Azure Computer Vision Service

By - Jayabharathi Hari
Deakin University
✉: s224643593@deakin.edu.au

Azure Image Analysis Introduction

The Azure AI Vision Image Analysis service can extract a wide variety of visual features from your images. For example, it can determine whether an image contains adult content, find specific brands or objects, or find human faces.[1]

can use Image Analysis through a client library SDK or by calling the REST API directly.[1]

Image Analysis versions and features[1]

Important

Select the Image Analysis API version that best fits your requirements.

Expand table

Version	Features available	Recommendation
version 4.0	Read text, Captions, Dense captions, Tags, Object detection, Custom image classification / object detection, People, Smart crop	Better models; use version 4.0 if it supports your use case.
version 3.2	Tags, Objects, Descriptions, Brands, Faces, Image type, Color scheme, Landmarks, Celebrities, Adult content, Smart crop	Wider range of features; use version 3.2 if your use case is not yet supported in version 4.0

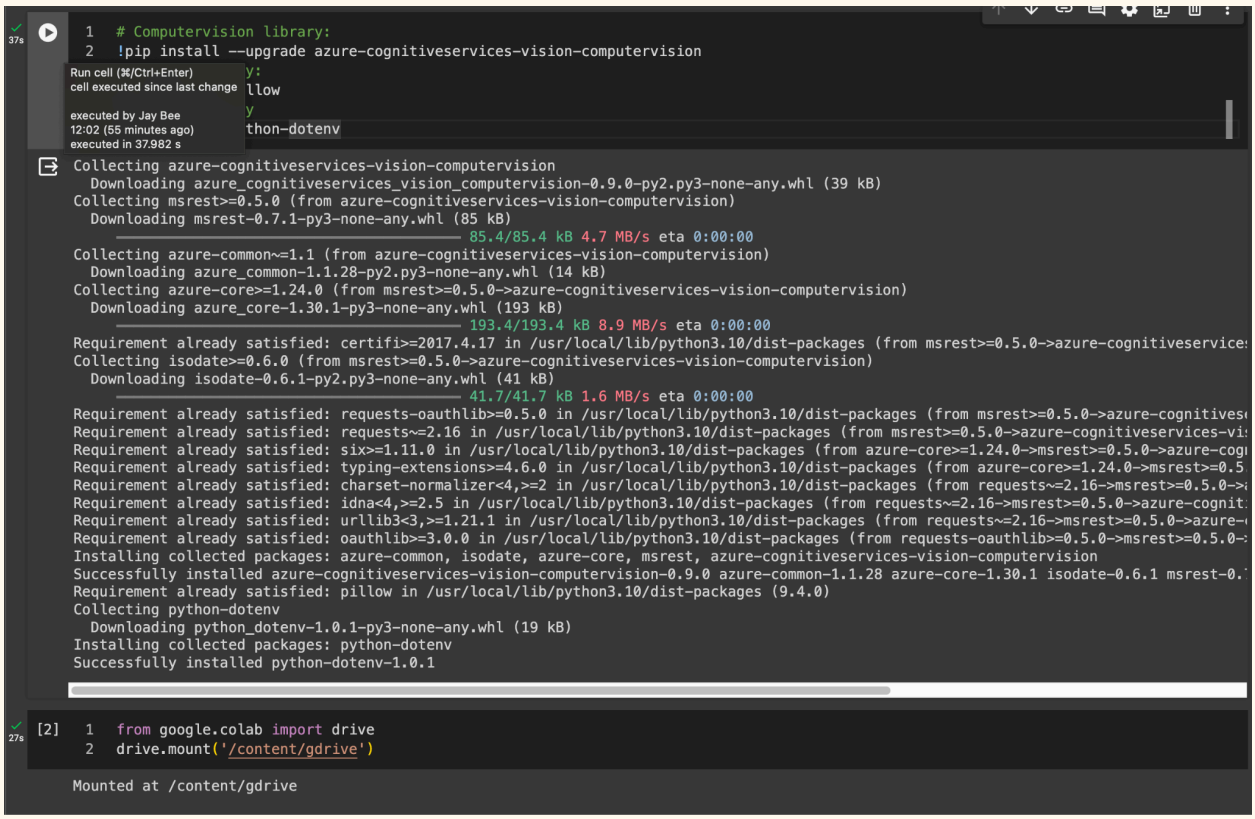
We recommend you use the Image Analysis 4.0 API if it supports your use case. Use version 3.2 if your use case is not yet supported by 4.0.

You'll also need to use version 3.2 if you want to do image captioning and your Vision resource is outside these Azure regions: East US, France Central, Korea Central, North Europe, Southeast Asia, West Europe, and West US, East Asia. The image captioning feature in Image Analysis 4.0 is only supported in these Azure regions. Image captioning in version 3.2 is available in all Azure AI Vision regions.

Fig1: fig as per dated on 20th Apr 2024

Object Detection using client library SDK - python

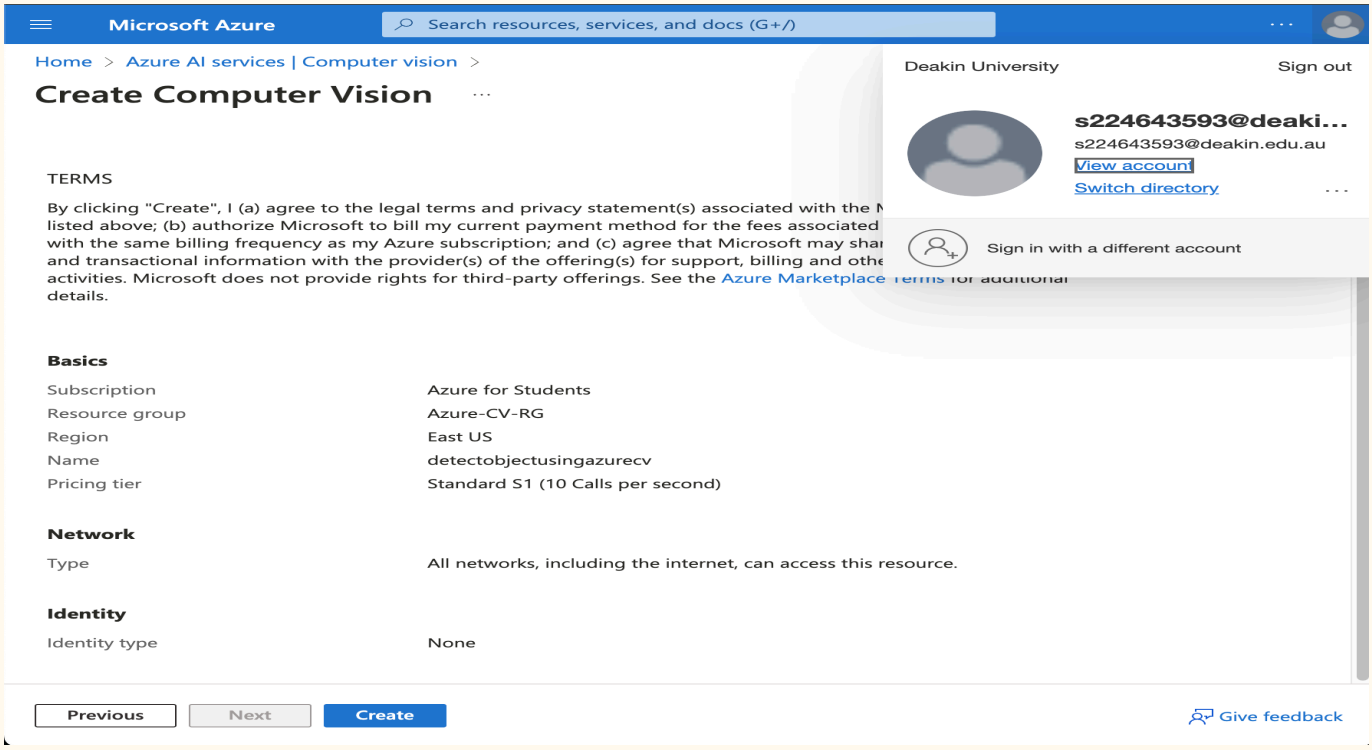
1.Setting Up Env and Workspace in Google Colab.

