NAME : C.ASHIKA

ROLL NO : 225229105

LAB - 13 IMAGE CLASSIFICATION USING PRE-TRAINED CNN MODELS

In [1]:
```
from keras.applications import VGG16
# Create VGG16 model with pre-trained weights
vgg_model = VGG16(weights='imagenet')
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5 (https://storage.googleapis.com/tensorflow/keras-applications/vgg16/vgg16_weights_tf_dim_ordering_tf_kernels.h5)
553467096/553467096 [==============================] - 496s 1us/step
```

In [2]:
```
# Display model summary
vgg_model.summary()
```

```
Model: "vgg16"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 224, 224, 3)]     0

 block1_conv1 (Conv2D)       (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)       (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)  (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)       (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)       (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)  (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)       (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)       (None, 56, 56, 256)       590080

 block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

 block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

 block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

 block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

 block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

 block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

 block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

 flatten (Flatten)           (None, 25088)             0

 fc1 (Dense)                 (None, 4096)              102764544

 fc2 (Dense)                 (None, 4096)              16781312

 predictions (Dense)         (None, 1000)              4097000

=================================================================
Total params: 138357544 (527.79 MB)
Trainable params: 138357544 (527.79 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

In [12]:
```
from PIL import Image
# Open the JPEG image
image = Image.open('my_photo.jpeg')

# Display the image using the default image viewer
image.show()
```

In [13]:
```
from keras.applications.vgg16 import VGG16, preprocess_input, decode_predic
import numpy as np
from PIL import Image
# Load the VGG16 model with pre-trained weights
vgg_model = VGG16(weights='imagenet')
# Load and preprocess your image
image_path = 'my_photo.jpeg'  # Replace with the actual path to your image
image = Image.open(image_path)
image = image.resize((224, 224))  # Resize to VGG16 input size
image_array = np.array(image)
expanded_image_array = np.expand_dims(image_array, axis=0)  # Add batch dim
preprocessed_image = preprocess_input(expanded_image_array)
# Make a prediction
predictions = vgg_model.predict(preprocessed_image)
# Decode and print the top predictions
decoded_predictions = decode_predictions(predictions, top=5)[0]  # Get top
for label, description, score in decoded_predictions:
    print(f"{description} ({label}): {score:.2f}")
```

```
1/1 [==============================] - 1s 696ms/step
jersey (n03595614): 0.03
ping-pong_ball (n03942813): 0.02
crutch (n03141823): 0.02
hammer (n03481172): 0.02
sunscreen (n04357314): 0.02
```

In [14]:
```
decoded_predictions = decode_predictions(predictions, top=10)[0]  # Get top
print("Top 10 Predictions:")
print("-------------------")
for i, (label, description, score) in enumerate(decoded_predictions, start=
    print(f"{i}. Predicted Class: {label}")
    print(f"   Name: {description}")
    print(f"   Probability: {score:.2f}")
    print("-------------------")
```

```
Top 10 Predictions:
-------------------
1. Predicted Class: n03595614
   Name: jersey
   Probability: 0.03
-------------------
2. Predicted Class: n03942813
   Name: ping-pong_ball
   Probability: 0.02
-------------------
3. Predicted Class: n03141823
   Name: crutch
   Probability: 0.02
-------------------
4. Predicted Class: n03481172
   Name: hammer
   Probability: 0.02
-------------------
5. Predicted Class: n04357314
   Name: sunscreen
   Probability: 0.02
-------------------
6. Predicted Class: n02786058
   Name: Band_Aid
   Probability: 0.01
-------------------
7. Predicted Class: n04350905
   Name: suit
   Probability: 0.01
-------------------
8. Predicted Class: n04370456
   Name: sweatshirt
   Probability: 0.01
-------------------
9. Predicted Class: n02865351
   Name: bolo_tie
   Probability: 0.01
-------------------
10. Predicted Class: n04039381
   Name: racket
   Probability: 0.01
-------------------
```

In [15]:
```
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.applications.resnet50 import preprocess_input, decode
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import numpy as np

# Load the ResNet50 model with pre-trained weights
resnet_model = ResNet50(weights='imagenet')

# Load and preprocess your image
image_path = 'my_photo.jpeg'  # Replace with the actual path to your image
image = load_img(image_path, target_size=(224, 224))  # ResNet50 requires i
image_array = img_to_array(image)
preprocessed_image = preprocess_input(np.expand_dims(image_array, axis=0))

# Make a prediction
predictions = resnet_model.predict(preprocessed_image)

# Decode and print the top 10 predictions
decoded_predictions = decode_predictions(predictions, top=10)[0]  # Get top
print("Top 10 Predictions using ResNet50:")
print("----------------------------------")
for i, (label, description, score) in enumerate(decoded_predictions, start=
    print(f"{i}. Predicted Class: {label}")
    print(f"   Name: {description}")
    print(f"   Probability: {score:.2f}")
    print("----------------------------------")
```

```
1/1 [==============================] - 3s 3s/step
Top 10 Predictions using ResNet50:
---------------------------------
1. Predicted Class: n02865351
   Name: bolo_tie
   Probability: 0.07
---------------------------------
2. Predicted Class: n03485407
   Name: hand-held_computer
   Probability: 0.05
---------------------------------
3. Predicted Class: n02786058
   Name: Band_Aid
   Probability: 0.03
---------------------------------
4. Predicted Class: n04355933
   Name: sunglass
   Probability: 0.03
---------------------------------
5. Predicted Class: n02787622
   Name: banjo
   Probability: 0.02
---------------------------------
6. Predicted Class: n03595614
   Name: jersey
   Probability: 0.02
---------------------------------
7. Predicted Class: n03838899
   Name: oboe
   Probability: 0.02
---------------------------------
8. Predicted Class: n03720891
   Name: maraca
   Probability: 0.02
---------------------------------
9. Predicted Class: n03942813
   Name: ping-pong_ball
   Probability: 0.02
---------------------------------
10. Predicted Class: n03141823
   Name: crutch
   Probability: 0.02
---------------------------------
```

In [ ]: