In [1]: ```python
!pip install tensorflow==2.1.0
```

In [2]: ```python
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from urllib.request import urlopen
from PIL import Image

%matplotlib inline
plt.style.use('default')

print("TF  Version",tf.__version__)
```

TF Version 2.17.0

In [3]: ```python
model_vgg=tf.keras.applications.vgg16.VGG16(include_top=True, weights='imagenet')
```

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/v
gg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5
553467096/553467096 ──────────────────── 4s 0us/step

In [4]: ```python
model_vgg.summary()
```

Model: "vgg16"

Layer (type)	Output Shape	
input_layer (InputLayer)	(None, 224, 224, 3)	
block1_conv1 (Conv2D)	(None, 224, 224, 64)	
block1_conv2 (Conv2D)	(None, 224, 224, 64)	
block1_pool (MaxPooling2D)	(None, 112, 112, 64)	
block2_conv1 (Conv2D)	(None, 112, 112, 128)	
block2_conv2 (Conv2D)	(None, 112, 112, 128)	
block2_pool (MaxPooling2D)	(None, 56, 56, 128)	
block3_conv1 (Conv2D)	(None, 56, 56, 256)	
block3_conv2 (Conv2D)	(None, 56, 56, 256)	
block3_conv3 (Conv2D)	(None, 56, 56, 256)	
block3_pool (MaxPooling2D)	(None, 28, 28, 256)	
block4_conv1 (Conv2D)	(None, 28, 28, 512)	
block4_conv2 (Conv2D)	(None, 28, 28, 512)	
block4_conv3 (Conv2D)	(None, 28, 28, 512)	
block4_pool (MaxPooling2D)	(None, 14, 14, 512)	
block5_conv1 (Conv2D)	(None, 14, 14, 512)	
block5_conv2 (Conv2D)	(None, 14, 14, 512)	
block5_conv3 (Conv2D)	(None, 14, 14, 512)	
block5_pool (MaxPooling2D)	(None, 7, 7, 512)	
flatten (Flatten)	(None, 25088)	
fc1 (Dense)	(None, 4096)	1(
fc2 (Dense)	(None, 4096)	1
predictions (Dense)	(None, 1000)	

**Total params:** 138,357,544 (527.79 MB)

Trainable params: 138,357,544 (527.79 MB)

Non-trainable params: 0 (0.00 B)

In [5]:
```python
def preprocess_input(img):
    x=np.zeros((224,224,3),dtype="float32")
    x[:,:,0]=img[:,:,2]
    x[:,:,1]=img[:,:,1]
    x[:,:,2]=img[:,:,0]
    mean = [103.939, 116.779, 123.68]
    x[:,:, 0] = x[:,:, 0]-mean[0]
    x[:,:, 1] = x[:,:, 1]-mean[1]
    x[:,:, 2] = x[:,:, 2]-mean[2]
    return x

def undo_preprocess_input(img):
    mean = [103.939, 116.779, 123.68]
    img[:,:, 0] = img[:,:, 0]+mean[0]
    img[:,:, 1] = img[:,:, 1]+mean[1]
    img[:,:, 2] = img[:,:, 2]+mean[2]
    x=np.zeros((224,224,3),dtype="float32")
    x[:,:,0]=img[:,:,2]
    x[:,:,1]=img[:,:,1]
    x[:,:,2]=img[:,:,0]
    return x
```

In [8]:
```python
img1 = (Image.open(urlopen("https://raw.githubusercontent.com/tensorchiefs/dl_book/
plt.imshow(img1)
plt.show()
new_width  = 224
new_height = 224
# Replacing Image.ANTIALIAS with Image.Resampling.LANCZOS
img1 = img1.resize((new_width, new_height), Image.Resampling.LANCZOS)
plt.imshow(img1)
plt.show()
img1=np.array(img1)
```
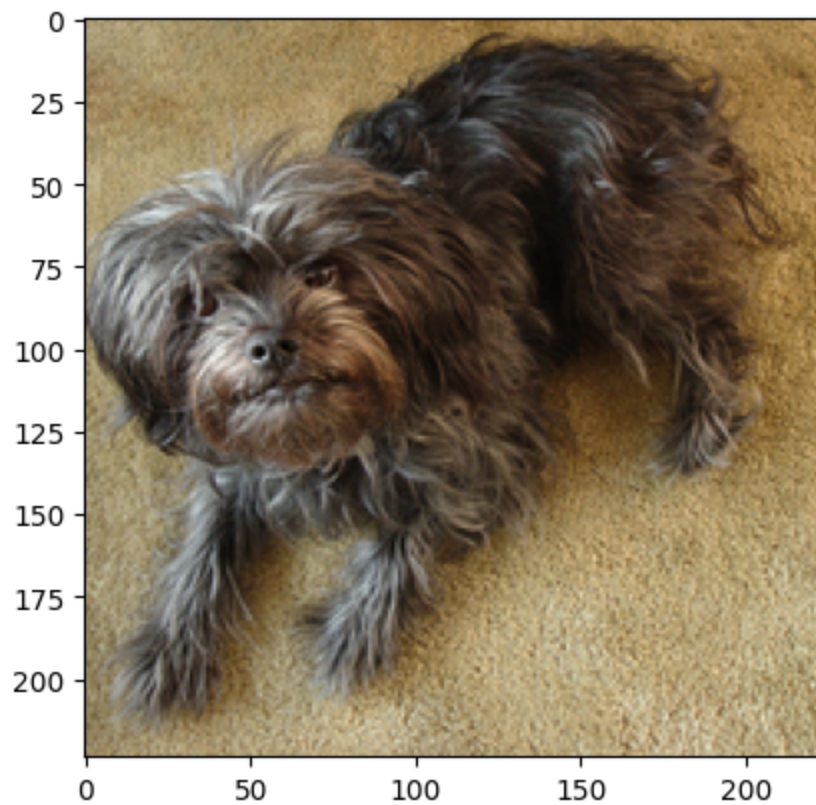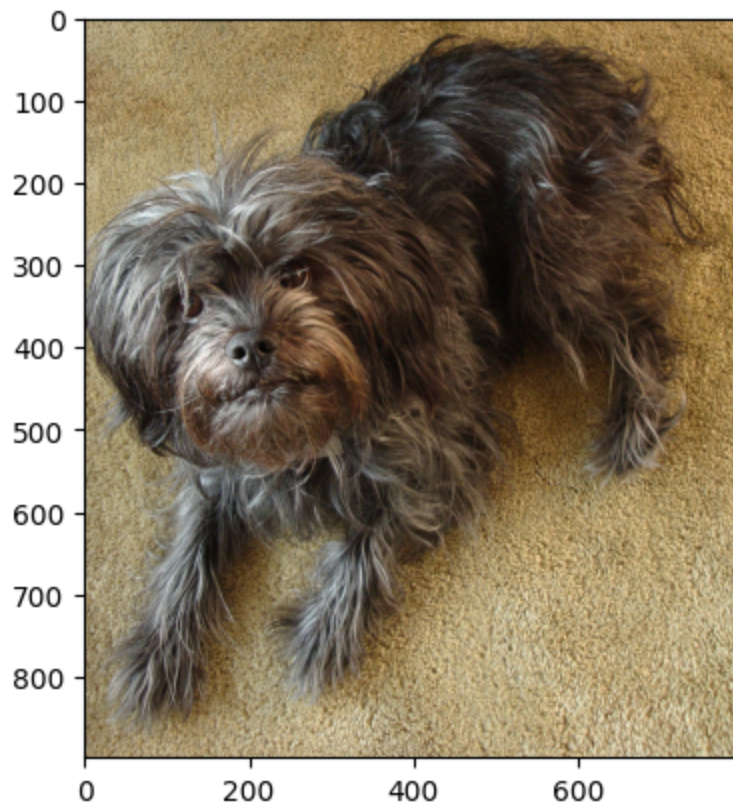
```
In [10]: img2 = (Image.open(urlopen("https://raw.githubusercontent.com/tensorchiefs/dl_book/
         plt.imshow(img2)
         plt.show()
         new_width  = 224
         new_height = 224
```

```python
# Replacing Image.ANTIALIAS with Image.Resampling.LANCZOS
img2 = img2.resize((new_width, new_height), Image.Resampling.LANCZOS)
plt.imshow(img2)
plt.show()
img2=np.array(img2)
```
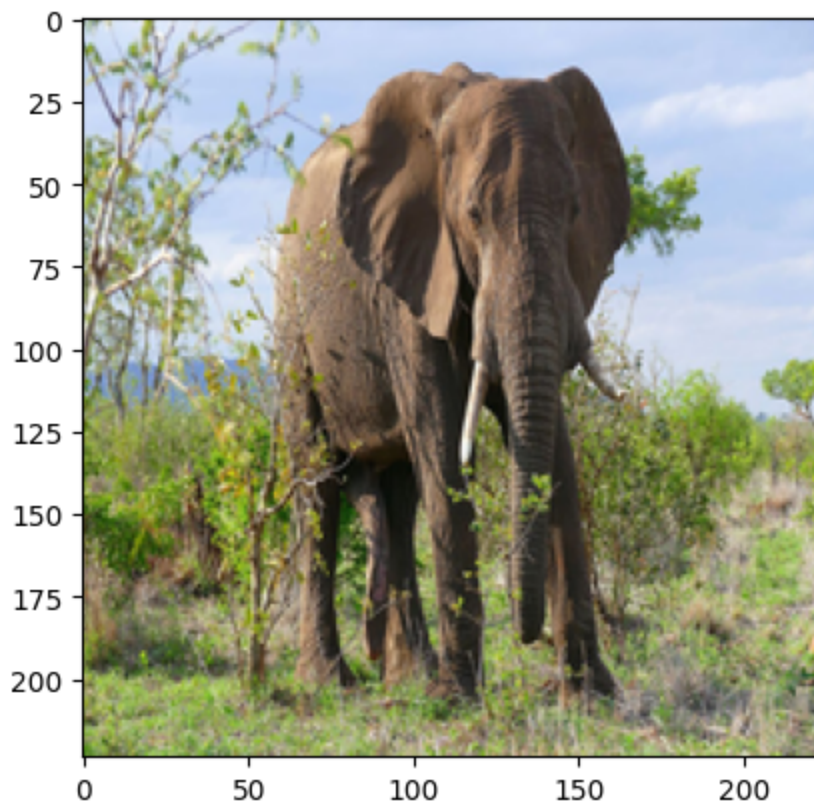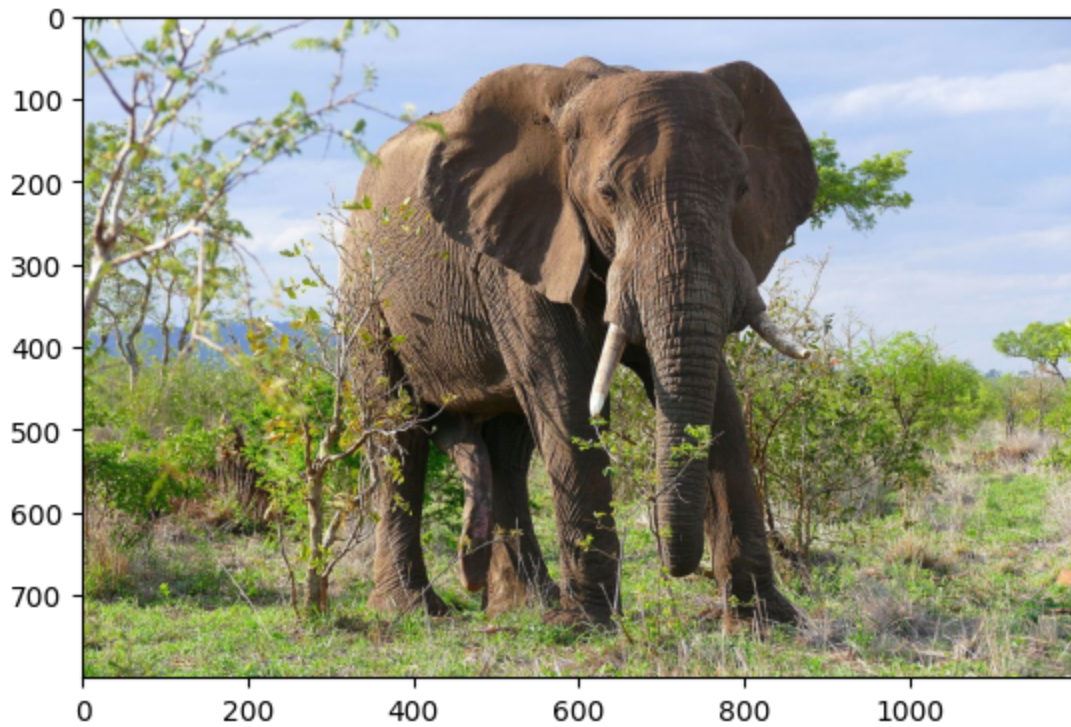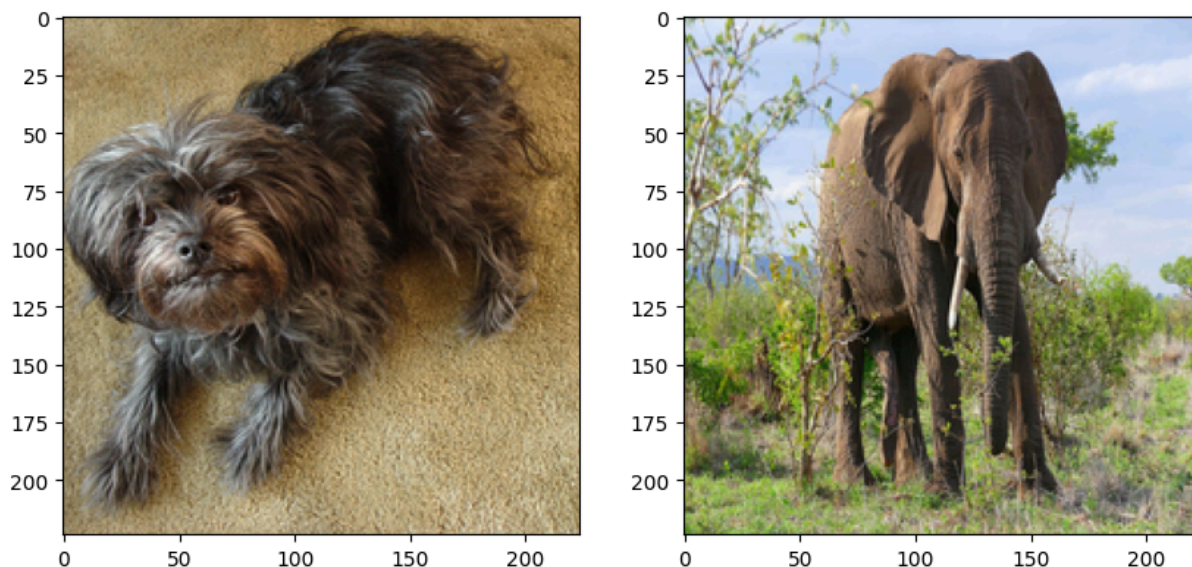


```python
In [11]:  plt.figure(figsize=(10,10))
          plt.subplot(1,2,1)
```

```
plt.imshow(img1)
plt.subplot(1,2,2)
plt.imshow(img2)
```

Out[11]: <matplotlib.image.AxesImage at 0x7e0130197a00>

In [12]:
```
img1=preprocess_input(img1)
print(img1.shape)
img2=preprocess_input(img2)
print(img2.shape)
```

```
(224, 224, 3)
(224, 224, 3)
```

In [13]:
```
pred1=model_vgg.predict(np.expand_dims(img1,axis=0))
tf.keras.applications.vgg16.decode_predictions(pred1)
```

1/1 ─────────────────── **1s** 947ms/step
Downloading data from https://storage.googleapis.com/download.tensorflow.org/data/im
agenet_class_index.json
35363/35363 ─────────────────── **0s** 0us/step

Out[13]: [[('n02110627', 'affenpinscher', 0.60996807),
          ('n02094433', 'Yorkshire_terrier', 0.115058616),
          ('n02097474', 'Tibetan_terrier', 0.06349397),
          ('n02086240', 'Shih-Tzu', 0.053632747),
          ('n02098413', 'Lhasa', 0.03696351)]]

In [14]:
```
pred2=model_vgg.predict(np.expand_dims(img2,axis=0))
tf.keras.applications.vgg16.decode_predictions(pred2)
```

1/1 ─────────────────── **1s** 539ms/step

Out[14]: [[('n01871265', 'tusker', 0.7851958),
          ('n02504458', 'African_elephant', 0.15363708),
          ('n02504013', 'Indian_elephant', 0.061132655),
          ('n02437312', 'Arabian_camel', 6.303232e-06),
          ('n02109047', 'Great_Dane', 5.9501795e-06)]]