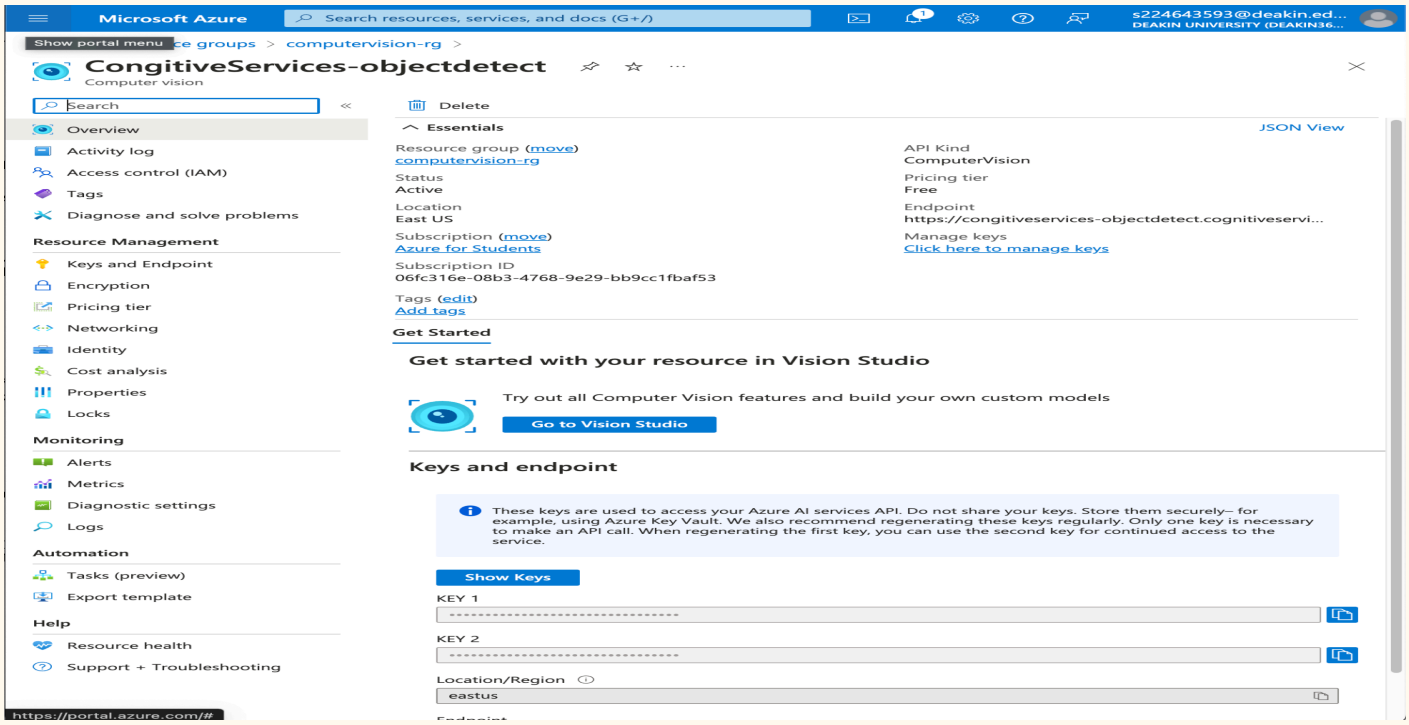


Code Explanation:

- the **“azure-cognitiveservices-vision-computervision”** library, which is used for interacting with the Azure Cognitive Services Computer Vision API.
- the **“Pillow”** library, which is a popular Python Imaging Library (PIL) fork. Pillow is used for opening, manipulating, and saving many different image file formats.

2.Creating Computer Vision Service in Azure.





Steps to Create Computer Vision Service:

1. Create Resource Group
2. Search for Computer Vision in Azure Portal then Create Computer Vision Recourse where we will get Keys and endpoint.

3.Downloading Packages.

```
[14]  1  from azure.cognitiveservices.vision.computervision import ComputerVisionClient
      2  from azure.cognitiveservices.vision.computervision.models import VisualFeatureTypes
      3  from msrest.authentication import CognitiveServicesCredentials
      4  from PIL import Image, ImageDraw, ImageFont
      5  import matplotlib.pyplot as plt
      6  import os
```

Code Explanation:

- **'ComputerVisionClient'** and **'VisualFeatureTypes'** from **'azure.cognitiveservices.vision.computervision'** are used to interact with the Computer Vision API and specify the visual features to be analyzed, respectively.
- **'CognitiveServicesCredentials'** from **'msrest.authentication'** is used for authenticating requests to the Cognitive Services API using the subscription key.
- **'Image', 'ImageDraw', and 'ImageFont'** from **'PIL'**(Python Imaging Library) are used for working with images, drawing on images, and adding text to images, respectively.
- '**matplotlib.pyplot**' is imported as **'plt'** to visualize images.
- **'os'** module is imported to interact with the operating system, primarily for file handling operations.

4. Initializing Azure Computer Vision Service

The **'ComputerVisionClient'** object is initialized with the **'endpoint'** URL and **'CognitiveServicesCredentials'** object created using the **'subscription_key'.** This client object is used to interact with the Computer Vision service, allowing access to its functionalities like image analysis and object detection.

```
# Authenticate
subscription_key = "2a058135e3d4491ebbf234cfd4d796c5"
endpoint = "https://congitiveservices-objectdetect.cognitiveservices.azure.com/"
computervision_client = ComputerVisionClient(endpoint, CognitiveServicesCredentials(subscription_key))
```

