

Azure Computer Vision Service

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

Azure Image Analysis Introduction.....	1
Image Analysis versions and features[1].....	2
Object Detection using client library SDK - python.....	3
1. Setting Up Env and Workspace in Google Colab.....	3
2.Creating Computer Vision Service in Azure.....	4
3.Downloading Packages.....	5
4. Initializing Azure Computer Vision Service.....	5
5. Declaring the Working Image.....	6
6. Calling the API, Drawing Bounding Boxes, Adding Text Annotation for Objects with Confidence Level.....	6
7. About API.....	8
8. Final Output:.....	9

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](#)

Vers ion	Features available	Recommendation
versi on 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.
versi on 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

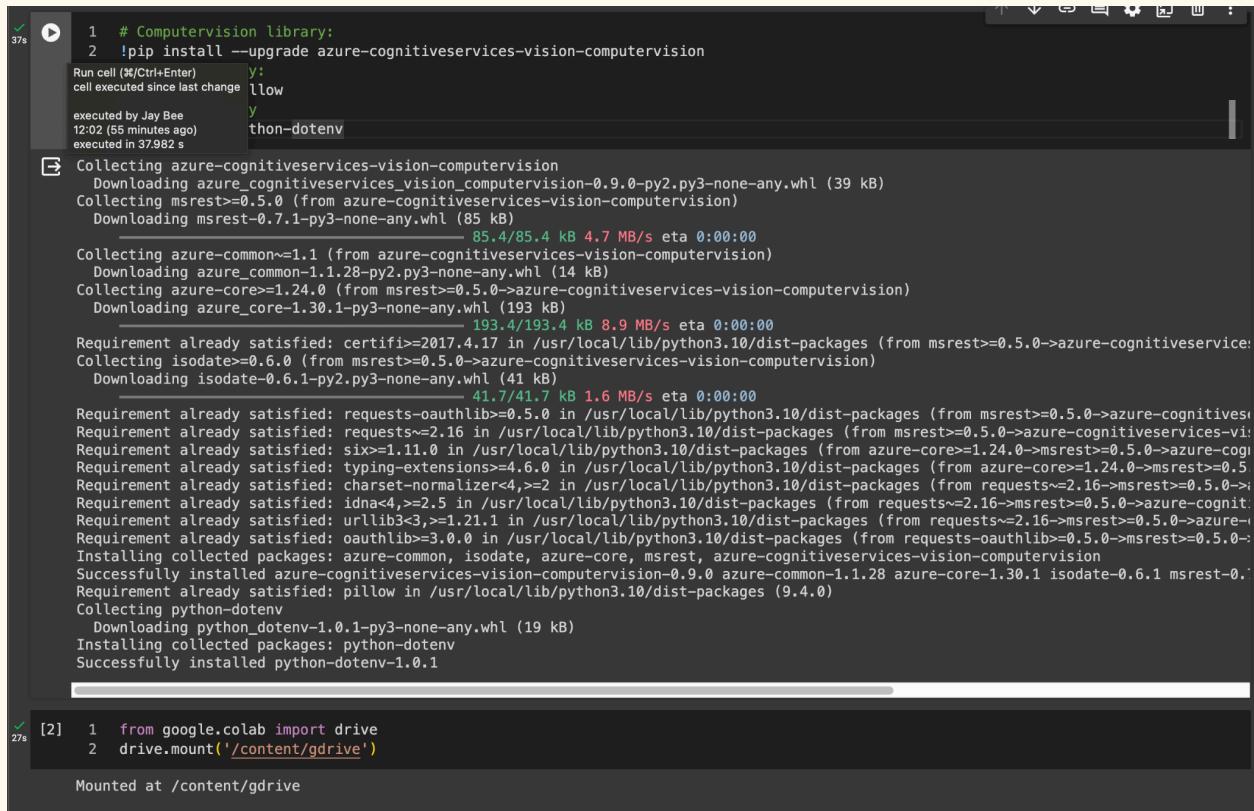
We recommend you use the Image Analysis 4.0 API if it supports your use case. Use version 3.2 if your use case is not yet supported by 4.0.

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.



```

1 # Computervision library:
2 !pip install --upgrade azure-cognitiveservices-vision-computervision
Run cell (%(Ctrl+Enter)
cell executed since last change
executed by Jay Bee
12:02 (55 minutes ago)
executed in 37.982 s
y:
thon-dotenv

Collecting azure-cognitiveservices-vision-computervision
  Downloading azure_cognitiveservices_vision_computervision-0.9.0-py2.py3-none-any.whl (39 kB)
Collecting msrest>=0.5.0 (from azure-cognitiveservices-vision-computervision)
  Downloading msrest-0.7.1-py3-none-any.whl (85 kB)
    85.4/85.4 kB 4.7 MB/s eta 0:00:00
Collecting azure-common<=1.1 (from azure-cognitiveservices-vision-computervision)
  Downloading azure_common-1.1.28-py2.py3-none-any.whl (14 kB)
Collecting azure-core>=1.24.0 (from msrest>=0.5.0->azure-cognitiveservices-vision-computervision)
  Downloading azure_core-1.30.1-py3-none-any.whl (193 kB)
    193.4/193.4 kB 8.9 MB/s eta 0:00:00
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from msrest>=0.5.0->azure-cognitiveservice)
Collecting isodate>=0.6.0 (from msrest>=0.5.0->azure-cognitiveservices-vision-computervision)
  Downloading isodate-0.6.1-py2.py3-none-any.whl (41 kB)
    41.7/41.7 kB 1.6 MB/s eta 0:00:00
Requirement already satisfied: requests-oauthlib>=0.5.0 in /usr/local/lib/python3.10/dist-packages (from msrest>=0.5.0->azure-cognitives)
Requirement already satisfied: requests<=2.16 in /usr/local/lib/python3.10/dist-packages (from msrest>=0.5.0->azure-cognitiveservices-vi
Requirement already satisfied: six>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from azure-core>=1.24.0->msrest>=0.5.0->azure-cog
Requirement already satisfied: typing-extensions>=4.6.0 in /usr/local/lib/python3.10/dist-packages (from azure-core>=1.24.0->msrest>=0.5
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests<~2.16->msrest>=0.5.0->
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<~2.16->msrest>=0.5.0->azure-cognit
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<~2.16->msrest>=0.5.0->azure-c
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.5.0->msrest>=0.5.0->
Installing collected packages: azure-common, isodate, azure-core, msrest, azure-cognitiveservices-vision-computervision
Successfully installed azure-cognitiveservices-vision-computervision-0.9.0 azure-common-1.1.28 azure-core-1.30.1 isodate-0.6.1 msrest-0.7.1
Requirement already satisfied: pillow in /usr/local/lib/python3.10/dist-packages (9.4.0)
Collecting python-dotenv
  Downloading python_dotenv-1.0.1-py3-none-any.whl (19 kB)
Installing collected packages: python-dotenv
Successfully installed python-dotenv-1.0.1

[2] 1 from google.colab import drive
 2 drive.mount('/content/gdrive')

Mounted at /content/gdrive

```

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.

Microsoft Azure

Search resources, services, and docs (G+/)

Home > Azure AI services | Computer vision > Create Computer Vision

Deakin University

s224643593@deakin.edu.au
s224643593@deakin.edu.au
[View account](#)
[Switch directory](#)

TERMS
By clicking "Create", I (a) agree to the legal terms and privacy statement(s) associated with the Microsoft products and services listed above; (b) authorize Microsoft to bill my current payment method for the fees associated with the same billing frequency as my Azure subscription; and (c) agree that Microsoft may share my usage and transactional information with the provider(s) of the offering(s) for support, billing and other activities. Microsoft does not provide rights for third-party offerings. See the [Azure Marketplace terms](#) for additional details.

Basics

Subscription	Azure for Students
Resource group	Azure-CV-RG
Region	East US
Name	detectobjectusingazurecv
Pricing tier	Standard S1 (10 Calls per second)

Network
Type: All networks, including the internet, can access this resource.

Identity
Identity type: None

Previous Next Create Give feedback

Show portal menu [Compute groups](#) > [computervision-rg](#)

CognitiveServices-objectdetect

Search Delete

Essentials

Resource group (move)	API Kind
computervision-rg	ComputerVision
Status	Pricing tier
Active	Free
Location	Endpoint
East US	https://cognitiveservices-objectdetect.cognitiveserv...
Subscription (move)	Manage keys
Azure for Students	Click here to manage keys

Subscription ID: 06fc316e-08b3-4768-9e29-bb9cc1fbaf53

Tags ([edit](#)) [Add tags](#)

Get Started

Get started with your resource in Vision Studio

Try out all Computer Vision features and build your own custom models

[Go to Vision Studio](#)

Keys and endpoint

Show Keys

KEY 1
KEY 2

Location/Region: eastus

<https://portal.azure.com/#>

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.

```

✓ 0s [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.

```
[s] 1 # Authenticate
2 subscription_key = "2a058135e3d4491ebbf234cf4d796c5"
3 endpoint = "https://cognitiveservices-objectdetect.cognitiveservices.azure.com/"
4 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] 1 # Image file path
2 file_path = "/content/gdrive/MyDrive/MS_DataScience/Azure-CompVision/tent_beach.jpg"

[s] [56] 1 Image.open(file_path)
```



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

```

 1 print("===== Tag an image - remote =====")
 2 # Call API with local image file
 3 with open(file_path, mode='rb') as image_stream:
 4     analysis = computervision_client.analyze_image_in_stream(image_stream, visual_features=[VisualFeatureTypes.objects])
 5     # Print results with confidence score
 6     print("Tags in the remote image: ")
 7
 8 ...
 9 Object Detection
10 This example locates objects, identifies them, and draws bounding boxes around them in a local image.
11 ''
12 print("===== Object Detection =====")
13 # Get objects in the image
14 if analysis.objects:
15     print("Objects in image:")
16     # Prepare image for drawing
17     image = Image.open(file_path)
18     draw = ImageDraw.Draw(image)
19     font = ImageFont.load_default() # Load default font
20     color = 'cyan'
21     for detected_object in analysis.objects:
22         # Print object name
23         print("- {} (confidence: {:.2f}%)".format(detected_object.object_property, detected_object.confidence * 100))
24         # Draw object bounding box
25         r = detected_object.rectangle
26         bounding_box = [(r.x, r.y), (r.x + r.w, r.y + r.h)]
27         draw.rectangle(bounding_box, outline=color, width=3)
28         # Add text annotation for object name and confidence level
29         text = "{} ({:.2f}%)".format(detected_object.object_property, detected_object.confidence * 100)
30         text_size = draw.textsize(text, font=font)
31         draw.rectangle([(r.x, r.y), (r.x + text_size[0], r.y + text_size[1])], fill=color)
32         draw.text((r.x, r.y), text, fill='black', font=font)
33     # Display the annotated image
34     plt.imshow(image)
35     plt.axis('off')
36     plt.show()
37     # Save the annotated image
38     outputfile = 'objects.jpg'
39     image.save(outputfile)
40     print('Results saved in', outputfile)
41 else:
42     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:

