Image Classification

Work done by: Kyle Chan kxc180021 Ryan Banafshay rcb170002

We are using a Kaggle dataset from here (https://www.kaggle.com/datasets/puneet6060/intel-image-classification). It consists of about 25,000 images of size 150x150 distributed over 6 categories.

First, we will import some packages.

In [7]:
```
import numpy as np
import os
import pandas as pd
import tensorflow as tf
```

We'll build a function that will load the images from the dataset into our training and testing sets. They are already pre-split from Kaggle.

In [8]:
```
import os
from tqdm import tqdm
import cv2

class_names = ['buildings', 'forest', 'glacier', 'mountain', 'sea', 'street']
class_names_label = {class_names:i for i, class_names in enumerate(class_names)

image_size = (150, 150)

def get_images_and_labels():
    datasets = ['archive/seg_train/seg_train', 'archive/seg_test/seg_test']
    output = []

    for dataset in datasets:
        images = []
        labels = []

        print(f"loading {dataset}")

        for folder in os.listdir(dataset):
            label = class_names_label[folder]

            for file in tqdm(os.listdir(os.path.join(dataset, folder))):

                img_path = os.path.join(os.path.join(dataset, folder), file)

                img = cv2.resize((cv2.cvtColor(cv2.imread(img_path), cv2.COLOR_

                images.append(img)
                labels.append(label)

        images = np.array(images, dtype='float32')
        labels = np.array(labels, dtype='int32')

        output.append((images, labels))
    return output
```

In [9]:
```
(train_images, train_labels), (test_images, test_labels) = get_images_and_label
```

```
loading archive/seg_train/seg_train
100%|██████████████████████████████████████| 2271/2271 [00:01<00:00, 1579.12it/
s]
100%|██████████████████████████████████████| 2191/2191 [00:01<00:00, 1594.53it/
s]
100%|███████████████████████████████████████| 2404/2404 [00:02<00:00, 889.67it/
s]
100%|██████████████████████████████████████| 2382/2382 [00:01<00:00, 1287.19it/
s]
100%|██████████████████████████████████████| 2512/2512 [00:01<00:00, 1753.12it/
s]
100%|██████████████████████████████████████| 2274/2274 [00:01<00:00, 1658.95it/
s]
loading archive/seg_test/seg_test
100%|█████████████████████████████████████████| 474/474 [00:01<00:00, 316.47it/
s]
100%|█████████████████████████████████████████| 437/437 [00:00<00:00, 551.17it/
s]
100%|████████████████████████████████████████| 553/553 [00:00<00:00, 1705.90it/
s]
100%|████████████████████████████████████████| 501/501 [00:00<00:00, 1673.66it/
s]
100%|████████████████████████████████████████| 525/525 [00:00<00:00, 1856.44it/
s]
100%|████████████████████████████████████████| 510/510 [00:00<00:00, 1868.57it/
s]
```

In [10]:
```
from sklearn.utils import shuffle
train_images, train_labels = shuffle(train_images, train_labels, random_state=1
```

Data exploration

Let's explore our data and find out the shape and size of our dataset.

In [11]:
```
print(f'Train shape: {train_labels.shape}')
print(f'Test shape: {test_labels.shape}')
print(f'Image sizes: {image_size}')
```

```
Train shape: (14034,)
Test shape: (3000,)
Image sizes: (150, 150)
```

We will use Matplotlib to plot a pie graph showing the distribution of images and their respective labels, then plot a bar graph showing the proportions of the size of the training and testing dataset.

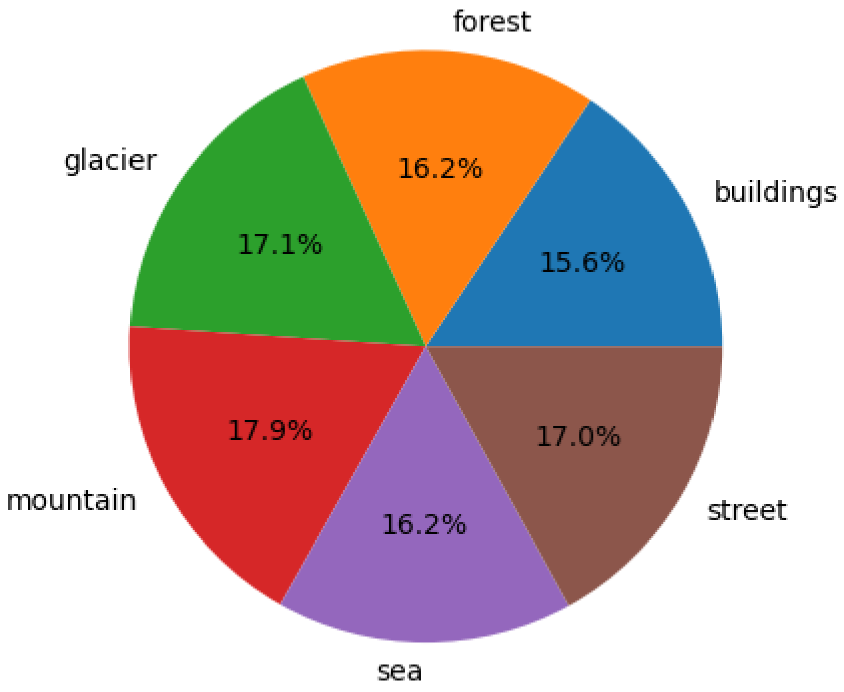
In [12]:
```
import matplotlib.pyplot as plt

_, train_count = np.unique(train_labels, return_counts=True)
_, test_count = np.unique(test_labels, return_counts=True)
```

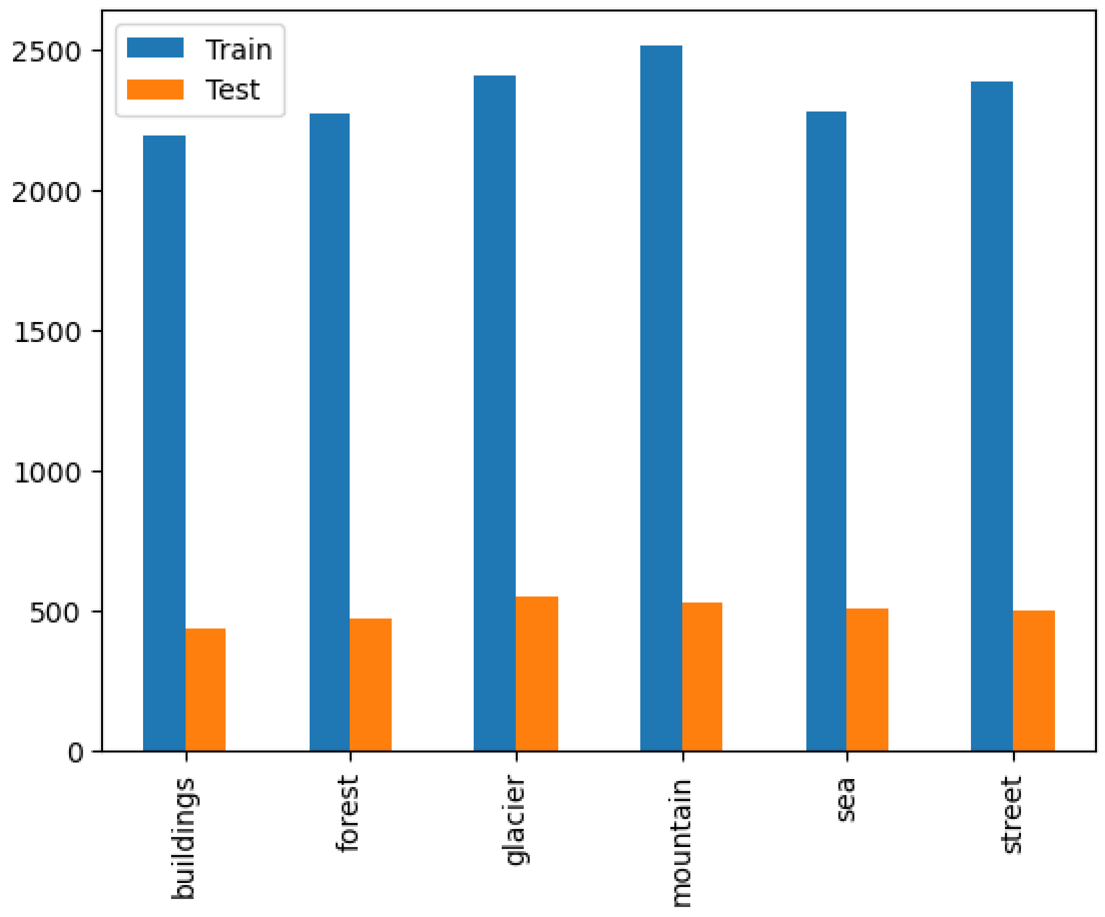
```
plt.pie(train_count,
        explode=(0,0,0,0,0,0),
        labels=class_names,
        autopct='%1.1f%%')
plt.show()


df = pd.DataFrame({'Train': train_count,
              'Test': test_count},
              index=class_names,)

df.plot.bar()
```



```
Out[12]: <Axes: >
```



As seen above, the images are labelled at a roughly equal distribution, meaning that we will have an equal amount of training to predict each category.

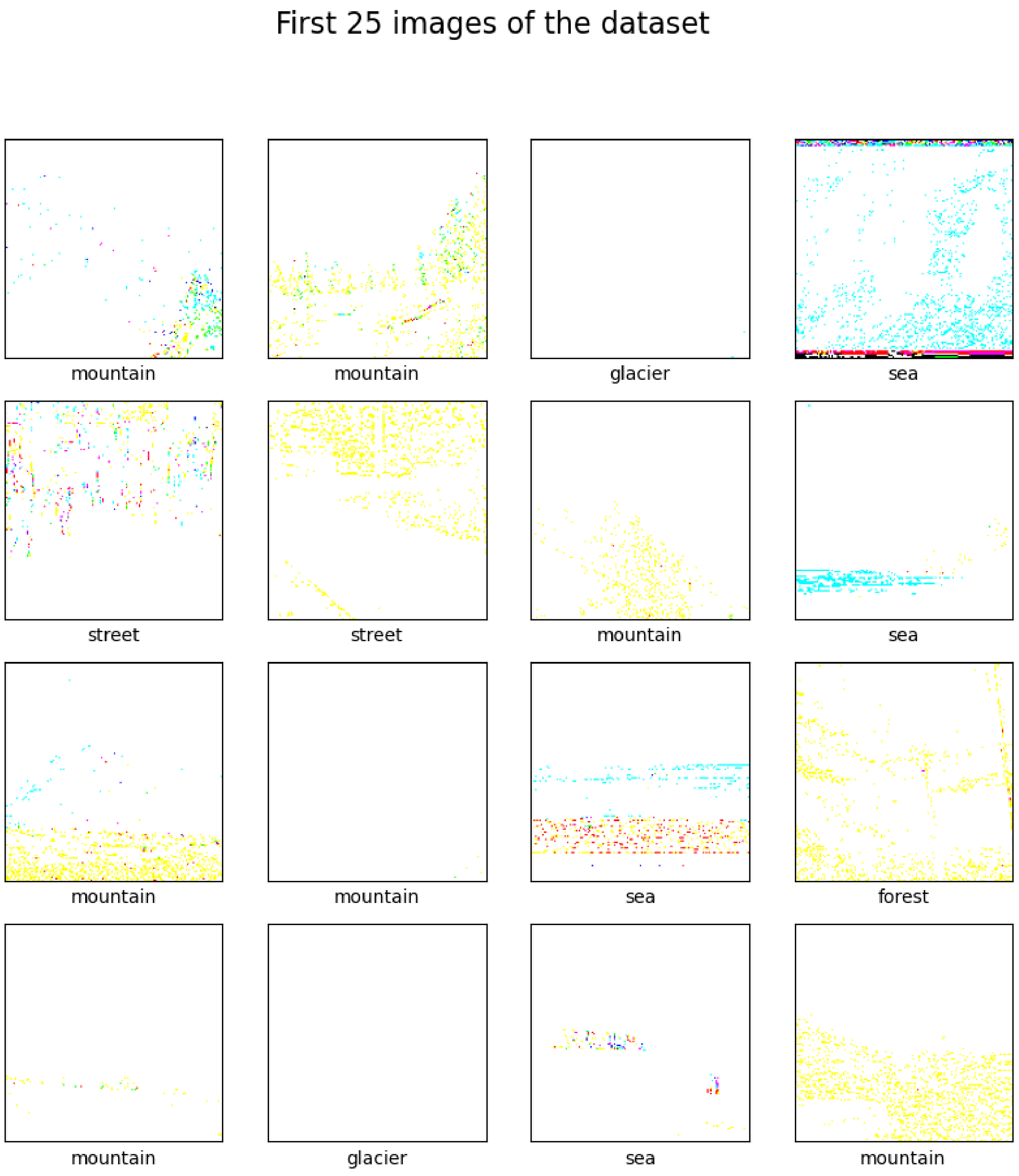
Next, let's see what kind of images we are working with. Below, we have created a function that will display the first 25 images of the dataset.

```
In [13]: def display_examples(class_names, images, labels):

             fig = plt.figure(figsize=(10,10))
             fig.suptitle("First 25 images of the dataset", fontsize=16)
             for i in range(16):
                 plt.subplot(4,4,i+1)
                 plt.xticks([])
                 plt.yticks([])
                 plt.grid(False)
                 plt.imshow(images[i], cmap=plt.cm.binary)
                 plt.xlabel(class_names[labels[i]])
             plt.show()

         display_examples(class_names, train_images, train_labels)
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
Clipping input data to the valid range for imshow with RGB data ([0..1] for fl
oats or [0..255] for integers).
```



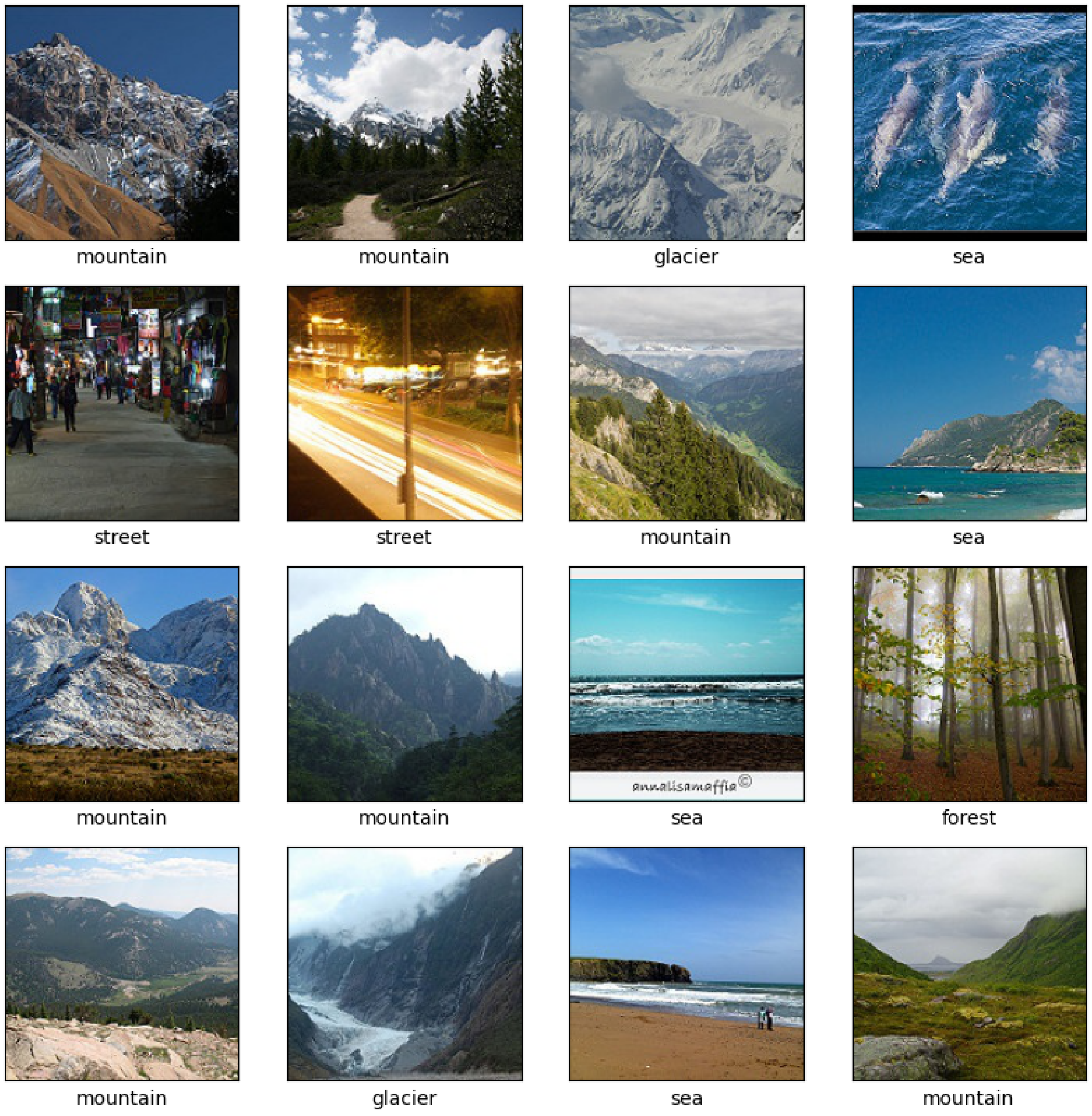
As we can see, the images are extremely distorted, oversaturated, and too bright.

We now know that we must scale our data down to a visible color spectrum. We will redisplay the images after scaling down.

In [14]:
```
train_images = train_images / 255
test_images = test_images / 255

display_examples(class_names, train_images, train_labels)
```

First 25 images of the dataset



Sequential (CNN)

Now, we will create a model using Tensorflow's Keras class.

There are several layers and components to the model such as:

- Conv2D: used for 'extracting' features from the image being evaluated
- MaxPooling2D: used to reduce the image size by half
- Flatten: transforms the image being read from a 2D-array to a 1D-array of pixel values
- Relu: an activation function that, given x, will return the max between x and 0

- Softmax: set to 6 neurons in our case, corresponding to one of the 6 unique labels we want as an output

We will use the 'adam' optimizer. 'Adam' is an optimization algorithm that combines the benefits of two algorithms:

- RMSProp: (Root Mean Square Propagation) exponentialy weights past gradients the further they are from the current layer
- Momentum: takes into account the past gradient and carries forward a small portion of the previous vector so that the optimizer keeps moving in relatively the same direction

We will also use a sparse categorical crossentropy loss function, since each image should only belong to one class.

In [15]:
```
CNN_model = tf.keras.models.Sequential(
    [
        tf.keras.Input(shape=(150,150,3)),
        tf.keras.layers.Conv2D(32, kernel_size=(3, 3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Conv2D(64, kernel_size=(3, 3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size=(2, 2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation='relu'),
        tf.keras.layers.Dense(6, activation="softmax"),
    ]
)

CNN_model.compile(optimizer='adam', loss='sparse_categorical_crossentropy', met
```

```
2023-04-22 12:15:46.792759: I tensorflow/core/platform/cpu_feature_guard.cc:19
3] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical operati
ons:  SSE4.1 SSE4.2
To enable them in other operations, rebuild TensorFlow with the appropriate co
mpiler flags.
```

Let's output the layers and see the breakdown of our model's parameters.

In [16]:
```
CNN_model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 conv2d (Conv2D)             (None, 148, 148, 32)      896       
                                                                 
 max_pooling2d (MaxPooling2D  (None, 74, 74, 32)       0         
 )                                                               
                                                                 
 conv2d_1 (Conv2D)           (None, 72, 72, 64)        18496     
                                                                 
 max_pooling2d_1 (MaxPooling  (None, 36, 36, 64)       0         
 2D)                                                             
                                                                 
 flatten (Flatten)           (None, 82944)             0         
                                                                 
 dense (Dense)               (None, 128)               10616960  
                                                                 
 dense_1 (Dense)             (None, 6)                 774       
                                                                 
=================================================================
Total params: 10,637,126
Trainable params: 10,637,126
Non-trainable params: 0
_________________________________________________________________
```

Now, we can fit our training data to our newly created model.

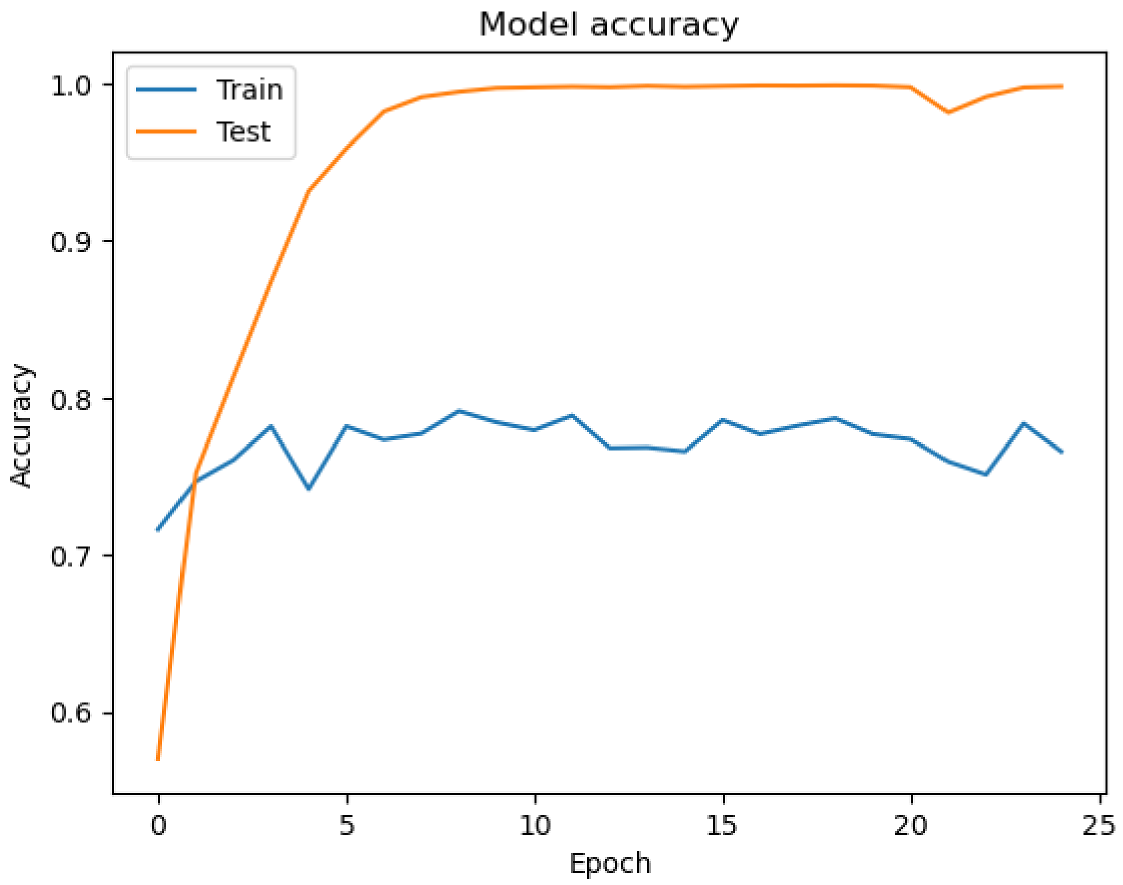
In [19]:
```
history = CNN_model.fit(train_images, train_labels,
                    batch_size=128,
                    epochs=25,
                    verbose=1,
                    validation_split=0.2)
```

```
Epoch 1/25
88/88 [==============================] - 150s 2s/step - loss: 1.2566 - accurac
y: 0.5705 - val_loss: 0.7828 - val_accuracy: 0.7164
Epoch 2/25
88/88 [==============================] - 139s 2s/step - loss: 0.6861 - accurac
y: 0.7522 - val_loss: 0.6732 - val_accuracy: 0.7471
Epoch 3/25
88/88 [==============================] - 136s 2s/step - loss: 0.5285 - accurac
y: 0.8136 - val_loss: 0.6513 - val_accuracy: 0.7606
Epoch 4/25
88/88 [==============================] - 141s 2s/step - loss: 0.3718 - accurac
y: 0.8741 - val_loss: 0.6093 - val_accuracy: 0.7820
Epoch 5/25
88/88 [==============================] - 129s 1s/step - loss: 0.2243 - accurac
y: 0.9318 - val_loss: 0.7446 - val_accuracy: 0.7421
Epoch 6/25
88/88 [==============================] - 142s 2s/step - loss: 0.1489 - accurac
y: 0.9587 - val_loss: 0.6833 - val_accuracy: 0.7820
Epoch 7/25
88/88 [==============================] - 132s 2s/step - loss: 0.0800 - accurac
y: 0.9822 - val_loss: 0.7619 - val_accuracy: 0.7734
Epoch 8/25
88/88 [==============================] - 153s 2s/step - loss: 0.0510 - accurac
y: 0.9914 - val_loss: 0.8822 - val_accuracy: 0.7773
Epoch 9/25
88/88 [==============================] - 157s 2s/step - loss: 0.0338 - accurac
y: 0.9947 - val_loss: 0.8734 - val_accuracy: 0.7916
Epoch 10/25
88/88 [==============================] - 156s 2s/step - loss: 0.0268 - accurac
y: 0.9971 - val_loss: 0.8946 - val_accuracy: 0.7845
Epoch 11/25
88/88 [==============================] - 134s 2s/step - loss: 0.0201 - accurac
y: 0.9976 - val_loss: 0.9490 - val_accuracy: 0.7795
Epoch 12/25
88/88 [==============================] - 115s 1s/step - loss: 0.0196 - accurac
y: 0.9980 - val_loss: 0.9528 - val_accuracy: 0.7887
Epoch 13/25
88/88 [==============================] - 116s 1s/step - loss: 0.0207 - accurac
y: 0.9977 - val_loss: 1.0635 - val_accuracy: 0.7681
Epoch 14/25
88/88 [==============================] - 134s 2s/step - loss: 0.0161 - accurac
y: 0.9985 - val_loss: 1.1884 - val_accuracy: 0.7684
Epoch 15/25
88/88 [==============================] - 148s 2s/step - loss: 0.0180 - accurac
y: 0.9980 - val_loss: 1.1503 - val_accuracy: 0.7659
Epoch 16/25
88/88 [==============================] - 128s 1s/step - loss: 0.0160 - accurac
y: 0.9984 - val_loss: 1.0237 - val_accuracy: 0.7859
Epoch 17/25
88/88 [==============================] - 122s 1s/step - loss: 0.0142 - accurac
y: 0.9988 - val_loss: 1.0572 - val_accuracy: 0.7770
Epoch 18/25
88/88 [==============================] - 129s 1s/step - loss: 0.0142 - accurac
y: 0.9987 - val_loss: 0.9818 - val_accuracy: 0.7823
Epoch 19/25
88/88 [==============================] - 133s 2s/step - loss: 0.0127 - accurac
y: 0.9988 - val_loss: 0.9722 - val_accuracy: 0.7870
Epoch 20/25
88/88 [==============================] - 128s 1s/step - loss: 0.0134 - accurac
y: 0.9987 - val_loss: 1.0220 - val_accuracy: 0.7770
```

```
Epoch 21/25
88/88 [==============================] - 130s 1s/step - loss: 0.0180 - accurac
y: 0.9977 - val_loss: 1.1190 - val_accuracy: 0.7738
Epoch 22/25
88/88 [==============================] - 135s 2s/step - loss: 0.0628 - accurac
y: 0.9816 - val_loss: 1.0601 - val_accuracy: 0.7595
Epoch 23/25
88/88 [==============================] - 132s 2s/step - loss: 0.0355 - accurac
y: 0.9915 - val_loss: 1.0806 - val_accuracy: 0.7513
Epoch 24/25
88/88 [==============================] - 125s 1s/step - loss: 0.0174 - accurac
y: 0.9975 - val_loss: 1.0987 - val_accuracy: 0.7838
Epoch 25/25
88/88 [==============================] - 127s 1s/step - loss: 0.0150 - accurac
y: 0.9981 - val_loss: 1.1927 - val_accuracy: 0.7659
```

We can see that our model has reached a peak accuracy of 99%, but let's plot the model's accuracy history over time to further analyze.

In [20]:
```python
plt.plot(history.history['val_accuracy'])
plt.plot(history.history['accuracy'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```



As shown above, the model's training accuracy seems to plateau very early, but our testing prediction accuracy continued to approach 100%. We believe that this may actually be a positive sign, since it appears that the model is generalizing the dataset better to the new data. The model does not seem like it is memorizing the training data, therefore we have determined that the model is not overfitting.

Pretrained Model

A pre-trained model is a saved network that was previously trained on with some other large image classification dataset. For this part, we will use MobileNetV2. This is a model with a convolutional neural network architecture that tailors its performace around mobile devices.

In [21]:
```
import matplotlib.pyplot as plt
import numpy as np
import os
import tensorflow as tf

data_augmentation = tf.keras.Sequential([
  tf.keras.layers.RandomFlip('horizontal'),
  tf.keras.layers.RandomRotation(0.2),
])
```

In [22]:
```
preprocess_input = tf.keras.applications.mobilenet_v2.preprocess_input

rescale = tf.keras.layers.Rescaling(1./127.5, offset=-1)
```

Create the base model from the pre-trained model MobileNet V2. This model is built completely independent from our image dataset.

In [24]:
```
IMG_SHAPE = image_size + (3,)
base_model = tf.keras.applications.MobileNetV2(input_shape=IMG_SHAPE,
                                               include_top=False,
                                               weights='imagenet')
```

```
WARNING:tensorflow:`input_shape` is undefined or non-square, or `rows` is not
in [96, 128, 160, 192, 224]. Weights for input shape (224, 224) will be loaded
as the default.
```

Adjusting the model to work for our image size and displaying the summary of the model.

In [25]:
```
base_model.trainable = False
base_model.summary()
```

```
Model: "mobilenetv2_1.00_224"
_______________________________________________________________________________
____________________
 Layer (type)                   Output Shape         Param #     Connected to
===============================================================================
====================
 input_3 (InputLayer)           [(None, 150, 150, 3  0           []
                                )]

 Conv1 (Conv2D)                 (None, 75, 75, 32)   864         ['input_3[0]
[0]']

 bn_Conv1 (BatchNormalization)  (None, 75, 75, 32)   128         ['Conv1[0]
[0]']

 Conv1_relu (ReLU)              (None, 75, 75, 32)   0           ['bn_Conv1[0]
[0]']

 expanded_conv_depthwise (Depth  (None, 75, 75, 32)  288         ['Conv1_relu
[0][0]']
 wiseConv2D)

 expanded_conv_depthwise_BN (Ba  (None, 75, 75, 32)  128         ['expanded_co
nv_depthwise[0][0]']
 tchNormalization)

 expanded_conv_depthwise_relu (  (None, 75, 75, 32)  0           ['expanded_co
nv_depthwise_BN[0][0
 ReLU)                                                           ]']

 expanded_conv_project (Conv2D)  (None, 75, 75, 16)  512         ['expanded_co
nv_depthwise_relu[0]
                                                                 [0]']

 expanded_conv_project_BN (Batc  (None, 75, 75, 16)  64          ['expanded_co
nv_project[0][0]']
 hNormalization)

 block_1_expand (Conv2D)        (None, 75, 75, 96)   1536        ['expanded_co
nv_project_BN[0][0]'
                                                                 ]

 block_1_expand_BN (BatchNormal  (None, 75, 75, 96)  384         ['block_1_exp
and[0][0]']
 ization)

 block_1_expand_relu (ReLU)     (None, 75, 75, 96)   0           ['block_1_exp
and_BN[0][0]']

 block_1_pad (ZeroPadding2D)    (None, 77, 77, 96)   0           ['block_1_exp
and_relu[0][0]']

 block_1_depthwise (DepthwiseCo  (None, 38, 38, 96)  864         ['block_1_pad
[0][0]']
 nv2D)

 block_1_depthwise_BN (BatchNor  (None, 38, 38, 96)  384         ['block_1_dep
thwise[0][0]']
 malization)
 block_1_depthwise_relu (ReLU)  (None, 38, 38, 96)   0           ['block_1_dep
thwise_BN[0][0]']

 block_1_project (Conv2D)       (None, 38, 38, 24)   2304        ['block_1_dep
thwise_relu[0][0]']

 block_1_project_BN (BatchNorma  (None, 38, 38, 24)  96          ['block_1_pro
ject[0][0]']
 lization)

 block_2_expand (Conv2D)        (None, 38, 38, 144)  3456        ['block_1_pro
ject_BN[0][0]']

 block_2_expand_BN (BatchNormal  (None, 38, 38, 144)  576        ['block_2_exp
and[0][0]']
 ization)

 block_2_expand_relu (ReLU)     (None, 38, 38, 144)  0           ['block_2_exp
and_BN[0][0]']

 block_2_depthwise (DepthwiseCo  (None, 38, 38, 144)  1296       ['block_2_exp
and_relu[0][0]']
 nv2D)

 block_2_depthwise_BN (BatchNor  (None, 38, 38, 144)  576        ['block_2_dep
thwise[0][0]']
 malization)

 block_2_depthwise_relu (ReLU)  (None, 38, 38, 144)  0           ['block_2_dep
thwise_BN[0][0]']

 block_2_project (Conv2D)       (None, 38, 38, 24)   3456        ['block_2_dep
thwise_relu[0][0]']

 block_2_project_BN (BatchNorma  (None, 38, 38, 24)  96          ['block_2_pro
ject[0][0]']
 lization)

 block_2_add (Add)              (None, 38, 38, 24)   0           ['block_1_pro
ject_BN[0][0]',
                                                                  'block_2_pro
ject_BN[0][0]']

 block_3_expand (Conv2D)        (None, 38, 38, 144)  3456        ['block_2_add
[0][0]']

 block_3_expand_BN (BatchNormal  (None, 38, 38, 144)  576        ['block_3_exp
and[0][0]']
 ization)

 block_3_expand_relu (ReLU)     (None, 38, 38, 144)  0           ['block_3_exp
and_BN[0][0]']

 block_3_pad (ZeroPadding2D)    (None, 39, 39, 144)  0           ['block_3_exp
and_relu[0][0]']

 block_3_depthwise (DepthwiseCo  (None, 19, 19, 144)  1296       ['block_3_pad
[0][0]']
 nv2D)
 block_3_depthwise_BN (BatchNor  (None, 19, 19, 144)  576         ['block_3_dep
thwise[0][0]']
 malization)

 block_3_depthwise_relu (ReLU)  (None, 19, 19, 144)  0           ['block_3_dep
thwise_BN[0][0]']

 block_3_project (Conv2D)       (None, 19, 19, 32)   4608        ['block_3_dep
thwise_relu[0][0]']

 block_3_project_BN (BatchNorma  (None, 19, 19, 32)  128         ['block_3_pro
ject[0][0]']
 lization)

 block_4_expand (Conv2D)        (None, 19, 19, 192)  6144        ['block_3_pro
ject_BN[0][0]']

 block_4_expand_BN (BatchNormal  (None, 19, 19, 192)  768         ['block_4_exp
and[0][0]']
 ization)

 block_4_expand_relu (ReLU)     (None, 19, 19, 192)  0           ['block_4_exp
and_BN[0][0]']

 block_4_depthwise (DepthwiseCo  (None, 19, 19, 192)  1728        ['block_4_exp
and_relu[0][0]']
 nv2D)

 block_4_depthwise_BN (BatchNor  (None, 19, 19, 192)  768         ['block_4_dep
thwise[0][0]']
 malization)

 block_4_depthwise_relu (ReLU)  (None, 19, 19, 192)  0           ['block_4_dep
thwise_BN[0][0]']

 block_4_project (Conv2D)       (None, 19, 19, 32)   6144        ['block_4_dep
thwise_relu[0][0]']

 block_4_project_BN (BatchNorma  (None, 19, 19, 32)  128         ['block_4_pro
ject[0][0]']
 lization)

 block_4_add (Add)              (None, 19, 19, 32)   0           ['block_3_pro
ject_BN[0][0]',
                                                                  'block_4_pro
ject_BN[0][0]']

 block_5_expand (Conv2D)        (None, 19, 19, 192)  6144        ['block_4_add
[0][0]']

 block_5_expand_BN (BatchNormal  (None, 19, 19, 192)  768         ['block_5_exp
and[0][0]']
 ization)

 block_5_expand_relu (ReLU)     (None, 19, 19, 192)  0           ['block_5_exp
and_BN[0][0]']

 block_5_depthwise (DepthwiseCo  (None, 19, 19, 192)  1728        ['block_5_exp
and_relu[0][0]']
 nv2D)
 block_5_depthwise_BN (BatchNor  (None, 19, 19, 192)  768         ['block_5_dep
thwise[0][0]']
 malization)

 block_5_depthwise_relu (ReLU)  (None, 19, 19, 192)  0           ['block_5_dep
thwise_BN[0][0]']

 block_5_project (Conv2D)       (None, 19, 19, 32)   6144        ['block_5_dep
thwise_relu[0][0]']

 block_5_project_BN (BatchNorma  (None, 19, 19, 32)  128         ['block_5_pro
ject[0][0]']
 lization)

 block_5_add (Add)              (None, 19, 19, 32)   0           ['block_4_add
[0][0]',
                                                                  'block_5_pro
ject_BN[0][0]']

 block_6_expand (Conv2D)        (None, 19, 19, 192)  6144        ['block_5_add
[0][0]']

 block_6_expand_BN (BatchNormal  (None, 19, 19, 192)  768         ['block_6_exp
and[0][0]']
 ization)

 block_6_expand_relu (ReLU)     (None, 19, 19, 192)  0           ['block_6_exp
and_BN[0][0]']

 block_6_pad (ZeroPadding2D)    (None, 21, 21, 192)  0           ['block_6_exp
and_relu[0][0]']

 block_6_depthwise (DepthwiseCo  (None, 10, 10, 192)  1728        ['block_6_pad
[0][0]']
 nv2D)

 block_6_depthwise_BN (BatchNor  (None, 10, 10, 192)  768         ['block_6_dep
thwise[0][0]']
 malization)

 block_6_depthwise_relu (ReLU)  (None, 10, 10, 192)  0           ['block_6_dep
thwise_BN[0][0]']

 block_6_project (Conv2D)       (None, 10, 10, 64)   12288       ['block_6_dep
thwise_relu[0][0]']

 block_6_project_BN (BatchNorma  (None, 10, 10, 64)  256         ['block_6_pro
ject[0][0]']
 lization)

 block_7_expand (Conv2D)        (None, 10, 10, 384)  24576       ['block_6_pro
ject_BN[0][0]']

 block_7_expand_BN (BatchNormal  (None, 10, 10, 384)  1536        ['block_7_exp
and[0][0]']
 ization)

 block_7_expand_relu (ReLU)     (None, 10, 10, 384)  0           ['block_7_exp
and_BN[0][0]']
```

```
 block_7_depthwise (DepthwiseCo  (None, 10, 10, 384)  3456       ['block_7_exp
and_relu[0][0]']
 nv2D)

 block_7_depthwise_BN (BatchNor  (None, 10, 10, 384)  1536       ['block_7_dep
thwise[0][0]']
 malization)

 block_7_depthwise_relu (ReLU)  (None, 10, 10, 384)  0          ['block_7_dep
thwise_BN[0][0]']

 block_7_project (Conv2D)       (None, 10, 10, 64)   24576      ['block_7_dep
thwise_relu[0][0]']

 block_7_project_BN (BatchNorma  (None, 10, 10, 64)  256        ['block_7_pro
ject[0][0]']
 lization)

 block_7_add (Add)              (None, 10, 10, 64)   0          ['block_6_pro
ject_BN[0][0]',
                                                                 'block_7_pro
ject_BN[0][0]']

 block_8_expand (Conv2D)        (None, 10, 10, 384)  24576      ['block_7_add
[0][0]']

 block_8_expand_BN (BatchNormal  (None, 10, 10, 384)  1536       ['block_8_exp
and[0][0]']
 ization)

 block_8_expand_relu (ReLU)     (None, 10, 10, 384)  0          ['block_8_exp
and_BN[0][0]']

 block_8_depthwise (DepthwiseCo  (None, 10, 10, 384)  3456       ['block_8_exp
and_relu[0][0]']
 nv2D)

 block_8_depthwise_BN (BatchNor  (None, 10, 10, 384)  1536       ['block_8_dep
thwise[0][0]']
 malization)

 block_8_depthwise_relu (ReLU)  (None, 10, 10, 384)  0          ['block_8_dep
thwise_BN[0][0]']

 block_8_project (Conv2D)       (None, 10, 10, 64)   24576      ['block_8_dep
thwise_relu[0][0]']

 block_8_project_BN (BatchNorma  (None, 10, 10, 64)  256        ['block_8_pro
ject[0][0]']
 lization)

 block_8_add (Add)              (None, 10, 10, 64)   0          ['block_7_add
[0][0]',
                                                                 'block_8_pro
ject_BN[0][0]']

 block_9_expand (Conv2D)        (None, 10, 10, 384)  24576      ['block_8_add
[0][0]']
```

```
 block_9_expand_BN (BatchNormal  (None, 10, 10, 384)  1536        ['block_9_exp
and[0][0]']
 ization)

 block_9_expand_relu (ReLU)     (None, 10, 10, 384)  0           ['block_9_exp
and_BN[0][0]']

 block_9_depthwise (DepthwiseCo  (None, 10, 10, 384)  3456        ['block_9_exp
and_relu[0][0]']
 nv2D)

 block_9_depthwise_BN (BatchNor  (None, 10, 10, 384)  1536        ['block_9_dep
thwise[0][0]']
 malization)

 block_9_depthwise_relu (ReLU)  (None, 10, 10, 384)  0           ['block_9_dep
thwise_BN[0][0]']

 block_9_project (Conv2D)       (None, 10, 10, 64)   24576       ['block_9_dep
thwise_relu[0][0]']

 block_9_project_BN (BatchNorma  (None, 10, 10, 64)  256         ['block_9_pro
ject[0][0]']
 lization)

 block_9_add (Add)              (None, 10, 10, 64)   0           ['block_8_add
[0][0]',
                                                                  'block_9_pro
ject_BN[0][0]']

 block_10_expand (Conv2D)       (None, 10, 10, 384)  24576       ['block_9_add
[0][0]']

 block_10_expand_BN (BatchNorma  (None, 10, 10, 384)  1536        ['block_10_ex
pand[0][0]']
 lization)

 block_10_expand_relu (ReLU)    (None, 10, 10, 384)  0           ['block_10_ex
pand_BN[0][0]']

 block_10_depthwise (DepthwiseC  (None, 10, 10, 384)  3456        ['block_10_ex
pand_relu[0][0]']
 onv2D)

 block_10_depthwise_BN (BatchNo  (None, 10, 10, 384)  1536        ['block_10_de
pthwise[0][0]']
 rmalization)

 block_10_depthwise_relu (ReLU)  (None, 10, 10, 384)  0           ['block_10_de
pthwise_BN[0][0]']

 block_10_project (Conv2D)      (None, 10, 10, 96)   36864       ['block_10_de
pthwise_relu[0][0]']

 block_10_project_BN (BatchNorm  (None, 10, 10, 96)  384         ['block_10_pr
oject[0][0]']
 alization)

 block_11_expand (Conv2D)       (None, 10, 10, 576)  55296       ['block_10_pr
oject_BN[0][0]']
```

```
 block_11_expand_BN (BatchNorma  (None, 10, 10, 576)  2304        ['block_11_ex
pand[0][0]']
 lization)

 block_11_expand_relu (ReLU)    (None, 10, 10, 576)  0           ['block_11_ex
pand_BN[0][0]']

 block_11_depthwise (DepthwiseC  (None, 10, 10, 576)  5184        ['block_11_ex
pand_relu[0][0]']
 onv2D)

 block_11_depthwise_BN (BatchNo  (None, 10, 10, 576)  2304        ['block_11_de
pthwise[0][0]']
 rmalization)

 block_11_depthwise_relu (ReLU)  (None, 10, 10, 576)  0           ['block_11_de
pthwise_BN[0][0]']

 block_11_project (Conv2D)      (None, 10, 10, 96)   55296       ['block_11_de
pthwise_relu[0][0]']

 block_11_project_BN (BatchNorm  (None, 10, 10, 96)  384         ['block_11_pr
oject[0][0]']
 alization)

 block_11_add (Add)             (None, 10, 10, 96)   0           ['block_10_pr
oject_BN[0][0]',
                                                                  'block_11_pr
oject_BN[0][0]']

 block_12_expand (Conv2D)       (None, 10, 10, 576)  55296       ['block_11_ad
d[0][0]']

 block_12_expand_BN (BatchNorma  (None, 10, 10, 576)  2304        ['block_12_ex
pand[0][0]']
 lization)

 block_12_expand_relu (ReLU)    (None, 10, 10, 576)  0           ['block_12_ex
pand_BN[0][0]']

 block_12_depthwise (DepthwiseC  (None, 10, 10, 576)  5184        ['block_12_ex
pand_relu[0][0]']
 onv2D)

 block_12_depthwise_BN (BatchNo  (None, 10, 10, 576)  2304        ['block_12_de
pthwise[0][0]']
 rmalization)

 block_12_depthwise_relu (ReLU)  (None, 10, 10, 576)  0           ['block_12_de
pthwise_BN[0][0]']

 block_12_project (Conv2D)      (None, 10, 10, 96)   55296       ['block_12_de
pthwise_relu[0][0]']

 block_12_project_BN (BatchNorm  (None, 10, 10, 96)  384         ['block_12_pr
oject[0][0]']
 alization)

 block_12_add (Add)             (None, 10, 10, 96)   0           ['block_11_ad
```

```
d[0][0]',
                                                                  'block_12_pr
oject_BN[0][0]']

 block_13_expand (Conv2D)       (None, 10, 10, 576)  55296       ['block_12_ad
d[0][0]']

 block_13_expand_BN (BatchNorma  (None, 10, 10, 576)  2304       ['block_13_ex
pand[0][0]']
 lization)

 block_13_expand_relu (ReLU)    (None, 10, 10, 576)  0           ['block_13_ex
pand_BN[0][0]']

 block_13_pad (ZeroPadding2D)   (None, 11, 11, 576)  0           ['block_13_ex
pand_relu[0][0]']

 block_13_depthwise (DepthwiseC  (None, 5, 5, 576)   5184        ['block_13_pa
d[0][0]']
 onv2D)

 block_13_depthwise_BN (BatchNo  (None, 5, 5, 576)   2304        ['block_13_de
pthwise[0][0]']
 rmalization)

 block_13_depthwise_relu (ReLU)  (None, 5, 5, 576)   0           ['block_13_de
pthwise_BN[0][0]']

 block_13_project (Conv2D)      (None, 5, 5, 160)    92160       ['block_13_de
pthwise_relu[0][0]']

 block_13_project_BN (BatchNorm  (None, 5, 5, 160)   640         ['block_13_pr
oject[0][0]']
 alization)

 block_14_expand (Conv2D)       (None, 5, 5, 960)    153600      ['block_13_pr
oject_BN[0][0]']

 block_14_expand_BN (BatchNorma  (None, 5, 5, 960)   3840        ['block_14_ex
pand[0][0]']
 lization)

 block_14_expand_relu (ReLU)    (None, 5, 5, 960)    0           ['block_14_ex
pand_BN[0][0]']

 block_14_depthwise (DepthwiseC  (None, 5, 5, 960)   8640        ['block_14_ex
pand_relu[0][0]']
 onv2D)

 block_14_depthwise_BN (BatchNo  (None, 5, 5, 960)   3840        ['block_14_de
pthwise[0][0]']
 rmalization)

 block_14_depthwise_relu (ReLU)  (None, 5, 5, 960)   0           ['block_14_de
pthwise_BN[0][0]']

 block_14_project (Conv2D)      (None, 5, 5, 160)    153600      ['block_14_de
pthwise_relu[0][0]']

 block_14_project_BN (BatchNorm  (None, 5, 5, 160)   640         ['block_14_pr
oject[0][0]']
 alization)

 block_14_add (Add)             (None, 5, 5, 160)    0           ['block_13_pr
oject_BN[0][0]',
                                                                  'block_14_pr
oject_BN[0][0]']

 block_15_expand (Conv2D)       (None, 5, 5, 960)    153600      ['block_14_ad
d[0][0]']

 block_15_expand_BN (BatchNorma  (None, 5, 5, 960)   3840        ['block_15_ex
pand[0][0]']
 lization)

 block_15_expand_relu (ReLU)    (None, 5, 5, 960)    0           ['block_15_ex
pand_BN[0][0]']

 block_15_depthwise (DepthwiseC  (None, 5, 5, 960)   8640        ['block_15_ex
pand_relu[0][0]']
 onv2D)

 block_15_depthwise_BN (BatchNo  (None, 5, 5, 960)   3840        ['block_15_de
pthwise[0][0]']
 rmalization)

 block_15_depthwise_relu (ReLU)  (None, 5, 5, 960)   0           ['block_15_de
pthwise_BN[0][0]']

 block_15_project (Conv2D)      (None, 5, 5, 160)    153600      ['block_15_de
pthwise_relu[0][0]']

 block_15_project_BN (BatchNorm  (None, 5, 5, 160)   640         ['block_15_pr
oject[0][0]']
 alization)

 block_15_add (Add)             (None, 5, 5, 160)    0           ['block_14_ad
d[0][0]',
                                                                  'block_15_pr
oject_BN[0][0]']

 block_16_expand (Conv2D)       (None, 5, 5, 960)    153600      ['block_15_ad
d[0][0]']

 block_16_expand_BN (BatchNorma  (None, 5, 5, 960)   3840        ['block_16_ex
pand[0][0]']
 lization)

 block_16_expand_relu (ReLU)    (None, 5, 5, 960)    0           ['block_16_ex
pand_BN[0][0]']

 block_16_depthwise (DepthwiseC  (None, 5, 5, 960)   8640        ['block_16_ex
pand_relu[0][0]']
 onv2D)

 block_16_depthwise_BN (BatchNo  (None, 5, 5, 960)   3840        ['block_16_de
pthwise[0][0]']
 rmalization)

 block_16_depthwise_relu (ReLU)  (None, 5, 5, 960)   0           ['block_16_de
```

```
pthwise_BN[0][0]']

 block_16_project (Conv2D)      (None, 5, 5, 320)    307200      ['block_16_de
pthwise_relu[0][0]']

 block_16_project_BN (BatchNorm  (None, 5, 5, 320)   1280        ['block_16_pr
oject[0][0]']
 alization)

 Conv_1 (Conv2D)                (None, 5, 5, 1280)   409600      ['block_16_pr
oject_BN[0][0]']

 Conv_1_bn (BatchNormalization)  (None, 5, 5, 1280)  5120        ['Conv_1[0]
[0]']

 out_relu (ReLU)                (None, 5, 5, 1280)   0           ['Conv_1_bn
[0][0]']

==============================================================================
====================
Total params: 2,257,984
Trainable params: 0
Non-trainable params: 2,257,984
______________________________________________________________________________
____________________
```

In [26]:
```
# manipulating data to make model more accurate.
prediction_layer = tf.keras.layers.Dense(1)
```

Transfer Learning

We will be using the previously built MobileNet model, but will add a new classifier, which will be trained from scratch, so that we can repurpose the feature maps learned previously for the dataset.

In [27]:
```
inputs = tf.keras.Input(shape=(150, 150, 3))
x = data_augmentation(inputs)
x = preprocess_input(x)
x = base_model(x, training=False)
outputs = prediction_layer(x)
model = tf.keras.Model(inputs, outputs)
```

In [28]:
```
base_learning_rate = 0.0001
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=base_learning_ra
              loss=tf.keras.losses.BinaryCrossentropy(from_logits=True),
              metrics=['accuracy'])
```

In [ ]:
```
history = model.fit(train_images,
                    epochs=25)
```

The results here were not great, but that very well could be due to the image set that we used rather than the model itself. I think if I ran with more epochs it would come out with a higher accuracy, but when I attempted to do more it crashed my machine.

Analysis

Our CNN architecture showed enough positive signs for us to determine it was the best model. Our accuracy started low, but once we go to the later epochs, our accuracy continued to increase to an incredibely high value. The model seems to not be overfitting as it is not memorizing the training data.

We also used the MobileNet V2 pretrained model and used to transfer learning to fine tune the pre-trained model on our new data. This was not very successful for us with a much lower accuracy than our previous architectures that we attempted. A big part of this might be the pretrained model that we decided to use and how decided to fine tune that model. Though, it is nice to have all the features all built in when loaded in, which saves a lot of time and resources which makes this method stil the most practical.

Unless you have a very specific dataset, I think its best to build your own model. This may not be as efficent as building off a pretrained model, but building your own will allow you . Obviously the downside to building your own model is that it is more prone to error, unless you use an incredibly large dataset, which would require a particularly power machine to run.