5. Declaring the Working Image

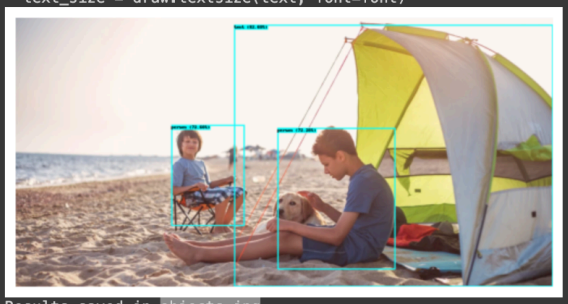
Printing the working Image using ‘Image.open()’ function is used to open an image file specified by the ‘file_path’ variable. This function is part of the PIL (Python Imaging Library) module, which is used for opening, manipulating, and saving many different image file formats.

```
[55]  # Image file path
      file_path = "/content/gdrive/MyDrive/MS_DataScience/Azure-CompVision/tent_beach.jpg"

[56]  Image.open(file_path)
```

6. Calling the API, Drawing Bounding Boxes, Adding Text Annotation for Objects with Confidence Level.

```python
print("===== Tag an image - remote =====")
# Call API with local image file
with open(file_path, mode='rb') as image_stream:
    analysis = computervision_client.analyze_image_in_stream(image_stream, visual_features=[VisualFeatureTypes.objects])
    # Print results with confidence score
    print("Tags in the remote image: ")

'''
Object Detection
This example locates objects, identifies them, and draws bounding boxes around them in a local image.
'''
print("===== Object Detection =====")
# Get objects in the image
if analysis.objects:
    print("Objects in image:")
    # Prepare image for drawing
    image = Image.open(file_path)
    draw = ImageDraw.Draw(image)
    font = ImageFont.load_default()  # Load default font
    color = 'cyan'
    for detected_object in analysis.objects:
        # Print object name
        print(" - {} (confidence: {:.2f}%)".format(detected_object.object_property, detected_object.confidence * 100))
        # Draw object bounding box
        r = detected_object.rectangle
        bounding_box = [(r.x, r.y), (r.x + r.w, r.y + r.h)]
        draw.rectangle(bounding_box, outline=color, width=3)
        # Add text annotation for object name and confidence level
        text = "{} ({:.2f}%)".format(detected_object.object_property, detected_object.confidence * 100)
        text_size = draw.textsize(text, font=font)
        draw.rectangle([(r.x, r.y), (r.x + text_size[0], r.y + text_size[1])], fill=color)
        draw.text((r.x, r.y), text, fill='black', font=font)
    # Display the annotated image
    plt.imshow(image)
    plt.axis('off')
    plt.show()
    # Save the annotated image
    outputfile = 'objects.jpg'
    image.save(outputfile)
    print('Results saved in', outputfile)
else:
    print('No objects found in the image.')
```

```
===== Tag an image - remote =====
Tags in the remote image:
===== Object Detection =====
Objects in image:
 - person (confidence: 72.60%)
 - person (confidence: 72.20%)
 - tent (confidence: 82.60%)
<ipython-input-57-3124a35d7987>:30: DeprecationWarning: textsize is deprecated and will be removed in Pillow 10 (2023-07-01). Use
  text_size = draw.textsize(text, font=font)
<ipython-input-57-3124a35d7987>:30: DeprecationWarning: textsize is deprecated and will be removed in Pillow 10 (2023-07-01). Use
  text_size = draw.textsize(text, font=font)
<ipython-input-57-3124a35d7987>:30: DeprecationWarning: textsize is deprecated and will be removed in Pillow 10 (2023-07-01). Use
  text_size = draw.textsize(text, font=font)
```



```
Results saved in objects.jpg
```

Code Explaination:

Tagging an Image:

- The code starts by printing a message indicating that it's tagging a remote image.

- The analyze_image_in_stream method of the computervision_client object is called to analyze the image in a stream.
- The method takes the image stream (opened with open(file_path, mode='rb')) as input and specifies VisualFeatureTypes.objects to request object detection.

Object Detection:

- The code then prints a message indicating that it's performing object detection.
- It checks if any objects were detected in the image by checking the analysis.objects attribute.
- If objects are detected:
    - It prints the name of each detected object along with its confidence score.
    - It prepares the image for drawing by opening it using Image.open(file_path).
    - It creates an ImageDraw object to draw on the image.
    - It sets up the font and color for drawing bounding boxes.
    - It iterates over each detected object:
        - Prints the object name and confidence score.
        - Draws a bounding box around the object on the image.
        - Adds text annotation for the object name and confidence level.
    - Displays the annotated image using Matplotlib.
    - Saves the annotated image to a file named 'objects.jpg'.
- If no objects are detected, it prints a message indicating that no objects were found in the image.

7. About API

The Azure Computer Vision API offers a wide range of image processing capabilities, including object detection, image tagging, image analysis, optical character recognition (OCR), and more. It allows developers to integrate advanced computer vision capabilities into their applications without needing to develop and train their own machine learning models. **Specifically, the analyze_image_in_stream method of the ComputerVisionClient object is used to analyze a given image and detect objects within it.** This method takes an image stream (binary data representing an image) as input and returns the results of the analysis, including information about the detected objects and their bounding boxes.

8. Final Output: