DLCV ASSIGNMENT 1

TEAM MEMBERS

DEVANG AGARWAL (21UCS057)

SAMEER SAXENA (21UCS179)

Assignment: Use a pre-trained suitable CNN architecture and classify the image. The result should contain the class and the accuracy of its prediction.

DESCRIPTION OF ARCHITECTURE: -

The VGG16 (Visual Geometry Group 16) is a convolutional neural network (CNN) architecture that was introduced by the Visual Geometry Group at the University of Oxford. It became well-known for its simplicity and achieved remarkable performance on image classification tasks. Here's a description of the VGG16 architecture:

Input Layer:

VGG16 takes an input image of size 224x224 pixels with three color channels (RGB).

Convolutional Blocks:

VGG16 consists of 13 convolutional layers, each followed by a rectified linear unit (ReLU) activation function.

The convolutional layers have small receptive fields (3x3) with a stride of 1 and are designed to capture local patterns.

Pooling Layers:

After each set of convolutional layers, max-pooling layers are used to downsample the spatial dimensions.

Max-pooling is performed using 2x2 windows with a stride of 2.

Fully Connected Layers:

Following the convolutional and pooling layers, VGG16 has three fully connected layers (also known as dense layers) for high-level reasoning.

The fully connected layers are followed by ReLU activation functions.

Output Layer:

The final layer is a softmax activation layer with 1000 units, corresponding to the 1000 ImageNet classes.

It outputs the probabilities for each class, and the class with the highest probability is considered the predicted class.

Parameter Count:

VGG16 has a relatively large number of parameters, approximately 138 million, due to its deep architecture.

1) Loading the required libraries:

```
from tensorflow.keras.preprocessing import image
from tensorflow.keras.applications.vgg16 import VGG16, preprocess_input, decode_predictions
import numpy as np
import matplotlib.pyplot as plt
```

2) Loading the pre-trained VGG-16 model

```
# we are loading the pre-trained VGG16 model
model = VGG16(weights='imagenet')
```

3) Creating a calculate accuracy function and declaring a variable ans to store the count of correctly predicted label by the model

```
# created a function to predict the class and calculate accuracy
def calculate_accuracy(image_paths, ground_truth_labels,total_img):
    ans=0
```

4) using for loop to predict class for each image

```
# used for loop to predict class for each image
  for image_path, ground_truth_label in zip(image_paths, ground_truth_labels):
```

5)Loading and preprocessing the image

```
# we are loading and preprocessing the input image
img = image.load_img(image_path, target_size=(224, 224))  # VGG16 input size
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array = preprocess_input(img_array)
```

6)Using model to make a prediction of the image

```
# Geting model predictions
predictions = model.predict(img_array)
decoded_predictions = decode_predictions(predictions, top=1)[0]
top_prediction = decoded_predictions[0]
predicted_label = top_prediction[1]
```

7)Displaying the image

```
# we are displaying the image with predicted class and its ground truth
plt.figure()
plt.imshow(img)
plt.axis('off')
plt.title(f"Predicted class: {predicted_label}\nGround truth: {ground_truth_label}")
plt.show()
```

8)Printing the predicted label and increasing the ans variable if predicted label==ground truth label to calculate accuracy.

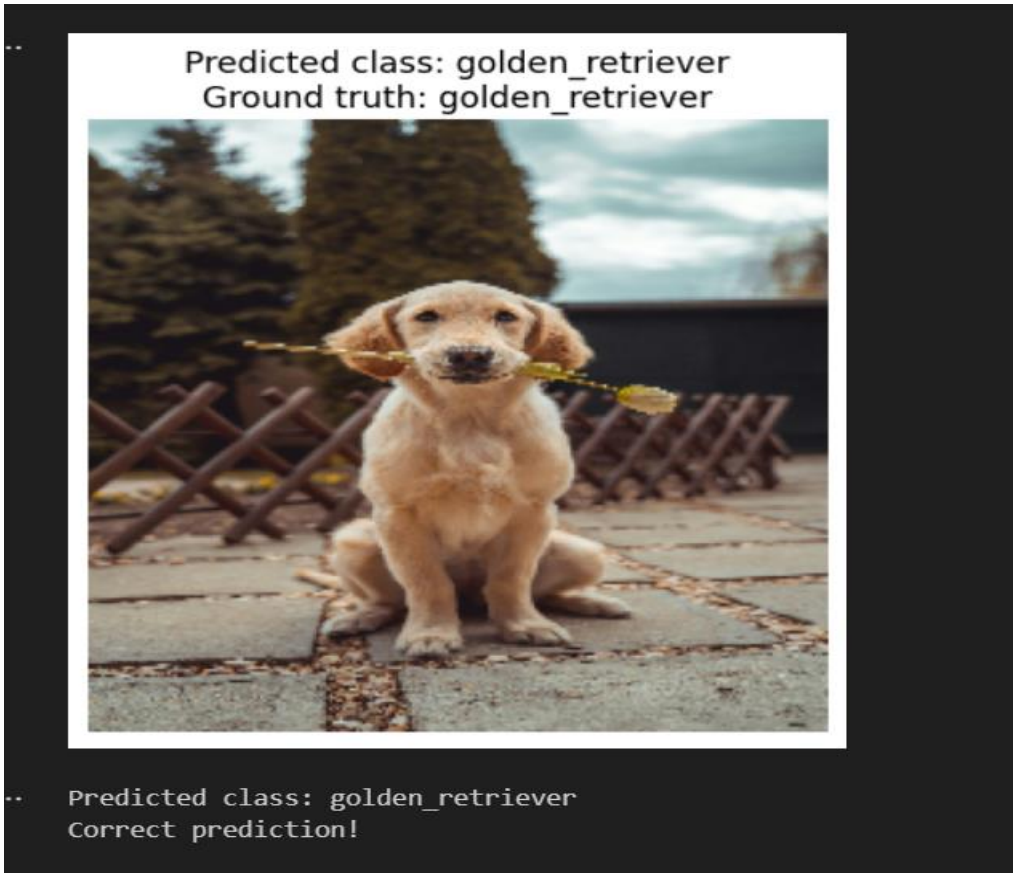
```
        print(f"Predicted class: {predicted_label}")


        if predicted_label == ground_truth_label:
            print("Correct prediction!")
            ans=ans+1
            print()
            # return 1
        else:
            print("Incorrect prediction")
            print()
            # return 0

    print("Accuracy:=",ans*100/total_img,"%")
```

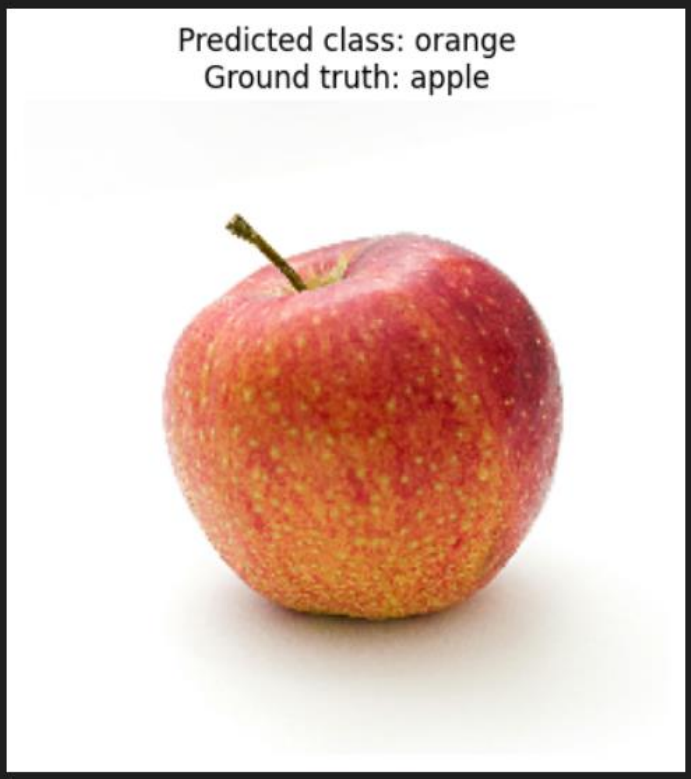
9)calling the calculate_accuaracy function and giving image paths and there ground truth label as its input

```
if __name__ == "__main__":


    image_paths = ['/content/amirali-mirhashemian-sc5sTPMrVfk-unsplash.jpg', '/content/richard-brutyo-Sg3XwuEpybU-unsplash.jpg', '/content/pexels-mali-maeder-102104.jpg','/conten
    ground_truth_labels = ['cheeseburger', 'golden_retriever', 'apple','soccer_ball','tiger']

    calculate_accuracy(image_paths, ground_truth_labels,5)
```

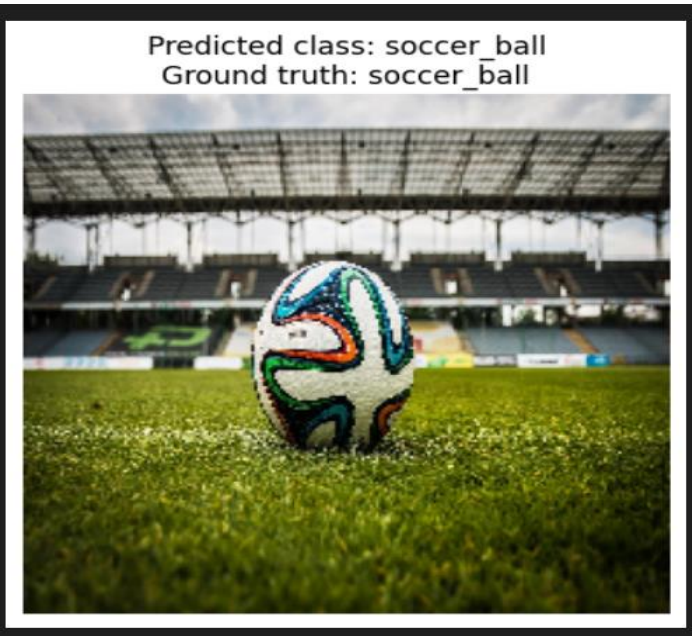
10)Output:


Predicted class: cheeseburger
Ground truth: cheeseburger
Predicted class: cheeseburger
Correct prediction!

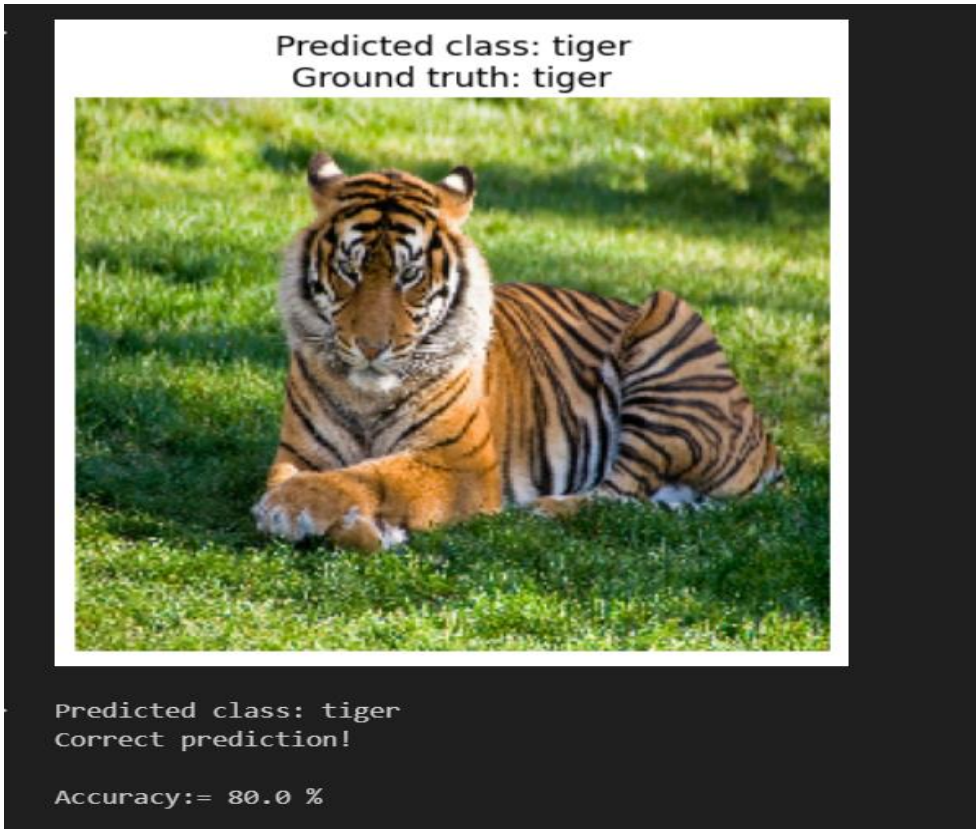

Predicted class: golden_retriever
Ground truth: golden_retriever
Predicted class: golden_retriever
Correct prediction!




```
Predicted class: orange
Incorrect prediction
```




```
Predicted class: soccer_ball
Correct prediction!
```



11) Conclusion:

The model predicts the label of the image with an

Accuracy: 80.0%