

Image classification with the Azure Custom Vision

Command to install required libraries:

pip install azure-cognitiveservices-vision-customvision

```
In [2]: from azure.cognitiveservices.vision.customvision.training import CustomVisionTrain
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPr
from azure.cognitiveservices.vision.customvision.training.models import ImageFile
from msrest.authentication import ApiKeyCredentials
import os, time, uuid
```

```
In [3]: # Replace with valid values
ENDPOINT = "https://sdai.cognitiveservices.azure.com/"
PredictionENDPOINT = "https://sdai-prediction.cognitiveservices.azure.com/"
training_key = "56e05f26d6d84d42a99c877e392c6170"
prediction_key = "e9eb4c1a158c4bfc8da8397bb161ef2c"
prediction_resource_id = "/subscriptions/5cf92398-d8a0-400a-9d32-05fcb62b046b/res
```

Authenticate the client

Instantiate a training and prediction client with your endpoint and keys. Create `ApiKeyServiceClientCredentials` objects with your keys, and use them with your endpoint to create a `CustomVisionTrainingClient` and `CustomVisionPredictionClient` object.

```
In [4]: credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_key})
predictor = CustomVisionPredictionClient(PredictionENDPOINT, prediction_credentials)
```

Create a new Custom Vision project

Add the following code to your script to create a new Custom Vision service project.

See the `create_project` method to specify other options when you create your project (explained in the Build a classifier web portal guide).

```
In [5]: publish_iteration_name = "classifyModel"

credentials = ApiKeyCredentials(in_headers={"Training-key": training_key})
trainer = CustomVisionTrainingClient(ENDPOINT, credentials)

# Create a new project
print("Creating project...")
project_name = uuid.uuid4()
project = trainer.create_project(project_name)
print('d')
```

Creating project...
d

```
In [6]: # Make two tags in the new project
mango_tag = trainer.create_tag(project.id, "Mango")
apple_tag = trainer.create_tag(project.id, "Apple")
```

Upload and tag images

```
In [7]: #__file__ : Put location where your "images" folder is located in your system

base_image_location = os.path.join (os.path.dirname("C:/Users/badda/Downloads/Data"), "images")

print("Adding images...")

image_list = []

for image_num in range(1, 11):
    file_name = "mango_{}.jpg".format(image_num)
    with open(os.path.join (base_image_location, "Mango", file_name), "rb") as image_file:
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_file.read()))

for image_num in range(1, 11):
    file_name = "apple_{}.jpg".format(image_num)
    with open(os.path.join (base_image_location, "Apple", file_name), "rb") as image_file:
        image_list.append(ImageFileCreateEntry(name=file_name, contents=image_file.read()))

upload_result = trainer.create_images_from_files(project.id, ImageFileCreateBatch(image_list))
if not upload_result.is_batch_successful:
    print("Image batch upload failed.")
    for image in upload_result.images:
        print("Image status: ", image.status)
    exit(-1)
```

Adding images...

In []:

Train the project

```
In [8]: print ("Training...")
iteration = trainer.train_project(project.id)
while (iteration.status != "Completed"):
    iteration = trainer.get_iteration(project.id, iteration.id)
    print ("Training status: " + iteration.status)
    print ("Waiting 10 seconds...")
    time.sleep(10)
```

[illegible]

Waiting 10 seconds...
 Training status: Training
 Waiting 10 seconds...
 Training status: Completed
 Waiting 10 seconds...

Publish the current iteration

```
In [9]: # The iteration is now trained. Publish it to the project endpoint
trainer.publish_iteration(project.id, iteration.id, publish_iteration_name, prediction_credentials)
print ("Done!")
```

Done!

Test the prediction endpoint

```
In [10]: # Now there is a trained endpoint that can be used to make a prediction
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_credentials})
predictor = CustomVisionPredictionClient(PredictionENDPOINT, prediction_credentials)

with open(os.path.join (base_image_location, "Test/test_image.jpg"), "rb") as image_contents:
    results = predictor.classify_image(
        project.id, publish_iteration_name, image_contents.read())

# Display the results.
for prediction in results.predictions:
    print("\t" + prediction.tag_name +
          ": {0:.2f}%".format(prediction.probability * 100))
```

Mango: 100.00%
 Apple: 0.00%

```
In [11]: # Now there is a trained endpoint that can be used to make a prediction
prediction_credentials = ApiKeyCredentials(in_headers={"Prediction-key": prediction_credentials})
predictor = CustomVisionPredictionClient(PredictionENDPOINT, prediction_credentials)

with open(os.path.join (base_image_location, "Test/test.jpg"), "rb") as image_contents:
    results = predictor.classify_image(
        project.id, publish_iteration_name, image_contents.read())

# Display the results.
for prediction in results.predictions:
    print("\t" + prediction.tag_name +
          ": {0:.2f}%".format(prediction.probability * 100))
```

Apple: 100.00%
 Mango: 0.00%

In []: