This project aims to use image classification to predict the artists of impressionist paintings. Namely,

> Camille Pisarro, Childe Hassam, Claude Monet, Edgar Degas, Henri Matisse John Singer-Sargent, Paul Cezanne, Paul Gauguin, Pierre-Auguste Renoir, and Vincent van Gogh

Importing Libraries and Mounting Dataset from Google Drive

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import glob as gb
import os
import torch
import torchvision
from torchvision import datasets, models, transforms
import torch.utils.data as data
from torch.utils.tensorboard import SummaryWriter
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import time, os, copy, argparse
import multiprocessing
from matplotlib import pyplot as plt
import tensorflow as tf
import cv2
from tensorflow import keras
from tensorflow.keras.models import Sequential, Model
from  matplotlib import pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline
```

```
from google.colab import drive
drive.mount("/content/drive", force_remount=True)
```

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

```
# import the libraries as shown below
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.inception_v3 import InceptionV3
#from keras.applications.vgg16 import VGG16
from tensorflow.keras.applications.inception_v3 import 
preprocess_input
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import 
ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import numpy as np
```

```
from glob import glob
#import matplotlib.pyplot as plt
```

```
!pip install pyyaml h5py
```

```
Looking in indexes: https://pypi.org/simple, https://us-
python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pyyaml in
/usr/local/lib/python3.8/dist-packages (6.0)
Requirement already satisfied: h5py in /usr/local/lib/python3.8/dist-
packages (3.1.0)
Requirement already satisfied: numpy>=1.17.5 in
/usr/local/lib/python3.8/dist-packages (from h5py) (1.21.6)
```

Preprocessing

We standardize the image size 224 x 224, which is normal for maching learning. Then, a path is created to the training and validation folders.

```
image_size = [224, 224]
BATCH_SIZE = 32
path = '/content/drive/MyDrive/impressionist'
training_path =
'/content/drive/MyDrive/impressionist/training/training'
validation_path =
'/content/drive/MyDrive/impressionist/validation/validation'
```

```
for folder in os.listdir(training_path):
    files = gb.glob(pathname= str(training_path+ '/'+ folder +
'/*.jpg'))
    print(f'For training data, found {len(files)} in folder {folder}')
```

```
For training data, found 399 in folder Monet
For training data, found 399 in folder Renoir
For training data, found 399 in folder Gauguin
For training data, found 399 in folder VanGogh
For training data, found 398 in folder Degas
For training data, found 399 in folder Matisse
For training data, found 398 in folder Pissarro
For training data, found 399 in folder Hassam
For training data, found 399 in folder Cezanne
For training data, found 399 in folder Sargent
```

```
for folder in os.listdir(validation_path):
    files = gb.glob(pathname= str(validation_path+ '/'+ folder +
'/*.jpg'))
    print(f'For validation data, found {len(files)} in folder
{folder}')
```

```
For validation data, found 99 in folder Matisse
For validation data, found 99 in folder Sargent
For validation data, found 99 in folder Gauguin
```

```
For validation data, found 99 in folder Monet
For validation data, found 99 in folder Degas
For validation data, found 99 in folder Cezanne
For validation data, found 99 in folder VanGogh
For validation data, found 99 in folder Renoir
For validation data, found 99 in folder Hassam
For validation data, found 99 in folder Pissarro
```

Data Exploration

For data exploration, we can see that there is an good number of training data for the program to look at and learn the different types of artist style. The model should be able to predict who made the piece from the given training data it has looked at.

```python
import random
import matplotlib.pyplot as plt
import matplotlib.image as mpimg

def view_random_image(target_dir, target_class):
    # We will view images from here
    target_folder = target_dir + target_class

    # Get a random image path
    random_image = random.sample(os.listdir(target_folder), 1)

    # read in the image and plot it using matplolib
    img = mpimg.imread(target_folder+'/'+random_image[0])
    plt.imshow(img)
    plt.title(target_class)
    plt.axis('off')
    print(f"Image shape {img.shape}")

    return img

class_names = ['Cezanne', 'Degas', 'Gauguin', 'Hassam', 'Matisse',
               'Monet', 'Pissarro', 'Renoir', 'Sargent', 'VanGogh']

plt.figure(figsize=(20,10))
for i in range(18):
    plt.subplot(6, 6, i+1)
    class_name = random.choice(class_names)
    img =
view_random_image(target_dir="/content/drive/MyDrive/impressionist/
training/training/", target_class=class_name)
```

```
Image shape (749, 991, 3)
Image shape (877, 1119, 3)
Image shape (986, 816, 3)
Image shape (600, 726, 3)
Image shape (525, 1105, 3)
Image shape (1600, 2024, 3)
```

```
Image shape (547, 901, 3)
Image shape (1316, 1776, 3)
Image shape (518, 640, 3)
Image shape (1600, 1081, 3)
Image shape (1070, 811, 3)
Image shape (1022, 1280, 3)
Image shape (1098, 758, 3)
Image shape (1182, 1600, 3)
Image shape (1018, 1037, 3)
Image shape (942, 1226, 3)
Image shape (1988, 1451, 3)
Image shape (1123, 872, 3)
```

Creating a Sequential Model

We used the Tensorflow notebook as reference over here:
https://www.tensorflow.org/tutorials/images/classification#a_basic_keras_model

```
#Importing images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import PIL


# Defining pre-processing transformations on raw images of training
data
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

# Defining pre-processing transformations on raw images of testing
data
test_datagen = ImageDataGenerator(rescale = 1./255)
```

We import pathlib to define a path since we used Google Drive. Then to test it, we used count and data.glob to find all pathnames that match this pattern, effectively finding the number of total training files.

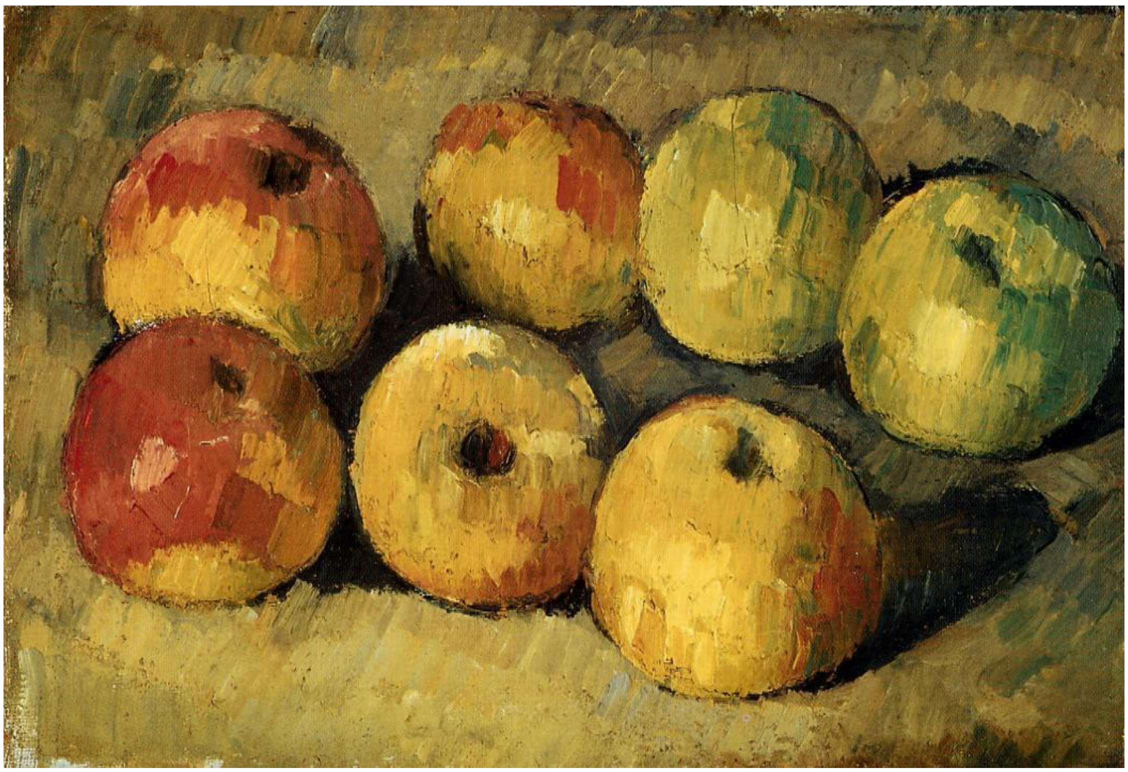
```
import pathlib
data = pathlib.Path('/content/drive/MyDrive/impressionist')
training =
```

```
pathlib.Path('/content/drive/MyDrive/impressionist/training/training')

validation =
pathlib.Path('/content/drive/MyDrive/impressionist/validation/validati
on')
count = len(list(training.glob('*/*.jpg')))
count
```

```
3988
```

For example, we can print a picture from any dataset.

```
Cezanne = list(training.glob('Cezanne/*'))
PIL.Image.open(str(Cezanne[0]))
```



```
training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/impressionis
t/training/training',
                                                 target_size = (224,
224),
                                                 batch_size = 32,
                                                 class_mode =
'categorical')
```

```
Found 3988 images belonging to 10 classes.
```

```
test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/impressionist
/validation/validation',
                                            target_size = (224, 224),
                                            batch_size = 32,
                                            class_mode =
'categorical')
```

```
Found 990 images belonging to 10 classes.
```

```
batch_size = 32
img_height = 224
img_width = 224
```

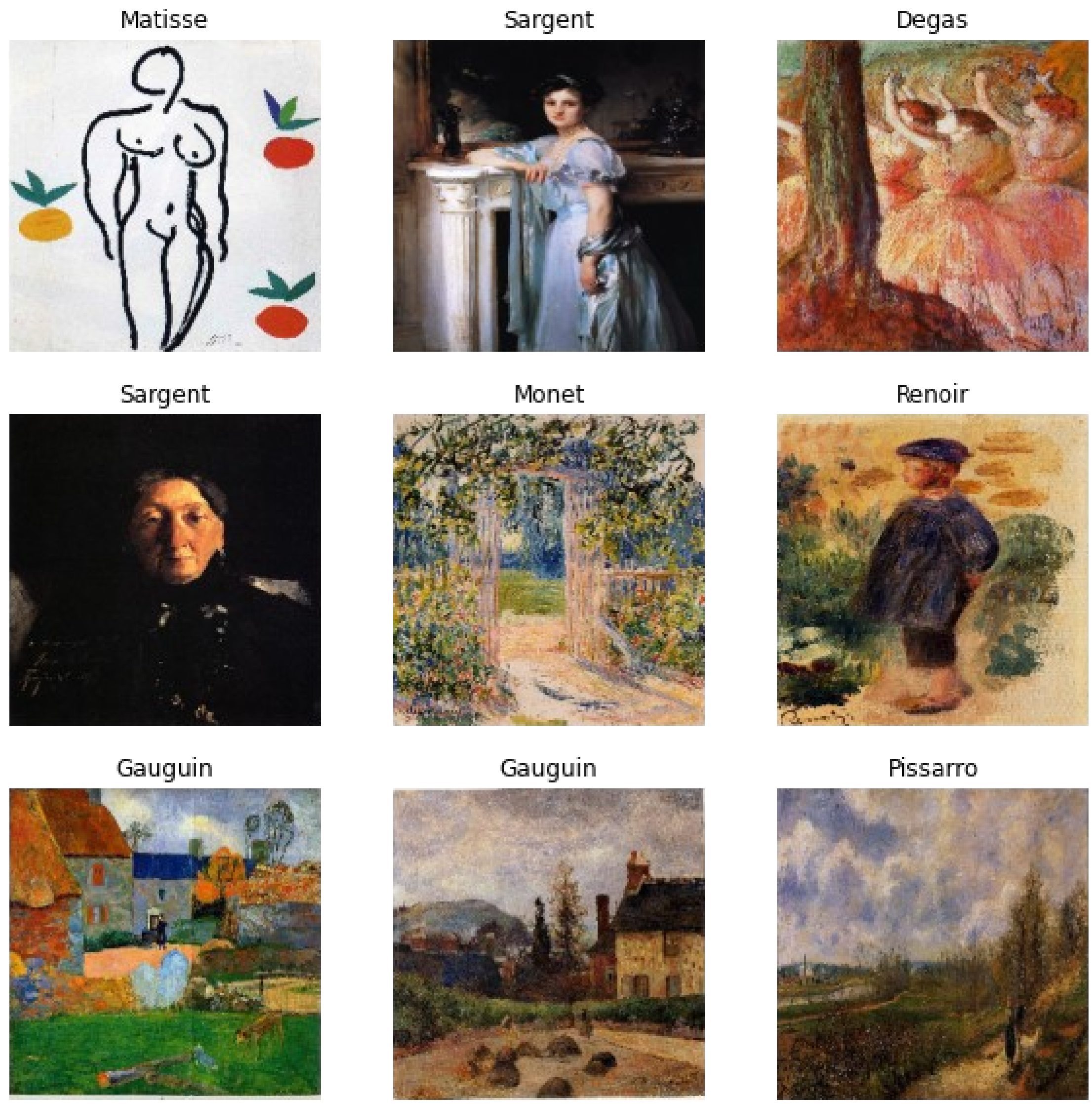
```
train_ds = tf.keras.utils.image_dataset_from_directory(
  training,
  validation_split=0.2,
  subset="training",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 3988 files belonging to 10 classes.
Using 3191 files for training.
```

```
val_ds = tf.keras.utils.image_dataset_from_directory(
  training,
  validation_split=0.2,
  subset="validation",
  seed=123,
  image_size=(img_height, img_width),
  batch_size=batch_size)
```

```
Found 3988 files belonging to 10 classes.
Using 797 files for validation.
```

```
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")
```




```
import matplotlib.pyplot as plt
import numpy as np
import PIL
import tensorflow as tf

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

normalization_layer = layers.Rescaling(1./255)
```

Optimizing Data

```
normalized_ds = train_ds.map(lambda x, y: (normalization_layer(x), y))
image_batch, labels_batch = next(iter(normalized_ds))
first_image = image_batch[0]
```

```
# Notice the pixel values are now in `[0,1]`.
print(np.min(first_image), np.max(first_image))
```

```
0.0 1.0
```

Creating a Sequential CNN model

Now we create the sequential mode with 3 layers of conv2D, and a flattened layer. The last line specifies that we have 10 classes.

```
model = Sequential([
  layers.Rescaling(1./255, input_shape=(img_height, img_width, 3)),
  layers.Conv2D(16, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(32, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, padding='same', activation='relu'),
  layers.MaxPooling2D(),
  layers.Flatten(),
  layers.Dense(128, activation='relu'),
  layers.Dense(10)
])
```

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

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

```
model.summary()
```

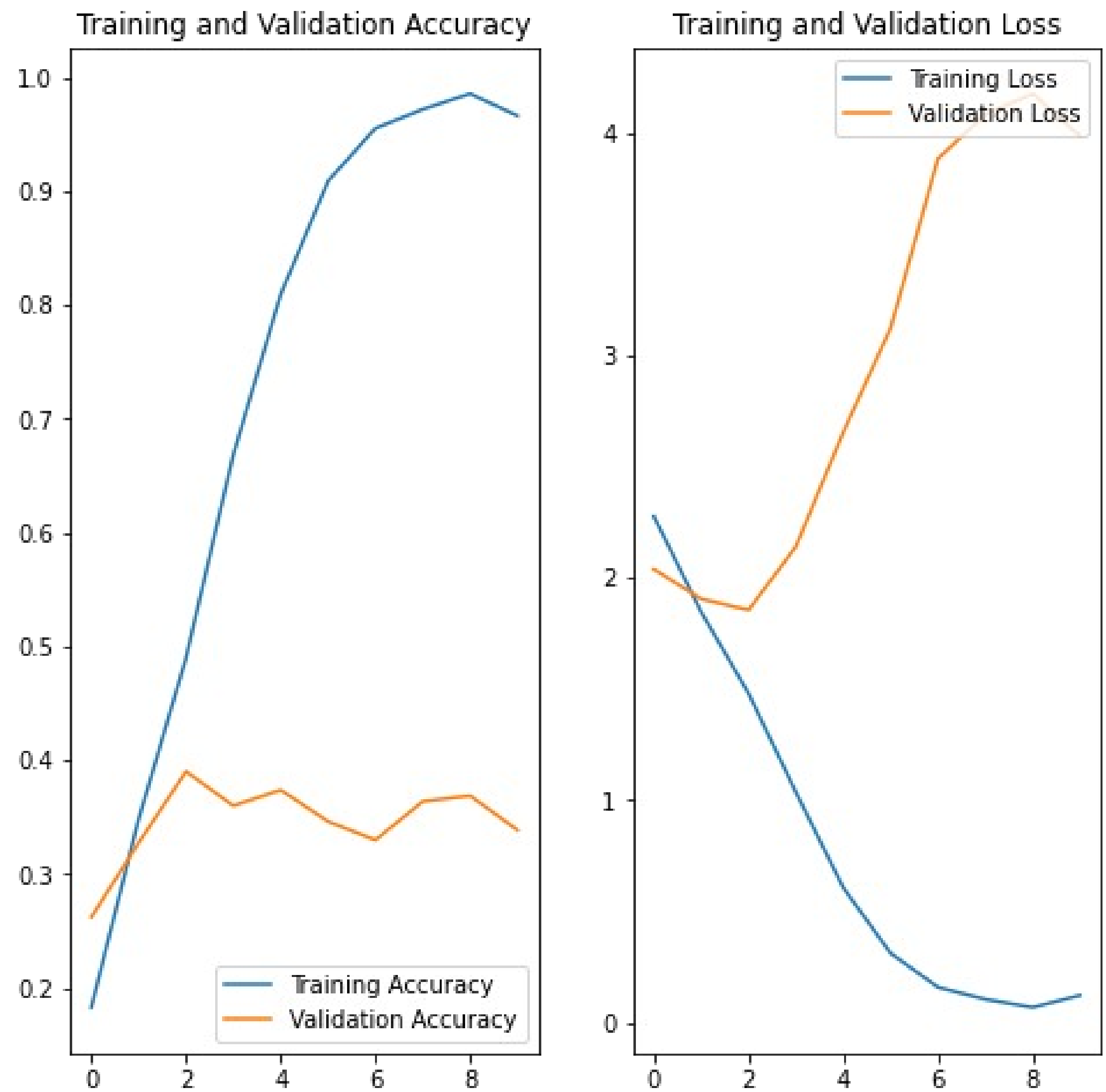
```
Model: "sequential_4"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 rescaling_5 (Rescaling)     (None, 224, 224, 3)       0         
                                                                 
 conv2d (Conv2D)             (None, 224, 224, 16)      448       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 112, 112, 16)     0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 112, 112, 32)      4640      
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 56, 56, 32)       0         
 2D)                                                             
                                                                 
 conv2d_2 (Conv2D)           (None, 56, 56, 64)        18496     
                                                                 
 max_pooling2d_2 (MaxPooling  (None, 28, 28, 64)       0         
 2D)                                                             
```

```
 flatten_2 (Flatten)         (None, 50176)             0

 dense_12 (Dense)            (None, 128)               6422656

 dense_13 (Dense)            (None, 10)                1290

=================================================================
Total params: 6,447,530
Trainable params: 6,447,530
Non-trainable params: 0
_________________________________________________________________
```

```
epochs=10
history = model.fit(
  train_ds,
  validation_data=val_ds,
  epochs=epochs
)
```

```
Epoch 1/10
100/100 [==============================] - 46s 407ms/step - loss:
2.2759 - accuracy: 0.1830 - val_loss: 2.0383 - val_accuracy: 0.2622
Epoch 2/10
100/100 [==============================] - 44s 413ms/step - loss:
1.8463 - accuracy: 0.3482 - val_loss: 1.9053 - val_accuracy: 0.3275
Epoch 3/10
100/100 [==============================] - 46s 439ms/step - loss:
1.4802 - accuracy: 0.4898 - val_loss: 1.8553 - val_accuracy: 0.3902
Epoch 4/10
100/100 [==============================] - 47s 427ms/step - loss:
1.0361 - accuracy: 0.6684 - val_loss: 2.1404 - val_accuracy: 0.3601
Epoch 5/10
100/100 [==============================] - 49s 466ms/step - loss:
0.6071 - accuracy: 0.8092 - val_loss: 2.6546 - val_accuracy: 0.3739
Epoch 6/10
100/100 [==============================] - 43s 406ms/step - loss:
0.3125 - accuracy: 0.9088 - val_loss: 3.1229 - val_accuracy: 0.3463
Epoch 7/10
100/100 [==============================] - 45s 426ms/step - loss:
0.1596 - accuracy: 0.9552 - val_loss: 3.8841 - val_accuracy: 0.3300
Epoch 8/10
100/100 [==============================] - 42s 399ms/step - loss:
0.1058 - accuracy: 0.9718 - val_loss: 4.0829 - val_accuracy: 0.3639
Epoch 9/10
100/100 [==============================] - 45s 420ms/step - loss:
0.0695 - accuracy: 0.9856 - val_loss: 4.1761 - val_accuracy: 0.3689
Epoch 10/10
100/100 [==============================] - 49s 464ms/step - loss:
0.1234 - accuracy: 0.9662 - val_loss: 3.9849 - val_accuracy: 0.3388
```

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

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

epochs_range = range(10)

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

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```




```
model.save('/content/drive/MyDrive/model_saving')
```

```
WARNING:absl:Found untraced functions such as
_jit_compiled_convolution_op, _jit_compiled_convolution_op,
_jit_compiled_convolution_op while saving (showing 3 of 3). These
functions will not be directly callable after loading.
```

Restoring the Model

```
# new_model = tf.keras.models.load_model('saved_model/my_model')

# Check its architecture
# new_model.summary()
```

Predicting on New Data

Let's make sure to make text classes for all 10 artists

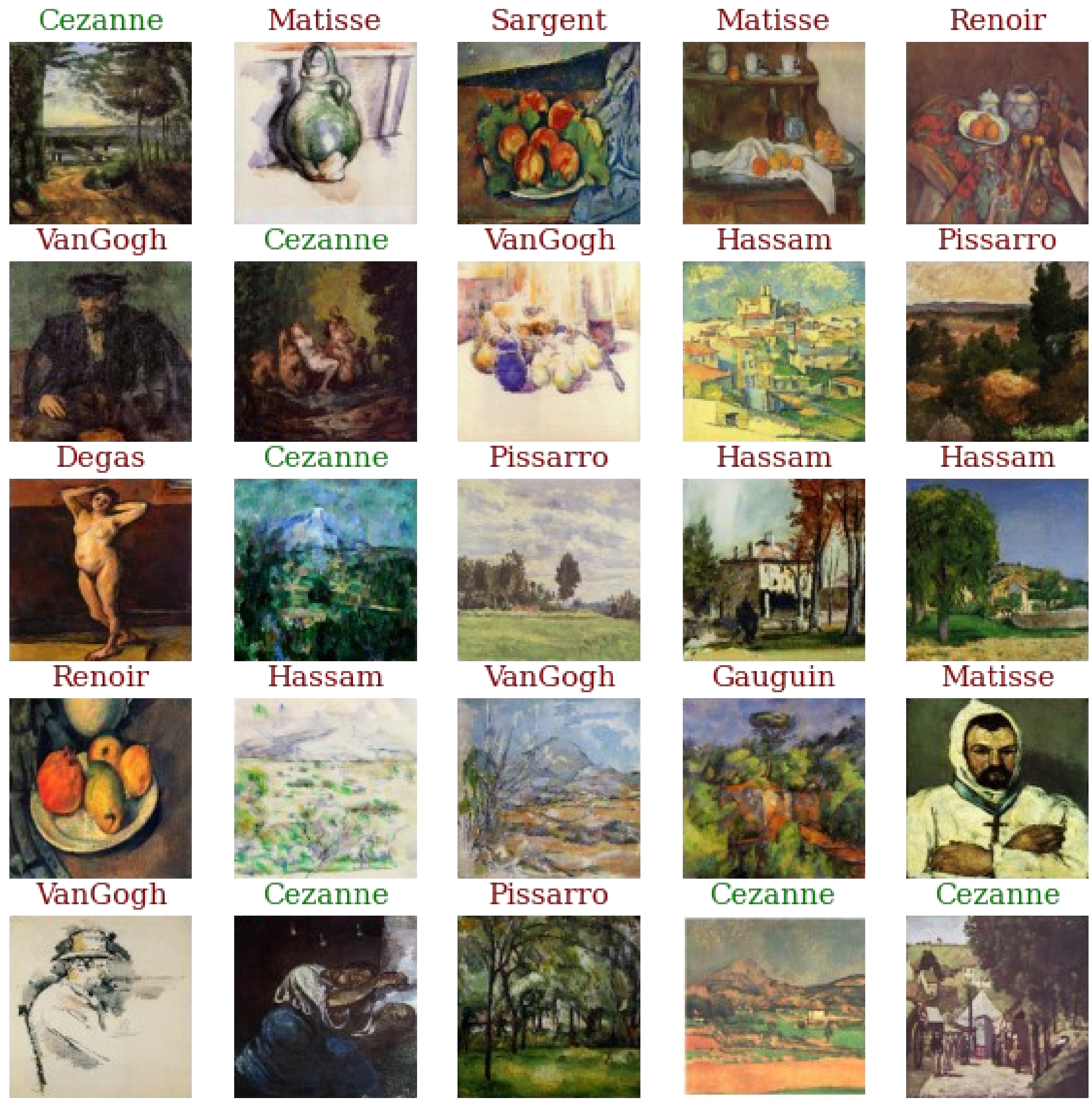
```
target_Cezanne= 'Cezanne'
target_Degas ='Degas'
target_Gauguin ='Gauguin'
target_Hassam ='Hassam'
target_Matisse='Matisse'
target_Monet = 'Monet'
target_Pissarro ='Pissarro'
target_Renoir ='Renoir'
target_Sargent ='Sargent'
target_VanGogh ='VanGogh'

target_dir =
"/content/drive/MyDrive/impressionist/validation/validation/"
```

Creating a function

```
def testPaintings(target_dir, target_class):
  images = []
  class_name = []
  scores = []


  target_folder = target_dir + target_class
  random_image = random.sample(os.listdir(target_folder), 25)

  for x in range(25):
    painting_url = target_folder + '/' + random_image[x]
    img = tf.keras.utils.load_img(
        painting_url, target_size=(img_height, img_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)
    score = tf.nn.softmax(predictions[0])
    # print("This image most likely belongs to {} with a {:.2f}
percent confidence.".format(class_names[np.argmax(score)], 100 *
np.max(score)))
    images.append(img)
    class_name.append(class_names[np.argmax(score)])
    scores.append(np.max(score))

  import matplotlib.pyplot as plt
  import matplotlib.pyplot as plt

  wrongPredict = {'family':'serif','color':'darkred','size':15}
  rightPredict = {'family':'serif','color':'green','size':15}

  plt.figure(figsize=(10, 10))
  for i in range(25):
```

```
    ax = plt.subplot(5, 5, i + 1)
    plt.imshow(images[i])
    if (class_name[i] == target_class):
      plt.title(class_name[i], fontdict = rightPredict)
    else:
      plt.title(class_name[i], fontdict = wrongPredict)
    plt.axis("off")

testPaintings(target_dir, target_Cezanne)
```

```
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
```



Testing on Degas's Paintings

```
testPaintings(target_dir, target_Degas)
```

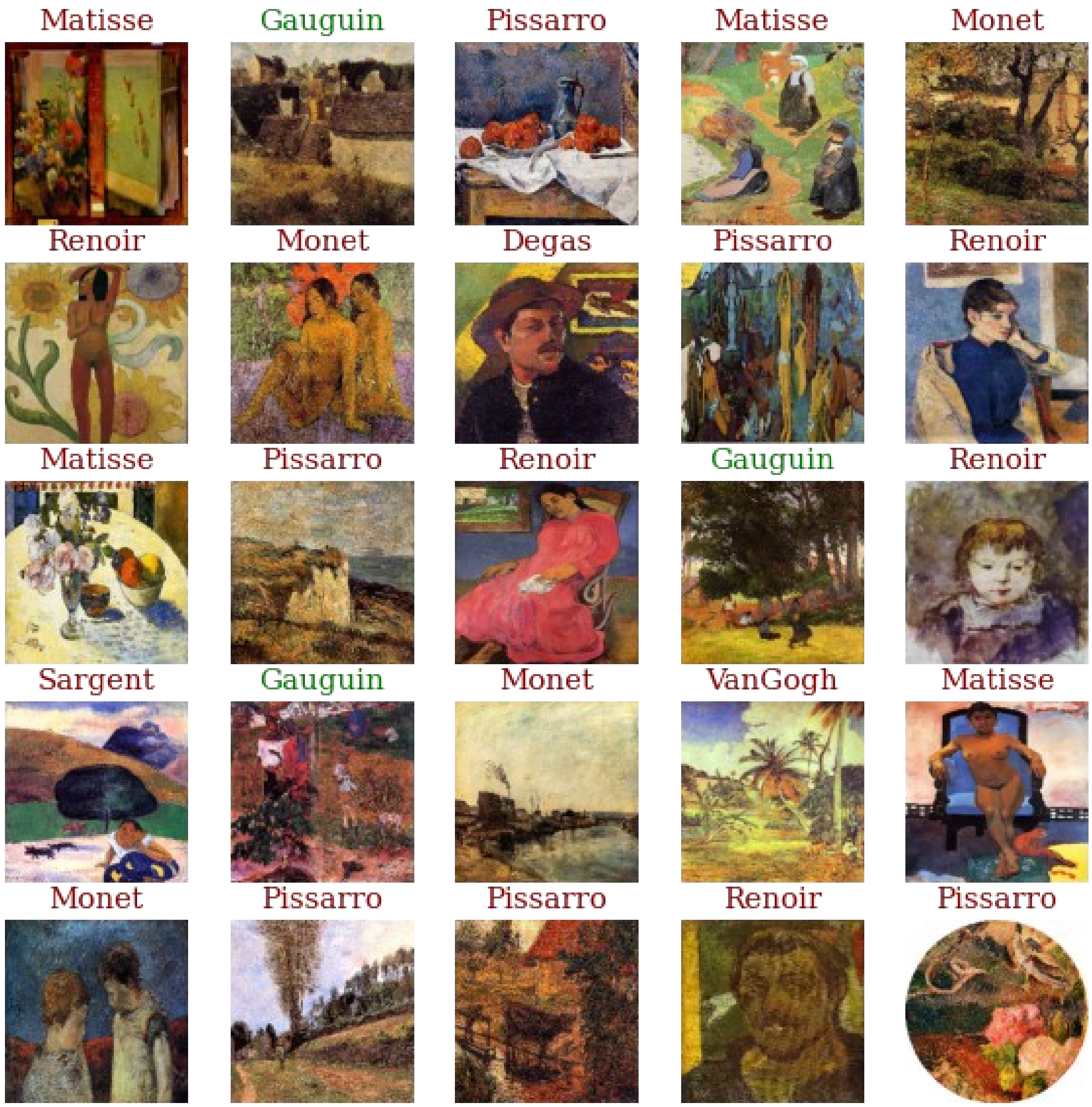
```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 20ms/step
```

```
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 28ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
```

Testing on Gauguin Paintings

```
testPaintings(target_dir, target_Gauguin)
```

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
```

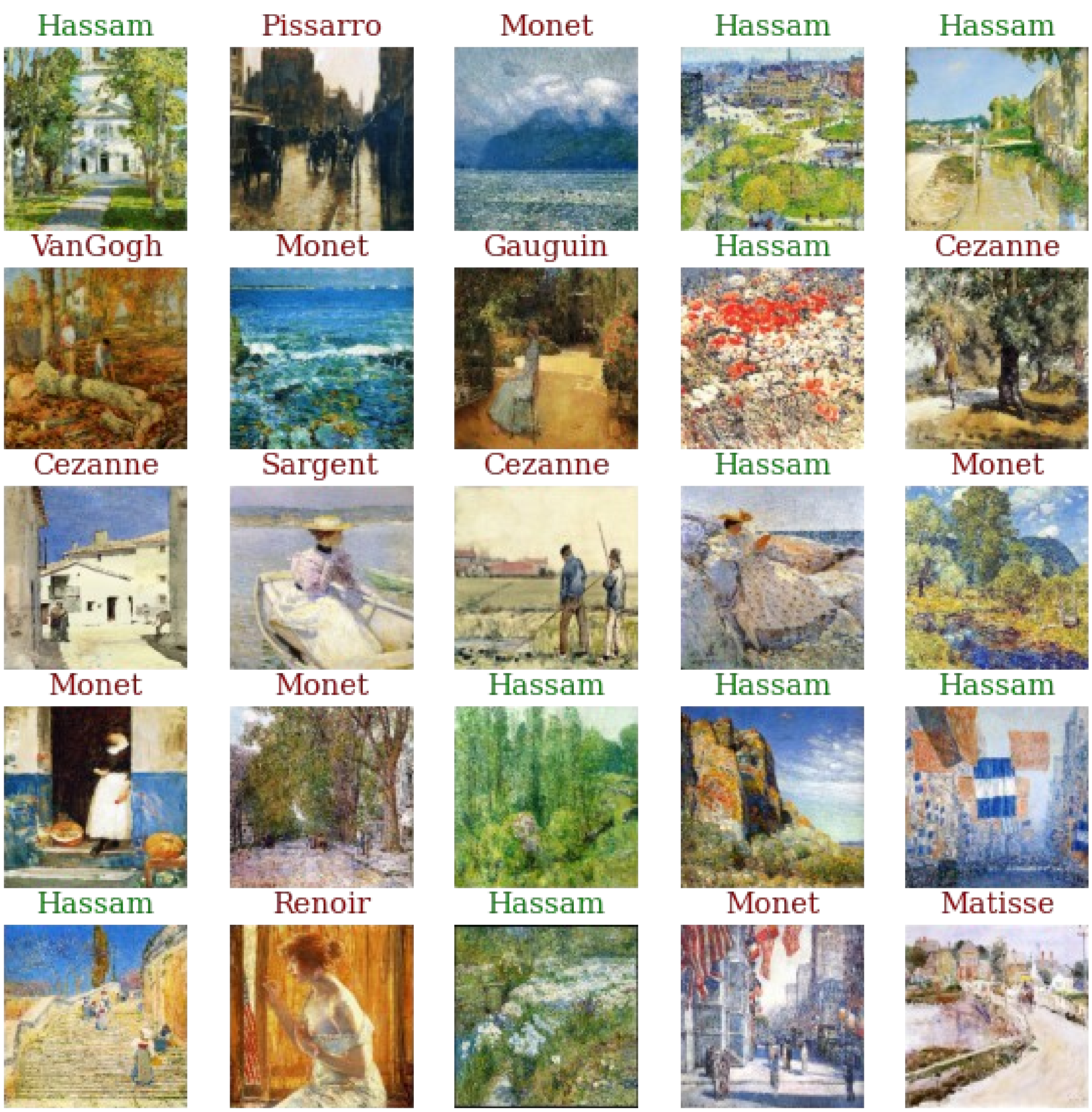


Testing on Hassam Paintings

```
testPaintings(target_dir, target_Hassam)
```

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 23ms/step
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
```



Testing Monet's Paintings

```
testPaintings(target_dir, target_VanGogh)
```

```
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 26ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
```


```
testPaintings(target_dir, target_Matisse)
```

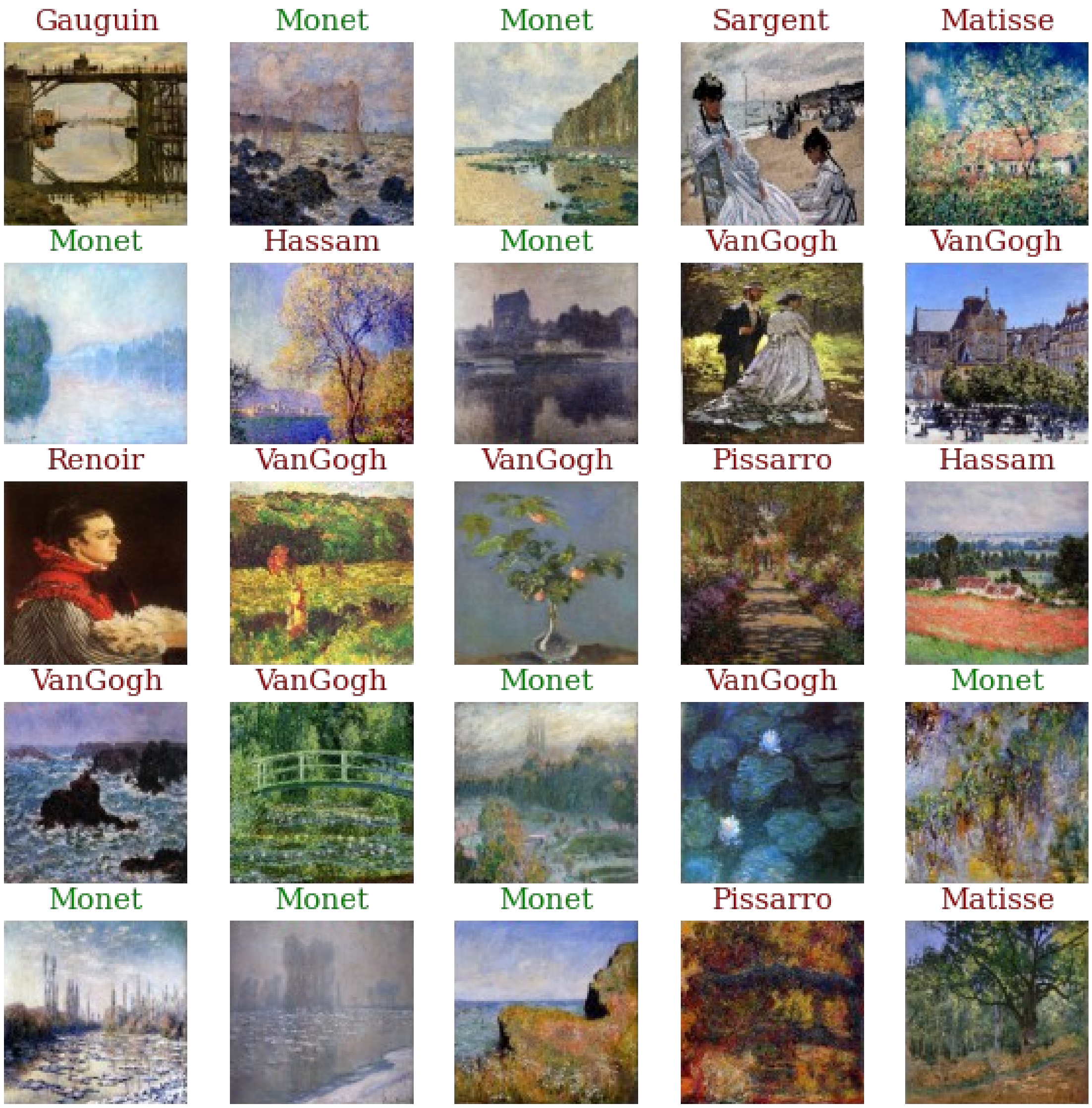
```
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
```

```
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
```

Predicting Pissarro's Paintings

```
testPaintings(target_dir, target_Monet)
```

```
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
```

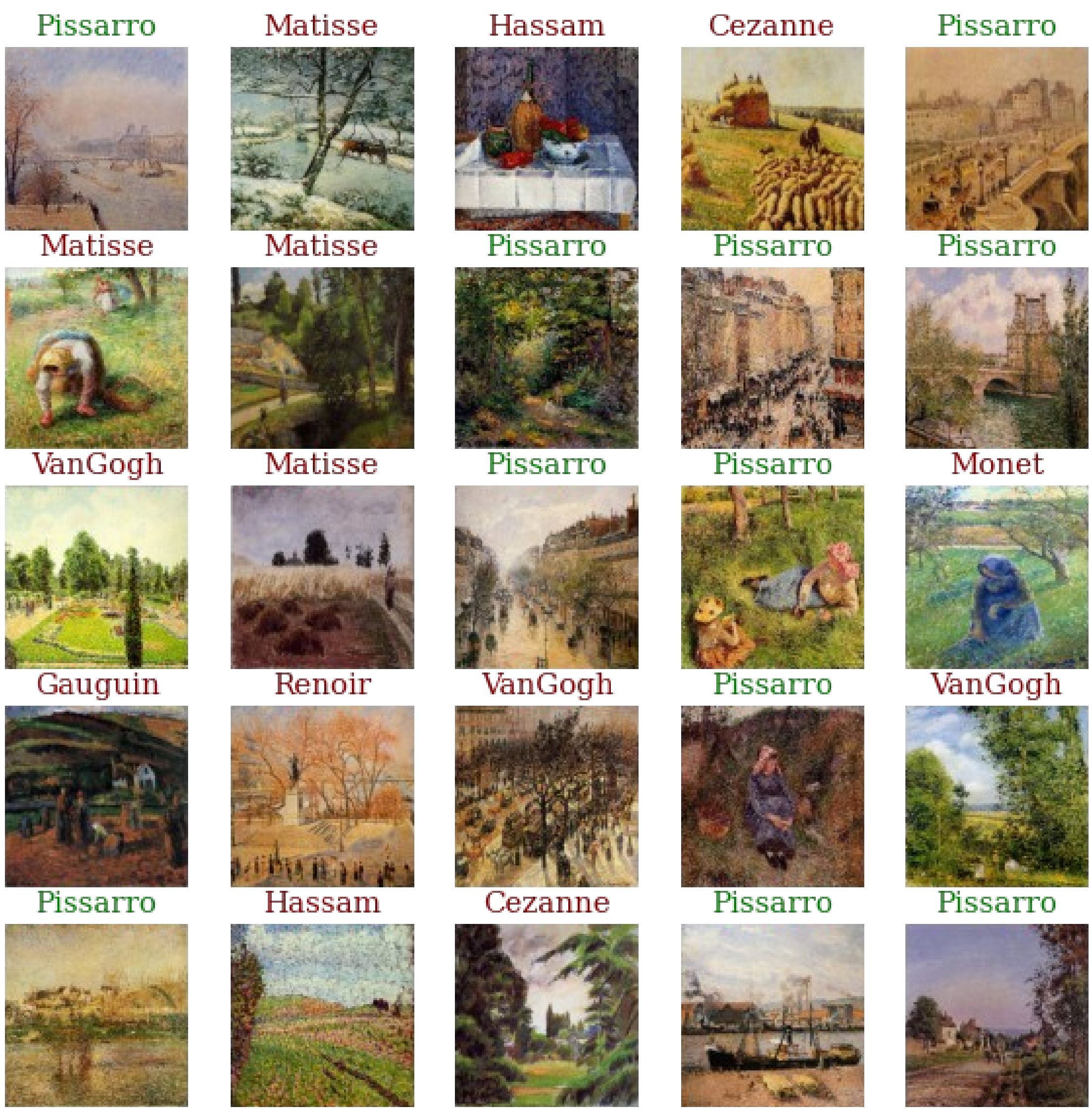


Predicting Pissarro's Paintings

```
testPaintings(target_dir, target_Pissarro)
```

```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 21ms/step
```

```
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
```




```
testPaintings(target_dir, target_Renoir)
```

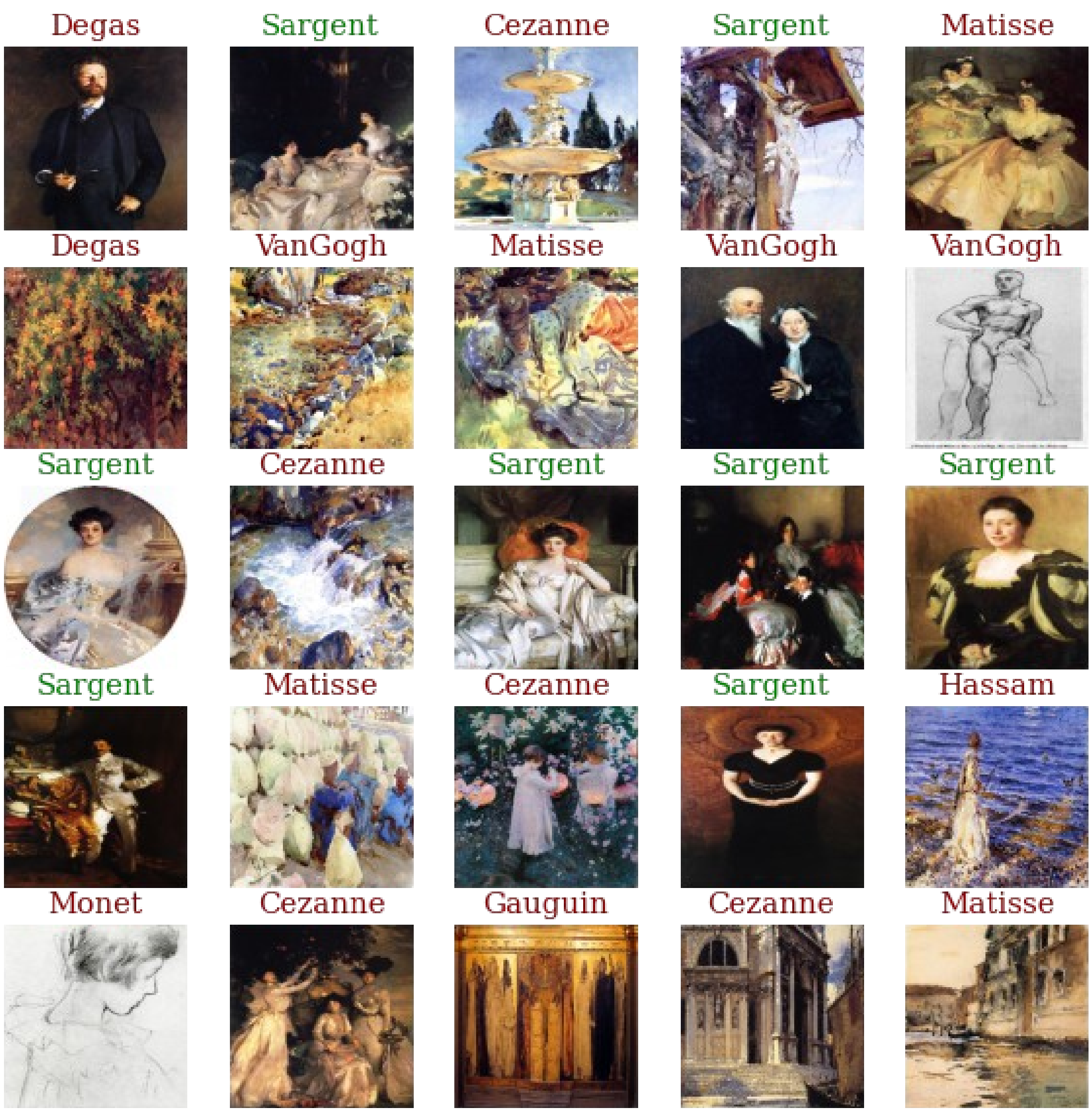
```
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 22ms/step
```

```
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 31ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 22ms/step
```


```
testPaintings(target_dir, target_Sargent)
```

```
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 15ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 16ms/step
```

```
1/1 [==============================] - 0s 20ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 17ms/step
1/1 [==============================] - 0s 16ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 16ms/step
```



RNN Architecture Model

```
max_features = 10000
maxlen = 500
batch_size = 32
```

```
from tensorflow.keras.models import Sequential


modelRNN = keras.Sequential([Input(shape=(224, 224, 3)),
modelRNN.add(layers.Embedding(max_features, 32))
modelRNN.add(layers.SimpleRNN(32))
modelRNN.add(layers.Dense(1, activation='sigmoid'))

modelRNN.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 224, 224, 16)      448

 max_pooling2d_3 (MaxPooling  (None, 112, 112, 16)     0
 2D)

 conv2d_4 (Conv2D)           (None, 112, 112, 32)      4640

 max_pooling2d_4 (MaxPooling  (None, 56, 56, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 56, 56, 64)        18496

 max_pooling2d_5 (MaxPooling  (None, 28, 28, 64)       0
 2D)

 flatten_1 (Flatten)         (None, 50176)             0

 dense_2 (Dense)             (None, 128)               6422656

 dense_3 (Dense)             (None, 10)                1290

 embedding (Embedding)       (None, 10, 32)            320000

 simple_rnn (SimpleRNN)      (None, 32)                2080

 dense_4 (Dense)             (None, 1)                 33

=================================================================
Total params: 6,769,643
Trainable params: 6,769,643
Non-trainable params: 0
_________________________________________________________________
```

```
modelRNN.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['accuracy'])
```

```
historyRNN = modelRNN.fit(
  training_set,
  validation_data=test_set,
  epochs= 10,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
Epoch 1/10

WARNING:tensorflow:Gradients do not exist for variables
['conv2d_3/kernel:0', 'conv2d_3/bias:0', 'conv2d_4/kernel:0',
'conv2d_4/bias:0', 'conv2d_5/kernel:0', 'conv2d_5/bias:0',
'dense_2/kernel:0', 'dense_2/bias:0', 'dense_3/kernel:0',
'dense_3/bias:0'] when minimizing the loss. If you're using
`model.compile()`, did you forget to provide a `loss`argument?
WARNING:tensorflow:Gradients do not exist for variables
['conv2d_3/kernel:0', 'conv2d_3/bias:0', 'conv2d_4/kernel:0',
'conv2d_4/bias:0', 'conv2d_5/kernel:0', 'conv2d_5/bias:0',
'dense_2/kernel:0', 'dense_2/bias:0', 'dense_3/kernel:0',
'dense_3/bias:0'] when minimizing the loss. If you're using
`model.compile()`, did you forget to provide a `loss`argument?

125/125 [==============================] - 1531s 12s/step - loss:
0.3339 - accuracy: 0.9000 - val_loss: 0.3252 - val_accuracy: 0.9000
Epoch 2/10
123/125 [============================>.] - ETA: 3s - loss: 0.3252 -
accuracy: 0.9000
```

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

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

epochs_range = range(10)

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

plt.subplot(1, 2, 2)
plt.plot(epochs_range, loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.legend(loc='upper right')
plt.title('Training and Validation Loss')
plt.show()
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-1-0ffab28fb573> in <module>
----> 1 acc = history.history['accuracy']
      2 val_acc = history.history['val_accuracy']
      3
      4 loss = history.history['loss']
      5 val_loss = history.history['val_loss']

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

Using Inception v3 Architecture

Using a pretrained model was trained on a large dataset, and can save time and energy. This model can serve as a general model for the visual world, and can be used as learned feature maps without starting from sctach for models we decide to train on later.

Instructions on how to apply Inception v3 and preprocessing were referenced usign this tutorial: https://www.youtube.com/watch?v=chQNuV9B-Rw&t=837s

Here we import the Inception V3 Library, adding preprocessing layer to the front.

```
inception = InceptionV3(input_shape=image_size + [3], weights='imagenet', include_top=False)
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/inception_v3/inception_v3_weights_tf_dim_ordering_tf_kernels_notop.h5
87910968/87910968 [==============================] - 1s 0us/step
```

```
# don't train existing weights
for layer in inception.layers:
    layer.trainable = False
```

```
x = Flatten()(inception.output)
```

Need 10, since we have 10 classes.

```
prediction = Dense(10, activation='softmax')(x)
```

```
# create a model object
model = Model(inputs=inception.input, outputs=prediction)
```

```
# view the structure of the model
model.summary()
```

```
Model: "model"
__________________________________________________________________________________________________
 Layer (type)                    Output Shape         Param #
```

```
Connected to
======================================================================
============================
 input_1 (InputLayer)           [(None, 224, 224, 3  0           []

                                )]



 conv2d_6 (Conv2D)              (None, 111, 111, 32  864
['input_1[0][0]']
                                )



 batch_normalization (BatchNorm  (None, 111, 111, 32  96
['conv2d_6[0][0]']
 alization)                     )



 activation (Activation)        (None, 111, 111, 32  0
['batch_normalization[0][0]']
                                )



 conv2d_7 (Conv2D)              (None, 109, 109, 32  9216
['activation[0][0]']
                                )



 batch_normalization_1 (BatchNo  (None, 109, 109, 32  96
['conv2d_7[0][0]']
 rmalization)                   )



 activation_1 (Activation)      (None, 109, 109, 32  0
['batch_normalization_1[0][0]']
                                )



 conv2d_8 (Conv2D)              (None, 109, 109, 64  18432
['activation_1[0][0]']
                                )
```

```
 batch_normalization_2 (BatchNo  (None, 109, 109, 64  192
['conv2d_8[0][0]']
 rmalization)                   )



 activation_2 (Activation)      (None, 109, 109, 64  0
['batch_normalization_2[0][0]']
                                )



 max_pooling2d_6 (MaxPooling2D)  (None, 54, 54, 64)  0
['activation_2[0][0]']


 conv2d_9 (Conv2D)              (None, 54, 54, 80)   5120
['max_pooling2d_6[0][0]']


 batch_normalization_3 (BatchNo  (None, 54, 54, 80)  240
['conv2d_9[0][0]']
 rmalization)



 activation_3 (Activation)      (None, 54, 54, 80)   0
['batch_normalization_3[0][0]']


 conv2d_10 (Conv2D)             (None, 52, 52, 192)  138240
['activation_3[0][0]']


 batch_normalization_4 (BatchNo  (None, 52, 52, 192)  576
['conv2d_10[0][0]']
 rmalization)



 activation_4 (Activation)      (None, 52, 52, 192)  0
['batch_normalization_4[0][0]']


 max_pooling2d_7 (MaxPooling2D)  (None, 25, 25, 192)  0
```

```
['activation_4[0][0]']


 conv2d_14 (Conv2D)             (None, 25, 25, 64)   12288
['max_pooling2d_7[0][0]']


 batch_normalization_8 (BatchNo  (None, 25, 25, 64)  192
['conv2d_14[0][0]']
 rmalization)



 activation_8 (Activation)      (None, 25, 25, 64)   0
['batch_normalization_8[0][0]']


 conv2d_12 (Conv2D)             (None, 25, 25, 48)   9216
['max_pooling2d_7[0][0]']


 conv2d_15 (Conv2D)             (None, 25, 25, 96)   55296
['activation_8[0][0]']


 batch_normalization_6 (BatchNo  (None, 25, 25, 48)  144
['conv2d_12[0][0]']
 rmalization)



 batch_normalization_9 (BatchNo  (None, 25, 25, 96)  288
['conv2d_15[0][0]']
 rmalization)



 activation_6 (Activation)      (None, 25, 25, 48)   0
['batch_normalization_6[0][0]']


 activation_9 (Activation)      (None, 25, 25, 96)   0
['batch_normalization_9[0][0]']


 average_pooling2d (AveragePool  (None, 25, 25, 192)  0
['max_pooling2d_7[0][0]']
 ing2D)
```

```
 conv2d_11 (Conv2D)             (None, 25, 25, 64)   12288
['max_pooling2d_7[0][0]']


 conv2d_13 (Conv2D)             (None, 25, 25, 64)   76800
['activation_6[0][0]']


 conv2d_16 (Conv2D)             (None, 25, 25, 96)   82944
['activation_9[0][0]']


 conv2d_17 (Conv2D)             (None, 25, 25, 32)   6144
['average_pooling2d[0][0]']


 batch_normalization_5 (BatchNo  (None, 25, 25, 64)  192
['conv2d_11[0][0]']
 rmalization)



 batch_normalization_7 (BatchNo  (None, 25, 25, 64)  192
['conv2d_13[0][0]']
 rmalization)



 batch_normalization_10 (BatchN  (None, 25, 25, 96)  288
['conv2d_16[0][0]']
 ormalization)



 batch_normalization_11 (BatchN  (None, 25, 25, 32)  96
['conv2d_17[0][0]']
 ormalization)



 activation_5 (Activation)      (None, 25, 25, 64)   0
['batch_normalization_5[0][0]']


 activation_7 (Activation)      (None, 25, 25, 64)   0
```

```
['batch_normalization_7[0][0]']


 activation_10 (Activation)     (None, 25, 25, 96)   0
['batch_normalization_10[0][0]']


 activation_11 (Activation)     (None, 25, 25, 32)   0
['batch_normalization_11[0][0]']


 mixed0 (Concatenate)           (None, 25, 25, 256)  0
['activation_5[0][0]',

'activation_7[0][0]',

'activation_10[0][0]',

'activation_11[0][0]']


 conv2d_21 (Conv2D)             (None, 25, 25, 64)   16384
['mixed0[0][0]']


 batch_normalization_15 (BatchN  (None, 25, 25, 64)  192
['conv2d_21[0][0]']
 ormalization)



 activation_15 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_15[0][0]']


 conv2d_19 (Conv2D)             (None, 25, 25, 48)   12288
['mixed0[0][0]']


 conv2d_22 (Conv2D)             (None, 25, 25, 96)   55296
['activation_15[0][0]']


 batch_normalization_13 (BatchN  (None, 25, 25, 48)  144
['conv2d_19[0][0]']
 ormalization)
```

```
 batch_normalization_16 (BatchN  (None, 25, 25, 96)  288
['conv2d_22[0][0]']
 ormalization)


 activation_13 (Activation)     (None, 25, 25, 48)   0
['batch_normalization_13[0][0]']


 activation_16 (Activation)     (None, 25, 25, 96)   0
['batch_normalization_16[0][0]']


 average_pooling2d_1 (AveragePo  (None, 25, 25, 256)  0
['mixed0[0][0]']
 oling2D)


 conv2d_18 (Conv2D)             (None, 25, 25, 64)   16384
['mixed0[0][0]']


 conv2d_20 (Conv2D)             (None, 25, 25, 64)   76800
['activation_13[0][0]']


 conv2d_23 (Conv2D)             (None, 25, 25, 96)   82944
['activation_16[0][0]']


 conv2d_24 (Conv2D)             (None, 25, 25, 64)   16384
['average_pooling2d_1[0][0]']


 batch_normalization_12 (BatchN  (None, 25, 25, 64)  192
['conv2d_18[0][0]']
 ormalization)


 batch_normalization_14 (BatchN  (None, 25, 25, 64)  192
['conv2d_20[0][0]']
 ormalization)
```

```
 batch_normalization_17 (BatchN  (None, 25, 25, 96)  288
['conv2d_23[0][0]']
 ormalization)


 batch_normalization_18 (BatchN  (None, 25, 25, 64)  192
['conv2d_24[0][0]']
 ormalization)


 activation_12 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_12[0][0]']

 activation_14 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_14[0][0]']

 activation_17 (Activation)     (None, 25, 25, 96)   0
['batch_normalization_17[0][0]']

 activation_18 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_18[0][0]']

 mixed1 (Concatenate)           (None, 25, 25, 288)  0
['activation_12[0][0]',

'activation_14[0][0]',

'activation_17[0][0]',

'activation_18[0][0]']

 conv2d_28 (Conv2D)             (None, 25, 25, 64)   18432
['mixed1[0][0]']

 batch_normalization_22 (BatchN  (None, 25, 25, 64)  192
['conv2d_28[0][0]']
 ormalization)
```

```
 activation_22 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_22[0][0]']


 conv2d_26 (Conv2D)             (None, 25, 25, 48)   13824
['mixed1[0][0]']


 conv2d_29 (Conv2D)             (None, 25, 25, 96)   55296
['activation_22[0][0]']


 batch_normalization_20 (BatchN  (None, 25, 25, 48)  144
['conv2d_26[0][0]']
 ormalization)



 batch_normalization_23 (BatchN  (None, 25, 25, 96)  288
['conv2d_29[0][0]']
 ormalization)



 activation_20 (Activation)     (None, 25, 25, 48)   0
['batch_normalization_20[0][0]']


 activation_23 (Activation)     (None, 25, 25, 96)   0
['batch_normalization_23[0][0]']


 average_pooling2d_2 (AveragePo  (None, 25, 25, 288)  0
['mixed1[0][0]']
 oling2D)



 conv2d_25 (Conv2D)             (None, 25, 25, 64)   18432
['mixed1[0][0]']


 conv2d_27 (Conv2D)             (None, 25, 25, 64)   76800
['activation_20[0][0]']


 conv2d_30 (Conv2D)             (None, 25, 25, 96)   82944
```

```
['activation_23[0][0]']


 conv2d_31 (Conv2D)             (None, 25, 25, 64)   18432
['average_pooling2d_2[0][0]']


 batch_normalization_19 (BatchN  (None, 25, 25, 64)  192
['conv2d_25[0][0]']
 ormalization)



 batch_normalization_21 (BatchN  (None, 25, 25, 64)  192
['conv2d_27[0][0]']
 ormalization)



 batch_normalization_24 (BatchN  (None, 25, 25, 96)  288
['conv2d_30[0][0]']
 ormalization)



 batch_normalization_25 (BatchN  (None, 25, 25, 64)  192
['conv2d_31[0][0]']
 ormalization)



 activation_19 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_19[0][0]']


 activation_21 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_21[0][0]']


 activation_24 (Activation)     (None, 25, 25, 96)   0
['batch_normalization_24[0][0]']


 activation_25 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_25[0][0]']


 mixed2 (Concatenate)           (None, 25, 25, 288)  0
```

```
['activation_19[0][0]',

'activation_21[0][0]',

'activation_24[0][0]',

'activation_25[0][0]']


 conv2d_33 (Conv2D)             (None, 25, 25, 64)   18432
['mixed2[0][0]']


 batch_normalization_27 (BatchN  (None, 25, 25, 64)  192
['conv2d_33[0][0]']
 ormalization)



 activation_27 (Activation)     (None, 25, 25, 64)   0
['batch_normalization_27[0][0]']


 conv2d_34 (Conv2D)             (None, 25, 25, 96)   55296
['activation_27[0][0]']


 batch_normalization_28 (BatchN  (None, 25, 25, 96)  288
['conv2d_34[0][0]']
 ormalization)



 activation_28 (Activation)     (None, 25, 25, 96)   0
['batch_normalization_28[0][0]']


 conv2d_32 (Conv2D)             (None, 12, 12, 384)  995328
['mixed2[0][0]']


 conv2d_35 (Conv2D)             (None, 12, 12, 96)   82944
['activation_28[0][0]']


 batch_normalization_26 (BatchN  (None, 12, 12, 384)  1152
['conv2d_32[0][0]']
 ormalization)
```

```
 batch_normalization_29 (BatchN  (None, 12, 12, 96)  288
['conv2d_35[0][0]']
 ormalization)



 activation_26 (Activation)     (None, 12, 12, 384)  0
['batch_normalization_26[0][0]']


 activation_29 (Activation)     (None, 12, 12, 96)   0
['batch_normalization_29[0][0]']


 max_pooling2d_8 (MaxPooling2D)  (None, 12, 12, 288)  0
['mixed2[0][0]']


 mixed3 (Concatenate)           (None, 12, 12, 768)  0
['activation_26[0][0]',

'activation_29[0][0]',

'max_pooling2d_8[0][0]']


 conv2d_40 (Conv2D)             (None, 12, 12, 128)  98304
['mixed3[0][0]']


 batch_normalization_34 (BatchN  (None, 12, 12, 128)  384
['conv2d_40[0][0]']
 ormalization)



 activation_34 (Activation)     (None, 12, 12, 128)  0
['batch_normalization_34[0][0]']


 conv2d_41 (Conv2D)             (None, 12, 12, 128)  114688
['activation_34[0][0]']


 batch_normalization_35 (BatchN  (None, 12, 12, 128)  384
```

```
['conv2d_41[0][0]']
 ormalization)



 activation_35 (Activation)     (None, 12, 12, 128)  0
['batch_normalization_35[0][0]']


 conv2d_37 (Conv2D)             (None, 12, 12, 128)  98304
['mixed3[0][0]']


 conv2d_42 (Conv2D)             (None, 12, 12, 128)  114688
['activation_35[0][0]']


 batch_normalization_31 (BatchN  (None, 12, 12, 128)  384
['conv2d_37[0][0]']
 ormalization)



 batch_normalization_36 (BatchN  (None, 12, 12, 128)  384
['conv2d_42[0][0]']
 ormalization)



 activation_31 (Activation)     (None, 12, 12, 128)  0
['batch_normalization_31[0][0]']


 activation_36 (Activation)     (None, 12, 12, 128)  0
['batch_normalization_36[0][0]']


 conv2d_38 (Conv2D)             (None, 12, 12, 128)  114688
['activation_31[0][0]']


 conv2d_43 (Conv2D)             (None, 12, 12, 128)  114688
['activation_36[0][0]']


 batch_normalization_32 (BatchN  (None, 12, 12, 128)  384
['conv2d_38[0][0]']
 ormalization)
```

```
 batch_normalization_37 (BatchN  (None, 12, 12, 128)  384
['conv2d_43[0][0]']
 ormalization)


 activation_32 (Activation)     (None, 12, 12, 128)  0
['batch_normalization_32[0][0]']


 activation_37 (Activation)     (None, 12, 12, 128)  0
['batch_normalization_37[0][0]']


 average_pooling2d_3 (AveragePo  (None, 12, 12, 768)  0
['mixed3[0][0]']
 oling2D)


 conv2d_36 (Conv2D)             (None, 12, 12, 192)  147456
['mixed3[0][0]']


 conv2d_39 (Conv2D)             (None, 12, 12, 192)  172032
['activation_32[0][0]']


 conv2d_44 (Conv2D)             (None, 12, 12, 192)  172032
['activation_37[0][0]']


 conv2d_45 (Conv2D)             (None, 12, 12, 192)  147456
['average_pooling2d_3[0][0]']


 batch_normalization_30 (BatchN  (None, 12, 12, 192)  576
['conv2d_36[0][0]']
 ormalization)


 batch_normalization_33 (BatchN  (None, 12, 12, 192)  576
['conv2d_39[0][0]']
 ormalization)
```

```
 batch_normalization_38 (BatchN  (None, 12, 12, 192)  576
['conv2d_44[0][0]']
 ormalization)


 batch_normalization_39 (BatchN  (None, 12, 12, 192)  576
['conv2d_45[0][0]']
 ormalization)


 activation_30 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_30[0][0]']


 activation_33 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_33[0][0]']


 activation_38 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_38[0][0]']


 activation_39 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_39[0][0]']


 mixed4 (Concatenate)           (None, 12, 12, 768)  0
['activation_30[0][0]',

'activation_33[0][0]',

'activation_38[0][0]',

'activation_39[0][0]']


 conv2d_50 (Conv2D)             (None, 12, 12, 160)  122880
['mixed4[0][0]']


 batch_normalization_44 (BatchN  (None, 12, 12, 160)  480
['conv2d_50[0][0]']
 ormalization)
```

```
 activation_44 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_44[0][0]']


 conv2d_51 (Conv2D)             (None, 12, 12, 160)  179200
['activation_44[0][0]']


 batch_normalization_45 (BatchN  (None, 12, 12, 160)  480
['conv2d_51[0][0]']
 ormalization)



 activation_45 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_45[0][0]']


 conv2d_47 (Conv2D)             (None, 12, 12, 160)  122880
['mixed4[0][0]']


 conv2d_52 (Conv2D)             (None, 12, 12, 160)  179200
['activation_45[0][0]']


 batch_normalization_41 (BatchN  (None, 12, 12, 160)  480
['conv2d_47[0][0]']
 ormalization)



 batch_normalization_46 (BatchN  (None, 12, 12, 160)  480
['conv2d_52[0][0]']
 ormalization)



 activation_41 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_41[0][0]']


 activation_46 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_46[0][0]']
```

```
 conv2d_48 (Conv2D)             (None, 12, 12, 160)  179200
['activation_41[0][0]']


 conv2d_53 (Conv2D)             (None, 12, 12, 160)  179200
['activation_46[0][0]']


 batch_normalization_42 (BatchN  (None, 12, 12, 160)  480
['conv2d_48[0][0]']
 ormalization)



 batch_normalization_47 (BatchN  (None, 12, 12, 160)  480
['conv2d_53[0][0]']
 ormalization)



 activation_42 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_42[0][0]']


 activation_47 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_47[0][0]']


 average_pooling2d_4 (AveragePo  (None, 12, 12, 768)  0
['mixed4[0][0]']
 oling2D)



 conv2d_46 (Conv2D)             (None, 12, 12, 192)  147456
['mixed4[0][0]']


 conv2d_49 (Conv2D)             (None, 12, 12, 192)  215040
['activation_42[0][0]']


 conv2d_54 (Conv2D)             (None, 12, 12, 192)  215040
['activation_47[0][0]']


 conv2d_55 (Conv2D)             (None, 12, 12, 192)  147456
```

```
['average_pooling2d_4[0][0]']


 batch_normalization_40 (BatchN  (None, 12, 12, 192)  576
['conv2d_46[0][0]']
 ormalization)


 batch_normalization_43 (BatchN  (None, 12, 12, 192)  576
['conv2d_49[0][0]']
 ormalization)


 batch_normalization_48 (BatchN  (None, 12, 12, 192)  576
['conv2d_54[0][0]']
 ormalization)


 batch_normalization_49 (BatchN  (None, 12, 12, 192)  576
['conv2d_55[0][0]']
 ormalization)


 activation_40 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_40[0][0]']


 activation_43 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_43[0][0]']


 activation_48 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_48[0][0]']


 activation_49 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_49[0][0]']


 mixed5 (Concatenate)           (None, 12, 12, 768)  0
['activation_40[0][0]',

'activation_43[0][0]',
```

```
'activation_48[0][0]',

'activation_49[0][0]']


 conv2d_60 (Conv2D)             (None, 12, 12, 160)  122880
['mixed5[0][0]']


 batch_normalization_54 (BatchN  (None, 12, 12, 160)  480
['conv2d_60[0][0]']
 ormalization)



 activation_54 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_54[0][0]']


 conv2d_61 (Conv2D)             (None, 12, 12, 160)  179200
['activation_54[0][0]']


 batch_normalization_55 (BatchN  (None, 12, 12, 160)  480
['conv2d_61[0][0]']
 ormalization)



 activation_55 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_55[0][0]']


 conv2d_57 (Conv2D)             (None, 12, 12, 160)  122880
['mixed5[0][0]']


 conv2d_62 (Conv2D)             (None, 12, 12, 160)  179200
['activation_55[0][0]']


 batch_normalization_51 (BatchN  (None, 12, 12, 160)  480
['conv2d_57[0][0]']
 ormalization)



 batch_normalization_56 (BatchN  (None, 12, 12, 160)  480
```

```
['conv2d_62[0][0]']
 ormalization)



 activation_51 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_51[0][0]']


 activation_56 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_56[0][0]']


 conv2d_58 (Conv2D)             (None, 12, 12, 160)  179200
['activation_51[0][0]']


 conv2d_63 (Conv2D)             (None, 12, 12, 160)  179200
['activation_56[0][0]']


 batch_normalization_52 (BatchN  (None, 12, 12, 160)  480
['conv2d_58[0][0]']
 ormalization)



 batch_normalization_57 (BatchN  (None, 12, 12, 160)  480
['conv2d_63[0][0]']
 ormalization)



 activation_52 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_52[0][0]']


 activation_57 (Activation)     (None, 12, 12, 160)  0
['batch_normalization_57[0][0]']


 average_pooling2d_5 (AveragePo  (None, 12, 12, 768)  0
['mixed5[0][0]']
 oling2D)



 conv2d_56 (Conv2D)             (None, 12, 12, 192)  147456
```

```
['mixed5[0][0]']


 conv2d_59 (Conv2D)             (None, 12, 12, 192)  215040
['activation_52[0][0]']


 conv2d_64 (Conv2D)             (None, 12, 12, 192)  215040
['activation_57[0][0]']


 conv2d_65 (Conv2D)             (None, 12, 12, 192)  147456
['average_pooling2d_5[0][0]']


 batch_normalization_50 (BatchN  (None, 12, 12, 192)  576
['conv2d_56[0][0]']
 ormalization)



 batch_normalization_53 (BatchN  (None, 12, 12, 192)  576
['conv2d_59[0][0]']
 ormalization)



 batch_normalization_58 (BatchN  (None, 12, 12, 192)  576
['conv2d_64[0][0]']
 ormalization)



 batch_normalization_59 (BatchN  (None, 12, 12, 192)  576
['conv2d_65[0][0]']
 ormalization)



 activation_50 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_50[0][0]']


 activation_53 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_53[0][0]']


 activation_58 (Activation)     (None, 12, 12, 192)  0
```

```
['batch_normalization_58[0][0]']


 activation_59 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_59[0][0]']


 mixed6 (Concatenate)           (None, 12, 12, 768)  0
['activation_50[0][0]',

'activation_53[0][0]',

'activation_58[0][0]',

'activation_59[0][0]']


 conv2d_70 (Conv2D)             (None, 12, 12, 192)  147456
['mixed6[0][0]']


 batch_normalization_64 (BatchN  (None, 12, 12, 192)  576
['conv2d_70[0][0]']
 ormalization)



 activation_64 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_64[0][0]']


 conv2d_71 (Conv2D)             (None, 12, 12, 192)  258048
['activation_64[0][0]']


 batch_normalization_65 (BatchN  (None, 12, 12, 192)  576
['conv2d_71[0][0]']
 ormalization)



 activation_65 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_65[0][0]']


 conv2d_67 (Conv2D)             (None, 12, 12, 192)  147456
['mixed6[0][0]']
```

```
 conv2d_72 (Conv2D)             (None, 12, 12, 192)  258048
['activation_65[0][0]']


 batch_normalization_61 (BatchN  (None, 12, 12, 192)  576
['conv2d_67[0][0]']
 ormalization)



 batch_normalization_66 (BatchN  (None, 12, 12, 192)  576
['conv2d_72[0][0]']
 ormalization)



 activation_61 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_61[0][0]']


 activation_66 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_66[0][0]']


 conv2d_68 (Conv2D)             (None, 12, 12, 192)  258048
['activation_61[0][0]']


 conv2d_73 (Conv2D)             (None, 12, 12, 192)  258048
['activation_66[0][0]']


 batch_normalization_62 (BatchN  (None, 12, 12, 192)  576
['conv2d_68[0][0]']
 ormalization)



 batch_normalization_67 (BatchN  (None, 12, 12, 192)  576
['conv2d_73[0][0]']
 ormalization)



 activation_62 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_62[0][0]']
```

```
 activation_67 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_67[0][0]']


 average_pooling2d_6 (AveragePo  (None, 12, 12, 768)  0
['mixed6[0][0]']
 oling2D)


 conv2d_66 (Conv2D)             (None, 12, 12, 192)  147456
['mixed6[0][0]']


 conv2d_69 (Conv2D)             (None, 12, 12, 192)  258048
['activation_62[0][0]']


 conv2d_74 (Conv2D)             (None, 12, 12, 192)  258048
['activation_67[0][0]']


 conv2d_75 (Conv2D)             (None, 12, 12, 192)  147456
['average_pooling2d_6[0][0]']


 batch_normalization_60 (BatchN  (None, 12, 12, 192)  576
['conv2d_66[0][0]']
 ormalization)


 batch_normalization_63 (BatchN  (None, 12, 12, 192)  576
['conv2d_69[0][0]']
 ormalization)


 batch_normalization_68 (BatchN  (None, 12, 12, 192)  576
['conv2d_74[0][0]']
 ormalization)


 batch_normalization_69 (BatchN  (None, 12, 12, 192)  576
['conv2d_75[0][0]']
 ormalization)
```

```
 activation_60 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_60[0][0]']


 activation_63 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_63[0][0]']


 activation_68 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_68[0][0]']


 activation_69 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_69[0][0]']


 mixed7 (Concatenate)           (None, 12, 12, 768)  0
['activation_60[0][0]',

'activation_63[0][0]',

'activation_68[0][0]',

'activation_69[0][0]']


 conv2d_78 (Conv2D)             (None, 12, 12, 192)  147456
['mixed7[0][0]']


 batch_normalization_72 (BatchN  (None, 12, 12, 192)  576
['conv2d_78[0][0]']
 ormalization)



 activation_72 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_72[0][0]']


 conv2d_79 (Conv2D)             (None, 12, 12, 192)  258048
['activation_72[0][0]']


 batch_normalization_73 (BatchN  (None, 12, 12, 192)  576
```

```
['conv2d_79[0][0]']
 ormalization)



 activation_73 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_73[0][0]']


 conv2d_76 (Conv2D)             (None, 12, 12, 192)  147456
['mixed7[0][0]']


 conv2d_80 (Conv2D)             (None, 12, 12, 192)  258048
['activation_73[0][0]']


 batch_normalization_70 (BatchN  (None, 12, 12, 192)  576
['conv2d_76[0][0]']
 ormalization)



 batch_normalization_74 (BatchN  (None, 12, 12, 192)  576
['conv2d_80[0][0]']
 ormalization)



 activation_70 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_70[0][0]']


 activation_74 (Activation)     (None, 12, 12, 192)  0
['batch_normalization_74[0][0]']


 conv2d_77 (Conv2D)             (None, 5, 5, 320)    552960
['activation_70[0][0]']


 conv2d_81 (Conv2D)             (None, 5, 5, 192)    331776
['activation_74[0][0]']


 batch_normalization_71 (BatchN  (None, 5, 5, 320)   960
['conv2d_77[0][0]']
 ormalization)
```

```
 batch_normalization_75 (BatchN  (None, 5, 5, 192)   576
['conv2d_81[0][0]']
 ormalization)



 activation_71 (Activation)     (None, 5, 5, 320)    0
['batch_normalization_71[0][0]']


 activation_75 (Activation)     (None, 5, 5, 192)    0
['batch_normalization_75[0][0]']


 max_pooling2d_9 (MaxPooling2D)  (None, 5, 5, 768)   0
['mixed7[0][0]']


 mixed8 (Concatenate)           (None, 5, 5, 1280)   0
['activation_71[0][0]',

'activation_75[0][0]',

'max_pooling2d_9[0][0]']


 conv2d_86 (Conv2D)             (None, 5, 5, 448)    573440
['mixed8[0][0]']


 batch_normalization_80 (BatchN  (None, 5, 5, 448)   1344
['conv2d_86[0][0]']
 ormalization)



 activation_80 (Activation)     (None, 5, 5, 448)    0
['batch_normalization_80[0][0]']


 conv2d_83 (Conv2D)             (None, 5, 5, 384)    491520
['mixed8[0][0]']


 conv2d_87 (Conv2D)             (None, 5, 5, 384)    1548288
```

```
['activation_80[0][0]']


 batch_normalization_77 (BatchN  (None, 5, 5, 384)   1152
['conv2d_83[0][0]']
 ormalization)



 batch_normalization_81 (BatchN  (None, 5, 5, 384)   1152
['conv2d_87[0][0]']
 ormalization)



 activation_77 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_77[0][0]']


 activation_81 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_81[0][0]']


 conv2d_84 (Conv2D)             (None, 5, 5, 384)    442368
['activation_77[0][0]']


 conv2d_85 (Conv2D)             (None, 5, 5, 384)    442368
['activation_77[0][0]']


 conv2d_88 (Conv2D)             (None, 5, 5, 384)    442368
['activation_81[0][0]']


 conv2d_89 (Conv2D)             (None, 5, 5, 384)    442368
['activation_81[0][0]']


 average_pooling2d_7 (AveragePo  (None, 5, 5, 1280)  0
['mixed8[0][0]']
 oling2D)



 conv2d_82 (Conv2D)             (None, 5, 5, 320)    409600
['mixed8[0][0]']
```

```
 batch_normalization_78 (BatchN  (None, 5, 5, 384)   1152
['conv2d_84[0][0]']
 ormalization)


 batch_normalization_79 (BatchN  (None, 5, 5, 384)   1152
['conv2d_85[0][0]']
 ormalization)


 batch_normalization_82 (BatchN  (None, 5, 5, 384)   1152
['conv2d_88[0][0]']
 ormalization)


 batch_normalization_83 (BatchN  (None, 5, 5, 384)   1152
['conv2d_89[0][0]']
 ormalization)


 conv2d_90 (Conv2D)             (None, 5, 5, 192)    245760
['average_pooling2d_7[0][0]']

 batch_normalization_76 (BatchN  (None, 5, 5, 320)   960
['conv2d_82[0][0]']
 ormalization)


 activation_78 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_78[0][0]']

 activation_79 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_79[0][0]']

 activation_82 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_82[0][0]']

 activation_83 (Activation)     (None, 5, 5, 384)    0
```

```
['batch_normalization_83[0][0]']


 batch_normalization_84 (BatchN  (None, 5, 5, 192)   576
['conv2d_90[0][0]']
 ormalization)



 activation_76 (Activation)     (None, 5, 5, 320)    0
['batch_normalization_76[0][0]']


 mixed9_0 (Concatenate)         (None, 5, 5, 768)    0
['activation_78[0][0]',

'activation_79[0][0]']


 concatenate (Concatenate)      (None, 5, 5, 768)    0
['activation_82[0][0]',

'activation_83[0][0]']


 activation_84 (Activation)     (None, 5, 5, 192)    0
['batch_normalization_84[0][0]']


 mixed9 (Concatenate)           (None, 5, 5, 2048)   0
['activation_76[0][0]',

'mixed9_0[0][0]',

'concatenate[0][0]',

'activation_84[0][0]']


 conv2d_95 (Conv2D)             (None, 5, 5, 448)    917504
['mixed9[0][0]']


 batch_normalization_89 (BatchN  (None, 5, 5, 448)   1344
['conv2d_95[0][0]']
 ormalization)
```

```
 activation_89 (Activation)     (None, 5, 5, 448)    0
['batch_normalization_89[0][0]']


 conv2d_92 (Conv2D)             (None, 5, 5, 384)    786432
['mixed9[0][0]']


 conv2d_96 (Conv2D)             (None, 5, 5, 384)    1548288
['activation_89[0][0]']


 batch_normalization_86 (BatchN  (None, 5, 5, 384)   1152
['conv2d_92[0][0]']
 ormalization)



 batch_normalization_90 (BatchN  (None, 5, 5, 384)   1152
['conv2d_96[0][0]']
 ormalization)



 activation_86 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_86[0][0]']


 activation_90 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_90[0][0]']


 conv2d_93 (Conv2D)             (None, 5, 5, 384)    442368
['activation_86[0][0]']


 conv2d_94 (Conv2D)             (None, 5, 5, 384)    442368
['activation_86[0][0]']


 conv2d_97 (Conv2D)             (None, 5, 5, 384)    442368
['activation_90[0][0]']


 conv2d_98 (Conv2D)             (None, 5, 5, 384)    442368
['activation_90[0][0]']
```

```
 average_pooling2d_8 (AveragePo  (None, 5, 5, 2048)  0
['mixed9[0][0]']
 oling2D)


 conv2d_91 (Conv2D)             (None, 5, 5, 320)    655360
['mixed9[0][0]']

 batch_normalization_87 (BatchN  (None, 5, 5, 384)   1152
['conv2d_93[0][0]']
 ormalization)


 batch_normalization_88 (BatchN  (None, 5, 5, 384)   1152
['conv2d_94[0][0]']
 ormalization)


 batch_normalization_91 (BatchN  (None, 5, 5, 384)   1152
['conv2d_97[0][0]']
 ormalization)


 batch_normalization_92 (BatchN  (None, 5, 5, 384)   1152
['conv2d_98[0][0]']
 ormalization)


 conv2d_99 (Conv2D)             (None, 5, 5, 192)    393216
['average_pooling2d_8[0][0]']

 batch_normalization_85 (BatchN  (None, 5, 5, 320)   960
['conv2d_91[0][0]']
 ormalization)


 activation_87 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_87[0][0]']
```

```
 activation_88 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_88[0][0]']


 activation_91 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_91[0][0]']


 activation_92 (Activation)     (None, 5, 5, 384)    0
['batch_normalization_92[0][0]']


 batch_normalization_93 (BatchN  (None, 5, 5, 192)   576
['conv2d_99[0][0]']
 ormalization)



 activation_85 (Activation)     (None, 5, 5, 320)    0
['batch_normalization_85[0][0]']


 mixed9_1 (Concatenate)         (None, 5, 5, 768)    0
['activation_87[0][0]',

'activation_88[0][0]']


 concatenate_1 (Concatenate)    (None, 5, 5, 768)    0
['activation_91[0][0]',

'activation_92[0][0]']


 activation_93 (Activation)     (None, 5, 5, 192)    0
['batch_normalization_93[0][0]']


 mixed10 (Concatenate)          (None, 5, 5, 2048)   0
['activation_85[0][0]',

'mixed9_1[0][0]',

'concatenate_1[0][0]',

'activation_93[0][0]']
```

```
 flatten_2 (Flatten)           (None, 51200)       0
['mixed10[0][0]']


 dense_4 (Dense)               (None, 10)          512010
['flatten_2[0][0]']


======================================================================
============================
Total params: 22,314,794
Trainable params: 512,010
Non-trainable params: 21,802,784
______________________________________________________________________
____________________________
```

```
model.compile(
  loss='categorical_crossentropy',
  optimizer='adam',
  metrics=['accuracy']
)

#Importing images from the dataset
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Defining pre-processing transformations on raw images of training
data
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   shear_range = 0.2,
                                   zoom_range = 0.2,
                                   horizontal_flip = True)

# Defining pre-processing transformations on raw images of testing
data
test_datagen = ImageDataGenerator(rescale = 1./255)

# Load the training set and find number of images
training_set =
train_datagen.flow_from_directory('/content/drive/MyDrive/impressionis
t/training/training',
                                                 target_size = (224,
224),
                                                 batch_size = 32,
                                                 class_mode =
'categorical')
```

```
Found 3988 images belonging to 10 classes.
```

```
test_set =
test_datagen.flow_from_directory('/content/drive/MyDrive/impressionist
```

```
/validation/validation',
                                                 target_size = (224, 224),
                                                 batch_size = 32,
                                                 class_mode =
'categorical')
```

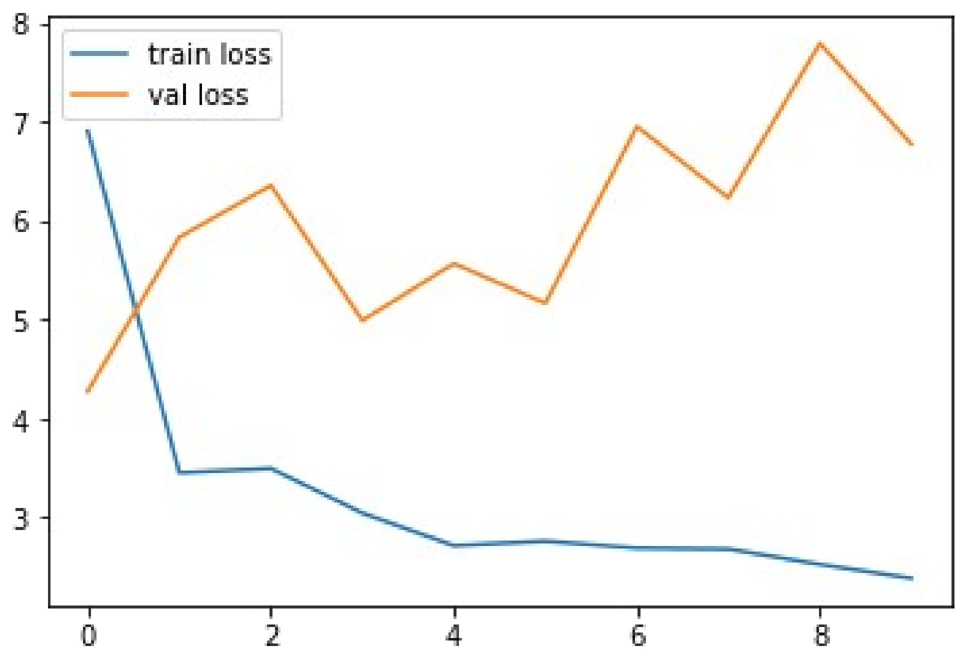
```
Found 990 images belonging to 10 classes.
```

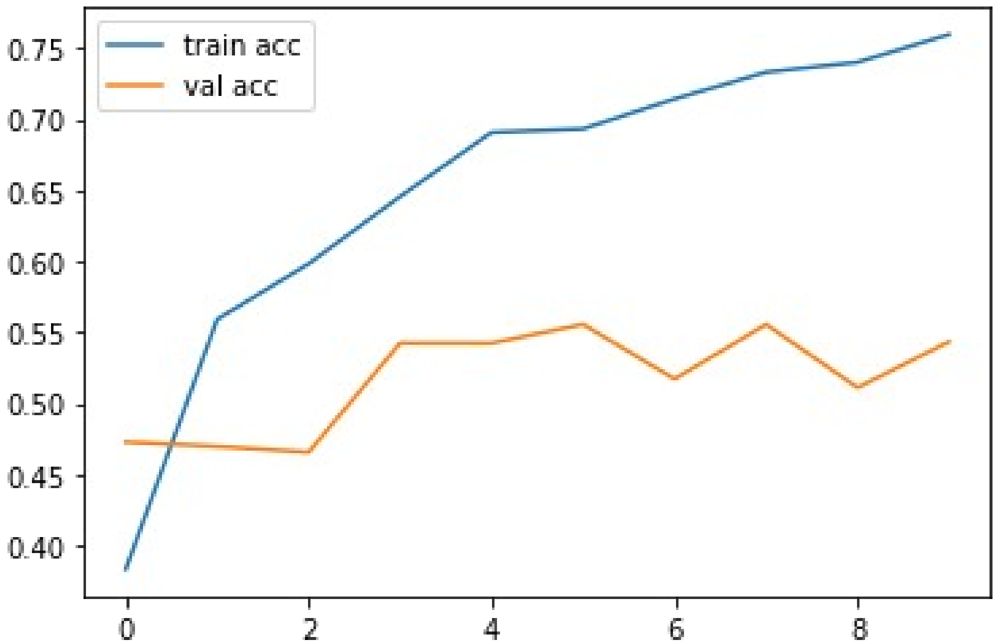
```
r = model.fit_generator(
  training_set,
  validation_data=test_set,
  epochs=10,
  steps_per_epoch=len(training_set),
  validation_steps=len(test_set)
)
```

```
<ipython-input-41-5932e3024fde>:1: UserWarning: `Model.fit_generator`
is deprecated and will be removed in a future version. Please use
`Model.fit`, which supports generators.
  r = model.fit_generator(

Epoch 1/10
125/125 [==============================] - 744s 6s/step - loss: 6.9032
- accuracy: 0.3831 - val_loss: 4.2751 - val_accuracy: 0.4727
Epoch 2/10
125/125 [==============================] - 720s 6s/step - loss: 3.4511
- accuracy: 0.5592 - val_loss: 5.8303 - val_accuracy: 0.4697
Epoch 3/10
125/125 [==============================] - 668s 5s/step - loss: 3.4942
- accuracy: 0.5983 - val_loss: 6.3525 - val_accuracy: 0.4657
Epoch 4/10
125/125 [==============================] - 670s 5s/step - loss: 3.0434
- accuracy: 0.6457 - val_loss: 4.9889 - val_accuracy: 0.5424
Epoch 5/10
125/125 [==============================] - 673s 5s/step - loss: 2.7136
- accuracy: 0.6908 - val_loss: 5.5642 - val_accuracy: 0.5424
Epoch 6/10
125/125 [==============================] - 674s 5s/step - loss: 2.7595
- accuracy: 0.6933 - val_loss: 5.1629 - val_accuracy: 0.5556
Epoch 7/10
125/125 [==============================] - 670s 5s/step - loss: 2.6889
- accuracy: 0.7144 - val_loss: 6.9475 - val_accuracy: 0.5172
Epoch 8/10
125/125 [==============================] - 680s 5s/step - loss: 2.6792
- accuracy: 0.7332 - val_loss: 6.2279 - val_accuracy: 0.5556
Epoch 9/10
125/125 [==============================] - 673s 5s/step - loss: 2.5250
- accuracy: 0.7402 - val_loss: 7.7884 - val_accuracy: 0.5111
Epoch 10/10
125/125 [==============================] - 673s 5s/step - loss: 2.3841
- accuracy: 0.7598 - val_loss: 6.7700 - val_accuracy: 0.5434
```

Plotting the Data

```
# plot the loss
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')

# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```





<Figure size 432x288 with 0 Axes>