Assignment 6

1. Name: Vaibhav Bichave
2. Batch: P-10
3. Roll No.: 43209

Problem Statement :

Object detection using Transfer Learning of CNN architectures

Importing the libraries

In [1]:

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

Out[1]:

```
'2.8.0'
```

Preprocessing for dataset

In [2]:

```
img_generator = tf.keras.preprocessing.image.ImageDataGenerator(#rotation_range=90,
                                                                brightness_range=(0
                                                                #shear_range=0.2,
                                                                #zoom_range=0.2,
                                                                channel_shift_range
                                                                horizontal_flip=Tru
                                                                vertical_flip=True,
                                                                rescale=1./255,
                                                                validation_split=0.
```

In [3]:

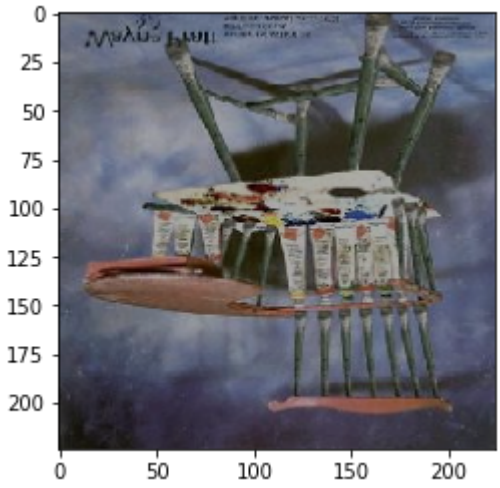
```
root_dir = '101_ObjectCategories'
img_generator_flow_train = img_generator.flow_from_directory(
    directory=root_dir,
    target_size=(224, 224),
    batch_size=32,
    shuffle=True,
    subset="training")

img_generator_flow_valid = img_generator.flow_from_directory(
    directory=root_dir,
    target_size=(224, 224),
    batch_size=32,
    shuffle=True,
    subset="validation")
```

```
Found 6444 images belonging to 102 classes.
Found 2700 images belonging to 102 classes.
```

In [4]:

```
imgs, labels = next(iter(img_generator_flow_train))
for img, label in zip(imgs, labels):
    plt.imshow(img)
    plt.show()
```



a. Load in a pretrained model (InceptionV3)

In [5]:

```
base_model = tf.keras.applications.InceptionV3(input_shape=(224,224,3),
                                               include_top=False,
                                               weights = "imagenet"
                                               )
```

b. Freeze parameters (weights) in model's lower convolutional layers

In [6]:

```
base_model.trainable = False
```

c. Add custom classifier with several layers of trainable parameters to model

In [7]:

```
model = tf.keras.Sequential([
    base_model,
    tf.keras.layers.MaxPooling2D(),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(102, activation="softmax")
])
```

In [8]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 inception_v3 (Functional)   (None, 5, 5, 2048)        21802784

 max_pooling2d_4 (MaxPooling  (None, 2, 2, 2048)       0
 2D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 102)               835686

=================================================================
Total params: 22,638,470
Trainable params: 835,686
Non-trainable params: 21,802,784
_________________________________________________________________
```

d. Train classifier layers on training data available for task

In [9]:

```
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate = 0.001),
              loss = tf.keras.losses.CategoricalCrossentropy(),
              metrics = [tf.keras.metrics.CategoricalAccuracy()])
```

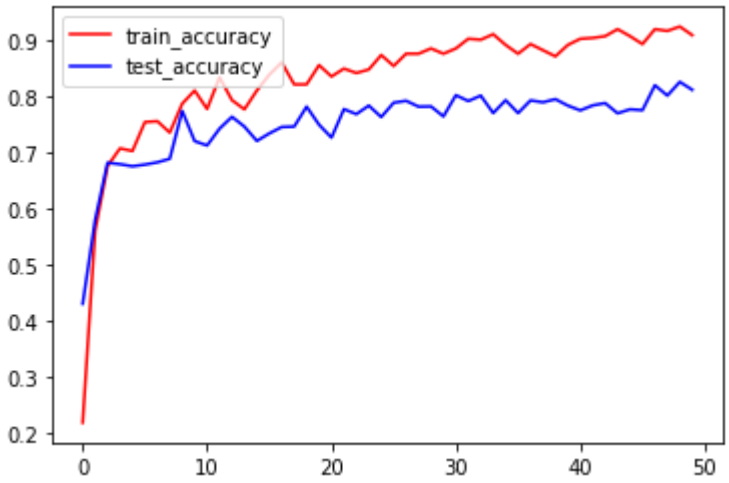
In [10]:

```
model.fit(img_generator_flow_train, validation_data=img_generator_flow_valid, steps
```

```
Epoch 1/50
20/20 [==============================] - 299s 15s/step - loss: 9.578
0 - categorical_accuracy: 0.2188 - val_loss: 4.9710 - val_categorica
l_accuracy: 0.4307
Epoch 2/50
20/20 [==============================] - 281s 15s/step - loss: 3.323
5 - categorical_accuracy: 0.5609 - val_loss: 2.7433 - val_categorica
l_accuracy: 0.5800
Epoch 3/50
20/20 [==============================] - 277s 14s/step - loss: 2.033
5 - categorical_accuracy: 0.6766 - val_loss: 2.1982 - val_categorica
l_accuracy: 0.6822
Epoch 4/50
20/20 [==============================] - 280s 15s/step - loss: 2.223
7 - categorical_accuracy: 0.7078 - val_loss: 2.1987 - val_categorica
l_accuracy: 0.6796
Epoch 5/50
20/20 [==============================] - 280s 15s/step - loss: 1.995
8 - categorical_accuracy: 0.7031 - val_loss: 2.3371 - val_categorica
```

In [11]:

```
# Visualise train / Valid Accuracy
plt.plot(model.history.history["categorical_accuracy"], c="r", label="train_accurac
plt.plot(model.history.history["val_categorical_accuracy"], c="b", label="test_accu
plt.legend(loc="upper left")
plt.show()
```



e. Fine-tune hyper parameters and unfreeze more layers as needed

In [12]:

```
base_model.trainable = True

model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate = 0.001),
              loss = tf.keras.losses.CategoricalCrossentropy(),
              metrics = [tf.keras.metrics.CategoricalAccuracy()])
```

In [ ]:

```
model.fit(img_generator_flow_train, validation_data=img_generator_flow_valid, steps
```

```
Epoch 1/50
20/20 [==============================] - 474s 24s/step - loss: 3.9578
- categorical_accuracy: 0.4359 - val_loss: 126.2205 - val_categorical_
accuracy: 0.0163
Epoch 2/50
20/20 [==============================] - 466s 24s/step - loss: 3.9319
- categorical_accuracy: 0.2875 - val_loss: 13552.6338 - val_categorica
l_accuracy: 0.0481
Epoch 3/50
20/20 [==============================] - 462s 24s/step - loss: 3.5396
- categorical_accuracy: 0.3250 - val_loss: 65.9670 - val_categorical_a
ccuracy: 0.0663
Epoch 4/50
20/20 [==============================] - 461s 24s/step - loss: 3.1368
- categorical_accuracy: 0.3812 - val_loss: 62.4981 - val_categorical_a
ccuracy: 0.0722
Epoch 5/50
20/20 [==============================] - 463s 24s/step - loss: 2.8024
- categorical_accuracy: 0.4109 - val_loss: 4.9331 - val_categorical_ac
curacy: 0.2341
Epoch 6/50
20/20 [==============================] - 457s 23s/step - loss: 2.5176
- categorical_accuracy: 0.4328 - val_loss: 4.8107 - val_categorical_ac
curacy: 0.2370
Epoch 7/50
20/20 [==============================] - 459s 23s/step - loss: 2.0832
- categorical_accuracy: 0.5250 - val_loss: 3.3150 - val_categorical_ac
curacy: 0.4315
Epoch 8/50
20/20 [==============================] - 458s 24s/step - loss: 1.6358
- categorical_accuracy: 0.6047 - val_loss: 2.4610 - val_categorical_ac
curacy: 0.4459
Epoch 9/50
20/20 [==============================] - 460s 24s/step - loss: 1.6275
- categorical_accuracy: 0.5953 - val_loss: 2.1677 - val_categorical_ac
curacy: 0.5081
Epoch 10/50
20/20 [==============================] - ETA: 0s - loss: 1.6000 - cate
gorical_accuracy: 0.5938
```

In [ ]: