

assume Azure Custom Vision Service context



Sure, I can assume an Azure Custom Vision Service context. Azure Custom Vision Service is a cloud-based image recognition service that allows you to easily create and train custom image classifiers using your own images. Here are some key concepts to keep in mind in this context:

Project: A project is a container for a set of images and their associated tags and metadata. You can create multiple projects in Custom Vision Service, and each project can have multiple image tags.

Tag: A tag is a label or category that you assign to an image. Custom Vision Service uses tags to train image classifiers.

Iteration: An iteration is a version of a trained classifier. After you train a classifier, you can create an iteration of it and test it with new images to see how well it performs. You can also retrain the classifier and create new iterations as needed.

Endpoint: An endpoint is a REST API that you can use to integrate your custom image classifier into your own applications or services.

Using these concepts, you can create custom image classifiers for a variety of use cases, such as identifying objects in images, detecting anomalies in manufacturing processes, or recognizing faces in security applications.



Assume following Endpoint: <https://customvisionclassifiercartoon-prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/0fcdbc74-d040-4e80-b6ad-ef819eb32223/classify/iterations/Iteration2/url>

Set Prediction-Key Header to : aaa

Set Content-Type Header to : application/json

Set Body to : {"Url": "https://example.com/image.png"}




Based on the information you provided, it looks like you are trying to make a prediction using an Azure Custom Vision Service endpoint. Here are the steps to make the prediction using the endpoint and headers you provided:

1. Set the endpoint URL to: `https://customvisionclassifiertoon-prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/0fcdbc74-d040-4e80-b6ad-ef819eb32223/classify/iterations/Iteration2/url`
2. Set the **Prediction-Key** header to `aaa`. This header is used to authenticate the request and must contain your Custom Vision Service prediction key.
3. Set the **Content-Type** header to `application/json`. This header specifies the format of the request body.
4. Set the request body to: `{"Url": "https://example.com/image.png"}`. This specifies the URL of the image that you want to classify.
5. Send the HTTP request to the endpoint using your preferred programming language or tool.

Here's an example code snippet in Python that demonstrates how to make the prediction:

python

 Copy code

```
import requests url = 'https://customvisionclassifiertoon-  
prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/0fcdbc74-  
d040-4e80-b6ad-ef819eb32223/classify/iterations/Iteration2/url' headers = {  
'Prediction-Key': 'aaa', 'Content-Type': 'application/json' } data = { 'Url':  
'https://example.com/image.png' } response = requests.post(url,  
headers=headers, json=data) print(response.json())
```

Note that the response will be a JSON object containing the predicted tags and their associated probabilities for the input image.




modify code for the cyclic reading of the image-url from the console input



Sure! Here's a modified code example in Python that will read the image URLs from console input in a loop until the user enters a specific input to exit:

python

 Copy code

```
import requests url = 'https://customvisionclassifiertoon-  
prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/0fcdbc74-  
d040-4e80-b6ad-ef819eb32223/classify/iterations/Iteration2/url' headers = {
```

```
'Prediction-Key': 'aaa', 'Content-Type': 'application/json' } while True:
url_input = input("Enter the URL of the image to classify (or 'exit' to
quit): ") if url_input.lower() == 'exit': break data = { 'Url': url_input }
response = requests.post(url, headers=headers, json=data) if response.ok:
predictions = response.json()['predictions'] if predictions: for prediction
in predictions: tag = prediction['tagName'] probability =
prediction['probability'] print(f"Tag: {tag}, Probability: {probability}")
else: print("No predictions found for the input image.") else: print(f"Error:
{response.status_code} - {response.reason}")
```

In this modified code, we've added a **while** loop that reads URLs from the console input until the user enters the word 'exit'. The **input** function is used to read the input from the user.

For each URL entered by the user, we create a **data** dictionary containing the URL and send a POST request to the endpoint using the **requests** library. We then check if the response is successful using the **ok** attribute of the response object.

If the response is successful, we parse the JSON response and print out the predicted tags and their associated probabilities using a **for** loop. If no predictions are found for the input image, we print a message indicating so.

If the response is not successful, we print an error message with the HTTP status code and reason.



Tell about use cases



Azure Custom Vision Service can be used in a variety of image classification use cases, some of which include:

1. Object detection and recognition: Custom Vision Service can be used to identify and recognize objects within images, such as vehicles, animals, and furniture. This can be useful in applications such as automated inventory management, traffic monitoring, and wildlife tracking.
2. Face recognition: Custom Vision Service can also be used for face recognition, which can be useful in security and surveillance applications, as well as in social media and marketing campaigns.

3. Anomaly detection: Custom Vision Service can help identify anomalies in manufacturing processes, medical imaging, and other applications where detecting unusual patterns or deviations from a norm is critical.
4. Brand and logo recognition: Custom Vision Service can recognize and classify specific brands and logos, which can be useful in market research, advertising, and brand management.
5. Quality control: Custom Vision Service can be used in manufacturing and production lines to detect defects, such as scratches or dents, in products and parts.

These are just a few examples of the many possible use cases for Azure Custom Vision Service. The service can be customized and trained to recognize a wide range of objects and patterns, making it a versatile tool for many industries and applications.



modify code for the cyclic reading of the image from local drive giving file-path in the console input



Sure! Here's a modified code example in Python that will read the file paths from console input in a loop until the user enters a specific input to exit:

python

Copy code

```
import requests url = 'https://customvisionclassifierecartoon-
prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/0fcdbc74-
d040-4e80-b6ad-ef819eb32223/classify/iterations/Iteration2/image' headers = {
'Prediction-Key': 'aaa', 'Content-Type': 'application/octet-stream' } while
True: file_path = input("Enter the file path of the image to classify (or
'exit' to quit): ") if file_path.lower() == 'exit': break with
open(file_path, 'rb') as file: data = file.read() response =
requests.post(url, headers=headers, data=data) if response.ok: predictions =
response.json()['predictions'] if predictions: for prediction in predictions:
tag = prediction['tagName'] probability = prediction['probability']
print(f"Tag: {tag}, Probability: {probability}") else: print("No predictions
found for the input image.") else: print(f"Error: {response.status_code} -
{response.reason}")
```

In this modified code, we've added a **while** loop that reads file paths from the console input until the user enters the word 'exit'. The **input** function is used to read the input from the user.

For each file path entered by the user, we read the contents of the file into a bytes object using the **open** function and **read** method with the **'rb'** mode. We then send a POST request to the endpoint using the **requests** library. We set the **Content-Type** header to **'application/octet-stream'** to indicate that we are sending binary data.

We then check if the response is successful using the **ok** attribute of the response object. If the response is successful, we parse the JSON response and print out the predicted tags and their associated probabilities using a **for** loop. If no predictions are found for the input image, we print a message indicating so.

If the response is not successful, we print an error message with the HTTP status code and reason.



modify code for the cyclic reading of the images from local drive giving image-directory in the console input, prompt for reading next file from the directory or exit



Sure! Here's a modified code example in Python that will read the file paths from a specified directory in the console input in a loop until the user enters a specific input to exit or no more images are available:

python

Copy code

```
import os import requests url = 'https://customvisionclassifiercartoon-
prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/0fcdbc74-
d040-4e80-b6ad-ef819eb32223/classify/iterations/Iteration2/image' headers = {
'Prediction-Key': 'aaa', 'Content-Type': 'application/octet-stream' } while
True: dir_path = input("Enter the directory path of the images to classify
(or 'exit' to quit): ") if dir_path.lower() == 'exit': break file_paths =
[os.path.join(dir_path, f) for f in os.listdir(dir_path) if
os.path.isfile(os.path.join(dir_path, f))] file_count = len(file_paths)
file_index = 0 while file_index < file_count: file_path =
file_paths[file_index] with open(file_path, 'rb') as file: data = file.read()
response = requests.post(url, headers=headers, data=data) if response.ok:
predictions = response.json()['predictions'] if predictions: for prediction
in predictions: tag = prediction['tagName'] probability =
prediction['probability'] print(f"Tag: {tag}, Probability: {probability}")
else: print(f"No predictions found for the image: {file_path}") else:
print(f"Error: {response.status_code} - {response.reason}") file_index += 1
if file_index < file_count: next_file_input = input(f"
```

```
{file_index}/{file_count} images classified. Enter 'n' to classify next
image, 'exit' to quit: ") if next_file_input.lower() == 'exit': break if
file_index >= file_count: print("All images classified.") break
```

In this modified code, we first prompt the user to enter a directory path containing the images to classify. We then use `os.listdir` to get a list of all the files in the directory and filter out any subdirectories. We create a list of file paths by joining the directory path with each file name using `os.path.join`.

We then enter a loop that iterates over each file in the list and reads its contents into a bytes object using the `open` function and `read` method with the `'rb'` mode. We then send a POST request to the endpoint using the `requests` library. We set the **Content-Type** header to `'application/octet-stream'` to indicate that we are sending binary data.

We then check if the response is successful using the `ok` attribute of the response object. If the response is successful, we parse the JSON response and print out the predicted tags and their associated probabilities using a `for` loop. If no predictions are found for the input image, we print a message indicating so.

If the response is not successful, we print an error message with the HTTP status code and reason.

After processing each image, we prompt the user to enter 'n' to classify the next image or 'exit' to quit. If the user enters 'exit', we break out of the loop and exit the program. If we have processed all the images, we print a message indicating so and exit the program.



split url to base- (ending with .com) and endpoint-part



Sure! Here's an example Python code that will split the URL into a base part and an endpoint part:

```
python
```

 Copy code

```
url = 'https://customvisionclassifiertoon-
prediction.cognitiveservices.azure.com/customvision/v3.0/Prediction/0fcdbc74-
```

```
d040-4e80-b6ad-ef819eb32223/classify/iterations/Iteration2/url' # Split the
URL into base and endpoint parts base_url = url.split('.com')[0] + '.com'
endpoint = url.split('.com')[-1] # Print the base and endpoint parts
print(f"Base URL: {base_url}") print(f"Endpoint: {endpoint}")
```

In this code, we first define the URL as a string. We then use the **split** method to split the URL at the **.com** string. This will create a list of two strings, with the base part as the first string and the endpoint part as the second string.

We then concatenate the base part with the **.com** string to get the full base URL. We also retrieve the endpoint part by indexing the second string from the split list.

Finally, we print out the base and endpoint parts using the **print** function.