0.7586 - val_loss: 0.7938
Epoch 5/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.7338 - loss:
0.8352
Epoch 5: val_accuracy improved from 0.75862 to 0.79023, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 73ms/step - accuracy: 0.7333 - loss:
0.8383 - val_accuracy: 0.7902 - val_loss: 0.7027
Epoch 6/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 48ms/step - accuracy: 0.7402 - loss:
0.8620
Epoch 6: val_accuracy improved from 0.79023 to 0.80460, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 3s 83ms/step - accuracy: 0.7401 - loss:
0.8614 - val_accuracy: 0.8046 - val_loss: 0.6599
Epoch 7/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.7661 - loss:
0.8005
Epoch 7: val_accuracy did not improve from 0.80460
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.7661 - loss:
0.8008 - val_accuracy: 0.7902 - val_loss: 0.6540
Epoch 8/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.7380 - loss:
0.7764
Epoch 8: val_accuracy did not improve from 0.80460
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.7401 - loss:
0.7736 - val_accuracy: 0.8046 - val_loss: 0.6129
Epoch 9/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.7636 - loss:
0.8041
Epoch 9: val_accuracy did not improve from 0.80460
26/26 ━━━━━━━━━━━━━━━━━━━━ 3s 61ms/step - accuracy: 0.7645 - loss:
0.7990 - val_accuracy: 0.8017 - val_loss: 0.6112
Epoch 10/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.7714 - loss:
0.7370
Epoch 10: val_accuracy improved from 0.80460 to 0.83046, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 73ms/step - accuracy: 0.7727 - loss:
0.7311 - val_accuracy: 0.8305 - val_loss: 0.5695
Epoch 11/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.8146 - loss:
0.5684
Epoch 11: val_accuracy improved from 0.83046 to 0.83908, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 73ms/step - accuracy: 0.8141 - loss:
```

```
0.5716 - val_accuracy: 0.8391 - val_loss: 0.5502
Epoch 12/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.8190 - loss:
0.5820
Epoch 12: val_accuracy improved from 0.83908 to 0.84770, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 88ms/step - accuracy: 0.8187 - loss:
0.5838 - val_accuracy: 0.8477 - val_loss: 0.5192
Epoch 13/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 49ms/step - accuracy: 0.8158 - loss:
0.5652
Epoch 13: val_accuracy improved from 0.84770 to 0.85057, saving model
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 78ms/step - accuracy: 0.8155 - loss:
0.5666 - val_accuracy: 0.8506 - val_loss: 0.5072
Epoch 14/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.8519 - loss:
0.5318
Epoch 14: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.8511 - loss:
0.5323 - val_accuracy: 0.8333 - val_loss: 0.5093
Epoch 15/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.8508 - loss:
0.4784
Epoch 15: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.8488 - loss:
0.4832 - val_accuracy: 0.8391 - val_loss: 0.5050
Epoch 16/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.8357 - loss:
0.5292
Epoch 16: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.8354 - loss:
0.5294 - val_accuracy: 0.8448 - val_loss: 0.4983
Epoch 17/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.8329 - loss:
0.4961
Epoch 17: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 63ms/step - accuracy: 0.8341 - loss:
0.4923 - val_accuracy: 0.8362 - val_loss: 0.4988
Epoch 18/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.8718 - loss:
0.4101
Epoch 18: val_accuracy did not improve from 0.85057
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 61ms/step - accuracy: 0.8706 - loss:
0.4141 - val_accuracy: 0.8448 - val_loss: 0.5060
Epoch 19/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 47ms/step - accuracy: 0.8665 - loss:
0.3765
Epoch 19: val_accuracy did not improve from 0.85057
```

```
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 65ms/step - accuracy: 0.8663 - loss: 
0.3776 - val_accuracy: 0.8391 - val_loss: 0.5282
Epoch 20/100
26/26 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - accuracy: 0.8510 - loss: 
0.4585
Epoch 20: val_accuracy improved from 0.85057 to 0.85345, saving model 
to best_model_a.keras
26/26 ━━━━━━━━━━━━━━━━━━━━ 3s 75ms/step - accuracy: 0.8507 - loss: 
0.4588 - val_accuracy: 0.8534 - val_loss: 0.5163
Epoch 21/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.8501 - loss: 
0.4077
Epoch 21: val_accuracy did not improve from 0.85345
26/26 ━━━━━━━━━━━━━━━━━━━━ 2s 62ms/step - accuracy: 0.8486 - loss: 
0.4112 - val_accuracy: 0.8506 - val_loss: 0.4837
Epoch 22/100
25/26 ━━━━━━━━━━━━━━━━━━━━ 0s 44ms/step - accuracy: 0.8531 - loss: 
0.4400
Epoch 22: val_accuracy did not improve from 0.85345
26/26 ━━━━━━━━━━━━━━━━━━━━ 3s 62ms/step - accuracy: 0.8532 - loss: 
0.4389 - val_accuracy: 0.8506 - val_loss: 0.4906
Epoch 22: early stopping
```

##Print accuracy and loss curves

```
from tensorflow.keras.models import load_model
model_a = load_model('best_model_a.keras')
print_scores(model_t1,val_ds,train_ds,'Augmented Transfer Model')
# plot_accuracy_loss_graphs(history_a,'Augmented transfer model')

Displaying accuracy and loss for Augmented Transfer Model
Test loss:0.5220
Test accuracy:0.8592
Train loss:0.0325
train accuracy:0.9963
```

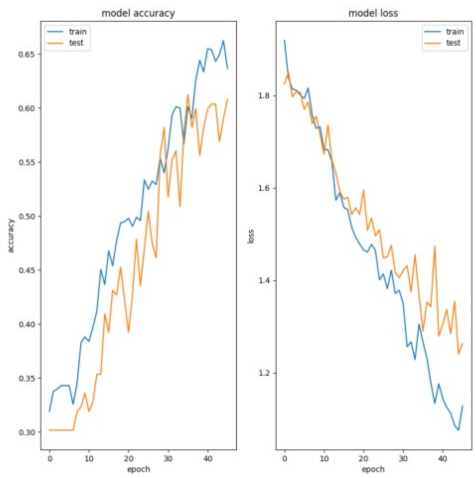
##Plot loss/accuracy curves for all models

The final accuracy and loss are the same for augmented ad non augmented models. The augmented model has a slightly lesser ovetfitting

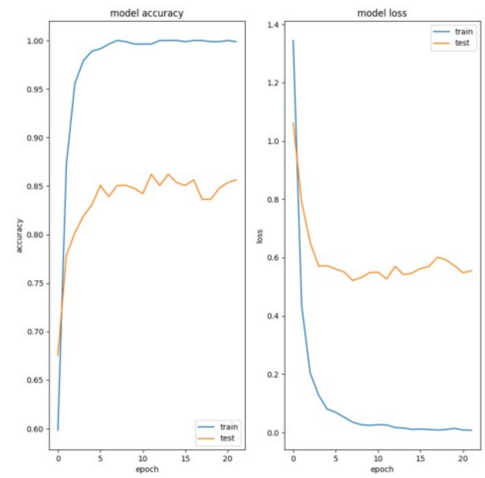
```
import matplotlib.image as mpimg
histories = [history_c,history_t,history_t1,history_a]
#titles = ['CNN Model','Transfer Learning model using 
MobileNet','Dropout adjusted transfer model','Augmented transfer 
model']
#print(test_accuracy.keys())

titles = list(test_accuracy.keys())
```

```
for i,history in enumerate(histories):
  plot_accuracy_loss_graphs(history,titles[i],show=False)
plt.figure(figsize=(16,16))
for i,title in enumerate(titles):
  title = titles[i]
  plt.subplot(2,2,i+1)
  figure = title+'.png'
  img = mpimg.imread(figure)
  plt.imshow(img)
  plt.axis('off')
  accuracy =  test_accuracy[title]
  loss = test_loss[title]
  title = title + f'\n(validation accuracy = {accuracy:.4f}, loss =
{loss:.4f})'
  plt.title(title)
plt.show()
```

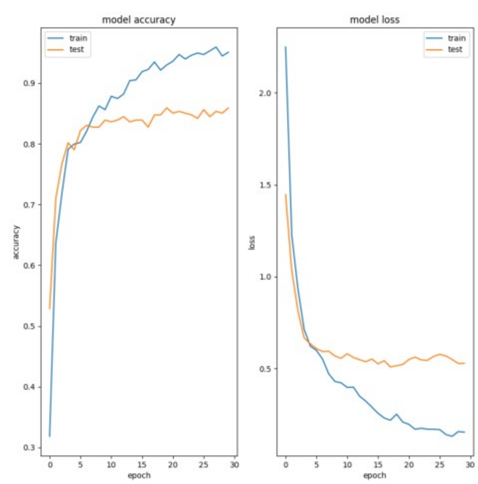
CNN Model
(validation accuracy = 0.6121, loss = 1.2901)



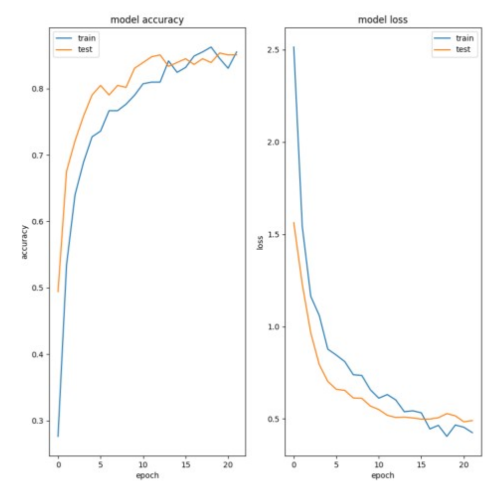
Transfer Model
(validation accuracy = 0.8621, loss = 0.5267)



Dropout Adjusted Transfer Model
(validation accuracy = 0.8592, loss = 0.5220)



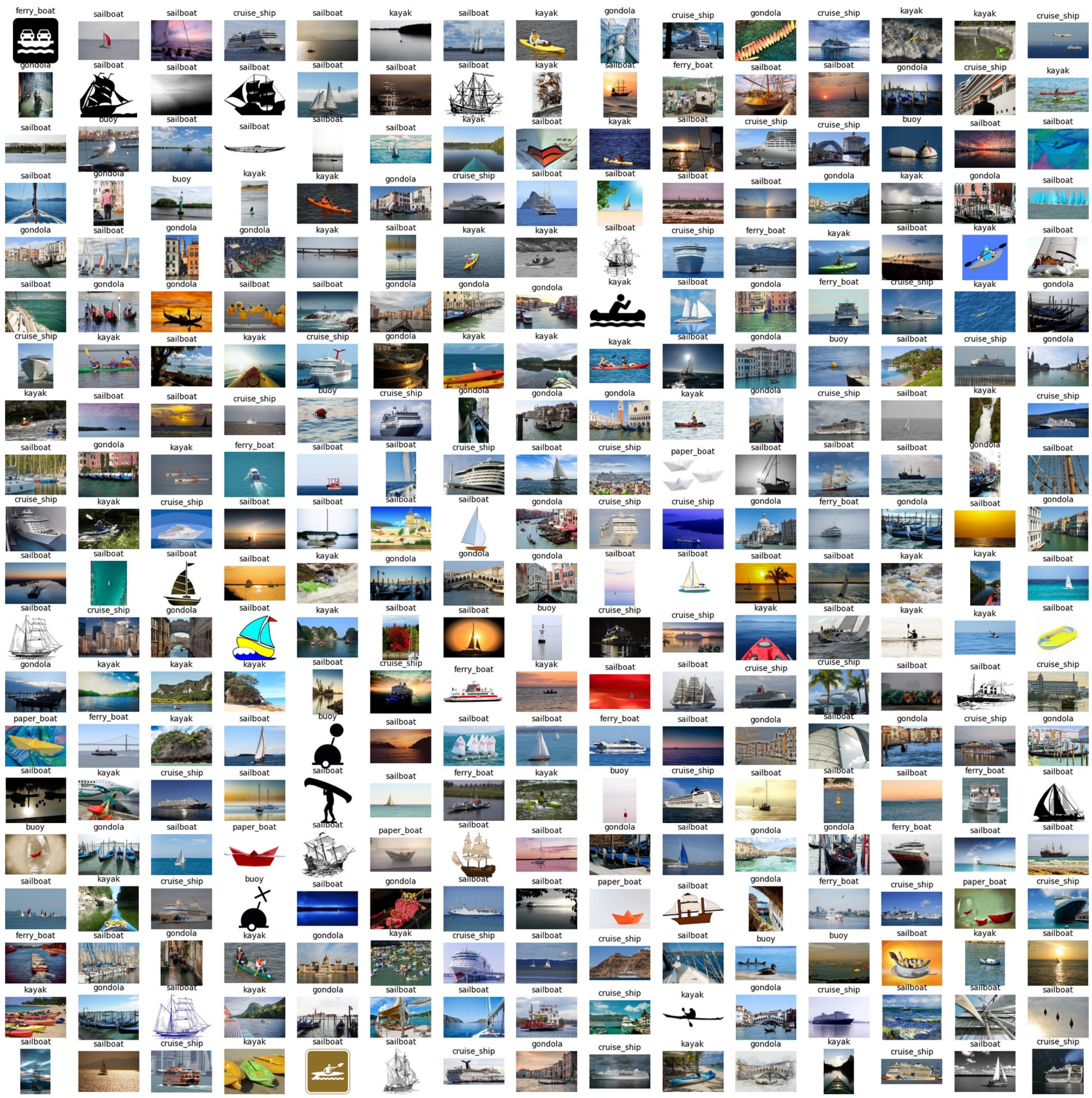
Augmented Transfer Model
(validation accuracy = 0.8592, loss = 0.5220)



##Visualize the predictions

```
image_map = get_predicted_labels(model_a,class_names)
print('size of image map = ',len(image_map))
display_test_images(image_map,'Transfer Learning model using
MobileNet. Title is the prediction')
```

```
size of image map =  300
Predictions for Transfer Learning model using MobileNet. Title is the
prediction
```



The predictions with the transfer model is much more accurate (expected as accuracy is 86%) when commpared to the CNN model. We can see that even classes with very few train images have been correctly identified(paper boat for example)

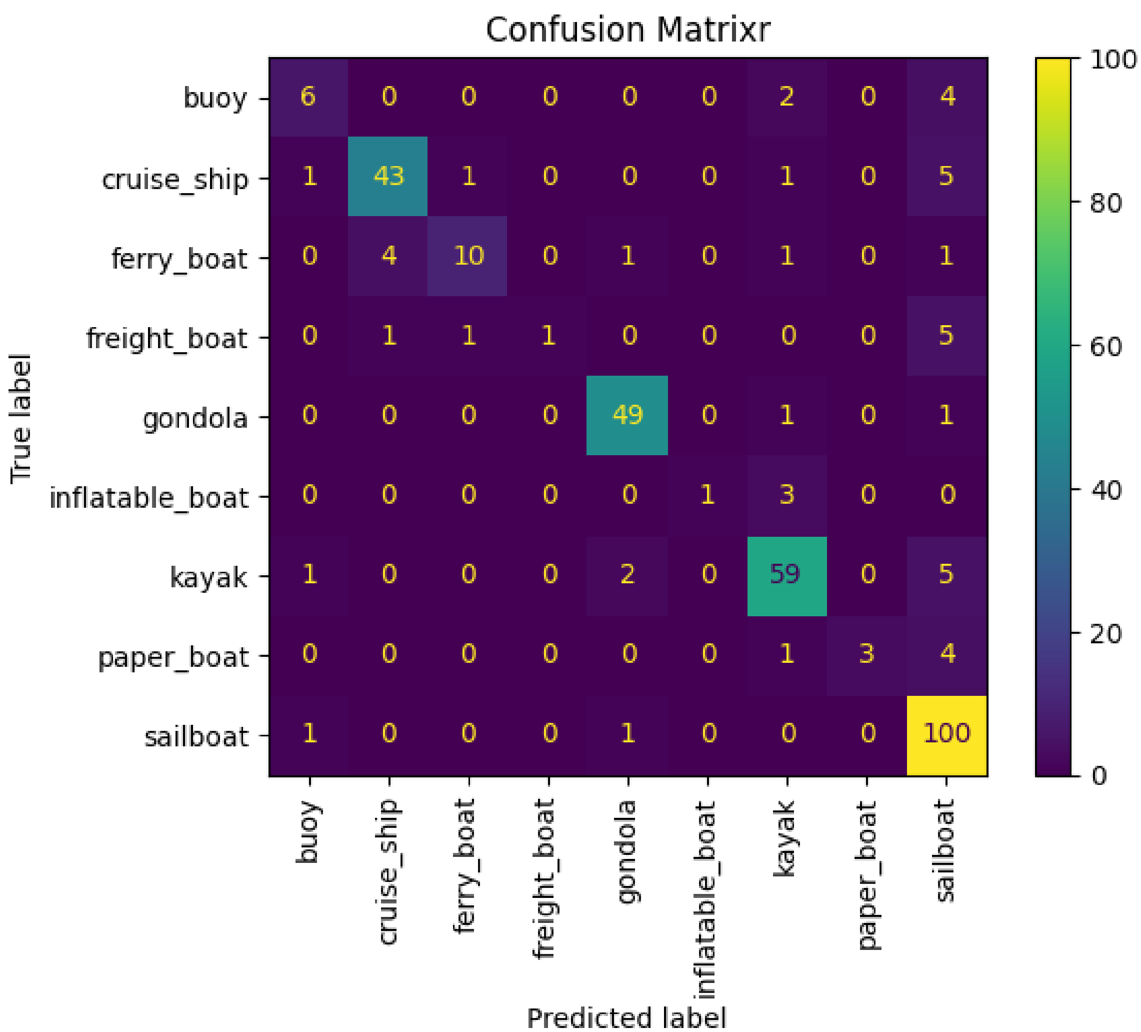
##Metrics

```
true_labels, predicted_labels =
get_metrics(model_a,val_ds,val_split=0.3)
```

##Confusion Matrix

```python
from sklearn.metrics import ConfusionMatrixDisplay
cm = confusion_matrix(true_labels,predicted_labels)
print('Confusion Matrix')
print(cm)
print()

print('Confusion Matrix Displayed\n')
plt.figure(figsize = (10,10))
disp =
ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=class_names)
disp.plot()
plt.title('Confusion Matrixr')
plt.xticks(rotation=90)
plt.show()
```

```
Confusion Matrix
[[  6   0   0   0   0   0   2   0   4]
 [  1  43   1   0   0   0   1   0   5]
 [  0   4  10   0   1   0   1   0   1]
 [  0   1   1   1   0   0   0   0   5]
 [  0   0   0   0  49   0   1   0   1]
 [  0   0   0   0   0   1   3   0   0]
 [  1   0   0   0   2   0  59   0   5]
 [  0   0   0   0   0   0   1   3   4]
 [  1   0   0   0   1   0   0   0 100]]

Confusion Matrix Displayed


<Figure size 1000x1000 with 0 Axes>
```



##Classification Report

```
from sklearn.metrics import classification_report

class_name_dict = {
    0 : ''
}

cl_report = classification_report(true_labels,predicted_labels)
print(cl_report)

print('Index to name mapping:\n')

for k,v in class_dict.items():
    print(f'{k} - > {v}')
```

```
              precision    recall  f1-score   support

           0       0.67      0.50      0.57        12
```

```
           1       0.90      0.84      0.87        51
           2       0.83      0.59      0.69        17
           3       1.00      0.12      0.22         8
           4       0.92      0.96      0.94        51
           5       1.00      0.25      0.40         4
           6       0.87      0.88      0.87        67
           7       1.00      0.38      0.55         8
           8       0.80      0.98      0.88       102

    accuracy                           0.85       320
   macro avg       0.89      0.61      0.67       320
weighted avg       0.86      0.85      0.83       320

Index to name mapping:

0 - > inflatable_boat
1 - > freight_boat
2 - > paper_boat
3 - > buoy
4 - > ferry_boat
5 - > cruise_ship
6 - > gondola
7 - > kayak
8 - > sailboat
```

As expected the classes that have higher support(more train images) have better overall scores

Inferences

##CNN Model

The CNN model built in part 1 of this project gave an accuracy of around 65%. When tested on the 300 test images, we could see many mis-classificatons. This could be because the training set contained just 1162 images and the code used only 80% of that for training. This is a very low number for the model to learn well. The model has a bit of overfitting, though :

##Transfer Learning

Accuracy significantly increases with transfer model (Mobilenet_V2) with fine tuning with custom FCN. However there is significant over fitting with low dropout values. With higher dropout values, overfitting gets reduced without sacrificing accuracy Higher dropout value with additional image augmentation reduces overfitting further at every epoch without compromising accuracy The higher accuracy is evident in the number of correct predictions. This can also be seen in the confusion matrix and classification report. This has been verified visually too on the unlabelled test data set

##Conclusion

For custom daasets that do not have sufficiently large number of images, transfer learning is the preferred method of choice. By freezing the model's weights and then fine tuning with a fully connected layer gives us a very accurrate model that otherwise would not have been possible.