Task: Perform the PET classification model using Transfer Learning principles employing VGG-Mobile Net or AlexNet model or any other pre-trained model of your choice.

Objective : To provide coding experience of Real Time Image Recognition using Pre-Trained models actually employed in the real world.

Domain :Computer Vision

Importing Libraries

In [1]:
```
import numpy as np
import tensorflow as tf
import tensorflow as keras
import matplotlib.pyplot as plt
from tensorflow.keras import datasets,layers,models
```

In [2]:
```
import os
import glob
```

In [3]:
```
import PIL
import matplotlib.image as mpimg
```

In [4]:
```
import cv2
import random
```

In [5]:
```
!pip install --upgrade tensorflow_hub
```

```
Collecting tensorflow_hub
  Downloading tensorflow_hub-0.12.0-py2.py3-none-any.whl (108 kB)
Requirement already satisfied, skipping upgrade: numpy>=1.12.0 in c:\users\utkar\anaconda3\lib\site-packages (fro
m tensorflow_hub) (1.19.2)
Requirement already satisfied, skipping upgrade: protobuf>=3.8.0 in c:\users\utkar\anaconda3\lib\site-packages (f
rom tensorflow_hub) (3.15.5)
Requirement already satisfied, skipping upgrade: six>=1.9 in c:\users\utkar\anaconda3\lib\site-packages (from pro
tobuf>=3.8.0->tensorflow_hub) (1.15.0)
Installing collected packages: tensorflow-hub
  Attempting uninstall: tensorflow-hub
    Found existing installation: tensorflow-hub 0.9.0
    Uninstalling tensorflow-hub-0.9.0:
      Successfully uninstalled tensorflow-hub-0.9.0
Successfully installed tensorflow-hub-0.12.0
```

```
ERROR: After October 2020 you may experience errors when installing or updating packages. This is because pip wil
l change the way that it resolves dependency conflicts.

We recommend you use --use-feature=2020-resolver to test your packages with the new resolver before it becomes th
e default.

tensorflowjs 3.6.0 requires tensorflow-hub<0.10,>=0.7.0, but you'll have tensorflow-hub 0.12.0 which is incompati
ble.
```

Specifying folder directory for training images

In [6]:
```
train_folder = r'C:\Users\utkar\OneDrive\Desktop\Machine Learning\catsndogs\data\train'
```

In [7]:
```
file = random.choice(os.listdir(train_folder))
image_path= os.path.join(train_folder, file)
img=cv2.imread(image_path)
```

In [8]:
```
type(img)
```

Out[8]:
```
NoneType
```

In [43]:
```
IMG_WIDTH=224
IMG_HEIGHT=224
img_folder= r'C:\Users\utkar\OneDrive\Desktop\Machine Learning\catsndogs\data\train'
```

Creating Dataset while resizing the images using open-cv library and also creating the labels

In [44]:
```
def create_dataset(img_folder):
```

```
    img_data_array=[]
    class_name=[]

    for dir1 in os.listdir(img_folder):
        for file in os.listdir(os.path.join(img_folder, dir1)):

            image_path= os.path.join(img_folder, dir1,  file)
            image= cv2.imread( image_path, cv2.COLOR_BGR2RGB)
            image=cv2.resize(image, (IMG_HEIGHT, IMG_WIDTH),interpolation = cv2.INTER_AREA)
            image=np.array(image)
            image = image.astype('float32')
            image /= 255
            img_data_array.append(image)
            class_name.append(dir1)
    return img_data_array, class_name
# extract the image array and class name
X, y =create_dataset(r'C:\Users\utkar\OneDrive\Desktop\Machine Learning\catsndogs\data\train')
```

In [45]:
```
IMG_WIDTH=224
IMG_HEIGHT=224
img_folder_test = r'C:\Users\utkar\OneDrive\Desktop\Machine Learning\catsndogs\data\test'
```

Same operation for test folder images

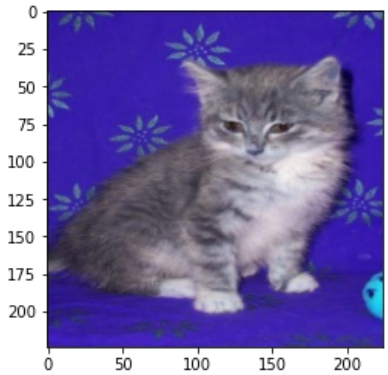
In [46]:
```
def create_dataset1(img_folder_test):

    img_data_array=[]
    class_name=[]

    for dir1 in os.listdir(img_folder_test):
        for file in os.listdir(os.path.join(img_folder_test, dir1)):

            image_path= os.path.join(img_folder_test, dir1,  file)
            image= cv2.imread( image_path, cv2.COLOR_BGR2RGB)
            image=cv2.resize(image, (IMG_HEIGHT, IMG_WIDTH),interpolation = cv2.INTER_AREA)
            image=np.array(image)
            image = image.astype('float32')
            image /= 255
            img_data_array.append(image)
            class_name.append(dir1)
    return img_data_array, class_name
# extract the image array and class name
X_test, y_test =create_dataset1(r'C:\Users\utkar\OneDrive\Desktop\Machine Learning\catsndogs\data\test')
```

In [47]:
```
type(X[0])
```

Out[47]:
```
numpy.ndarray
```

In [48]:
```
plt.imshow(X[34])
```

Out[48]:
```
<matplotlib.image.AxesImage at 0x207ad67cc10>
```



Converting lists into numpy array

In [49]:
```
X = np.array(X)
y = np.array(y)
```

Using label encoder for standardizing and labelling cat = 0 and dog = 1

In [50]:
```
#Import library:
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
#New variable for outlet
```

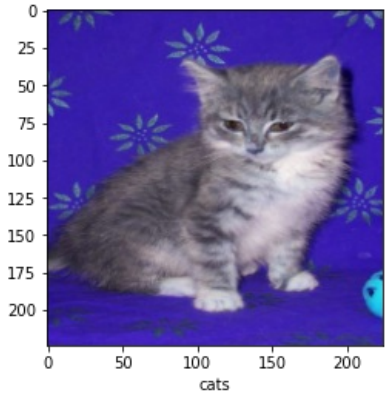
```
y = le.fit_transform(y)
le = LabelEncoder()
for i in y:
    y = le.fit_transform(y)
```

Function defined to plot image with label as well

In [51]:
```
def plt_show(X,y,index):
    plt.imshow(X[index])
    plt.xlabel(classes[y[index]])
```

In [52]:
```
classes = ["cats", "dogs"]
```

In [53]:
```
plt_show(X,y,34)
```



In [54]:
```
X = np.array(X)
y = np.array(y)
len(X)
```

Out[54]: 1642

In [55]:
```
len(y)
```

Out[55]: 1642

In [56]:
```
len(X)
```

Out[56]: 1642

In [57]:
```
y
```

Out[57]: array([0, 0, 0, ..., 1, 1, 1], dtype=int64)

We have used a pretrained model of mobilenet, using transfer learning

We use loss function as binary crossentropy since we have only two objects to detect.

In [58]:
```
mobilenet_Model = "https://tfhub.dev/google/tf2-preview/mobilenet_v2/feature_vector/4"
```

In [59]:
```
import tensorflow_hub as hub
```

In [64]:
```
pretrained_model_without_top_layer= hub.KerasLayer(mobilenet_Model,input_shape=(224,224,3),trainable=False)
```

In [68]:
```
cnn=models.Sequential([
                       pretrained_model_without_top_layer,
                       layers.Dense(1,activation='sigmoid')])
```

In [69]:
```
cnn.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])
```

In [70]:
```
cnn.fit(X,y,epochs=10)
```

```
Epoch 1/10
52/52 [==============================] - 51s 828ms/step - loss: 0.3900 - accuracy: 0.8242
```

```
Epoch 2/10
52/52 [==============================] - 43s 828ms/step - loss: 0.0894 - accuracy: 0.9748
Epoch 3/10
52/52 [==============================] - 41s 795ms/step - loss: 0.0814 - accuracy: 0.9776
Epoch 4/10
52/52 [==============================] - 42s 803ms/step - loss: 0.0464 - accuracy: 0.9893
Epoch 5/10
52/52 [==============================] - 43s 824ms/step - loss: 0.0519 - accuracy: 0.9897
Epoch 6/10
52/52 [==============================] - 43s 829ms/step - loss: 0.0370 - accuracy: 0.9911
Epoch 7/10
52/52 [==============================] - 44s 843ms/step - loss: 0.0340 - accuracy: 0.9935
Epoch 8/10
52/52 [==============================] - 48s 922ms/step - loss: 0.0315 - accuracy: 0.9944
Epoch 9/10
52/52 [==============================] - 52s 991ms/step - loss: 0.0293 - accuracy: 0.9964
Epoch 10/10
52/52 [==============================] - 48s 917ms/step - loss: 0.0284 - accuracy: 0.9935
```

Out[70]:
```
<tensorflow.python.keras.callbacks.History at 0x207aae62f70>
```

In [71]:
```
X_test = np.array(X_test)
y_test = np.array(y_test)
```

In [72]:
```
X_test = tf.convert_to_tensor(X_test)
y_test = tf.convert_to_tensor(y_test)
```

Using label encoder for standardizing and labelling cat = 0 and dog = 1 for test dataset as well

In [73]:
```
#Import library:
from sklearn.preprocessing import LabelEncoder, OneHotEncoder
le = LabelEncoder()
#New variable for outlet
y_test = le.fit_transform(y_test)
le = LabelEncoder()
for i in y:
    y_test = le.fit_transform(y_test)
```

In [74]:
```
y_test
```

Out[74]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1],
      dtype=int64)
```

We get an accuracy score of 1.00 after evaluating

In [75]:
```
cnn.evaluate(X_test,y_test)
```

```
1/1 [==============================] - 2s 2s/step - loss: 0.0380 - accuracy: 1.0000
```

Out[75]:
```
[0.038002051413059235, 1.0]
```

In [ ]: