Deakin University

SIG788- OnTrack Submission

Task 4.2 D

Submitted by

Neethu Sidhardhan
S223494027
Attempt # 2
3/5/2023

Target Grade: D

Task Details -

1) Developing a program using Azure computer vision and custom vision model to detect vehicles (car, bus, bicycle and etc. (up to 3 vehicles)) and track them in a one-minute video. You can use Azure custom vision for this task. The application must be developed using Python language, and Azure custom vision can be used for this task. It is important to use a small dataset of 50-100 images to train the custom vision program to recognise different types of vehicles accurately

2) Develop a near real-time vehicle tracking application. Your application should analyse video frames from a one-minute video using azure custom vision. In this assignment, you need to collect a dataset with different types of vehicles and label the images with corresponding vehicle type. Then you need to train your dataset using Azure custom vision and then deploy the model. You need to use the deployed model to track the vehicle in one-minute video. To decrease the cost of API usage, you need to send one frame per second to the API. The selected frames will be sent to the API to detect different vehicles. You will need to provide a demo (video) of how the model is developed and working (Max 3mins)

❖ Create custom vision in Azure. Select custom vision from Marketplace.

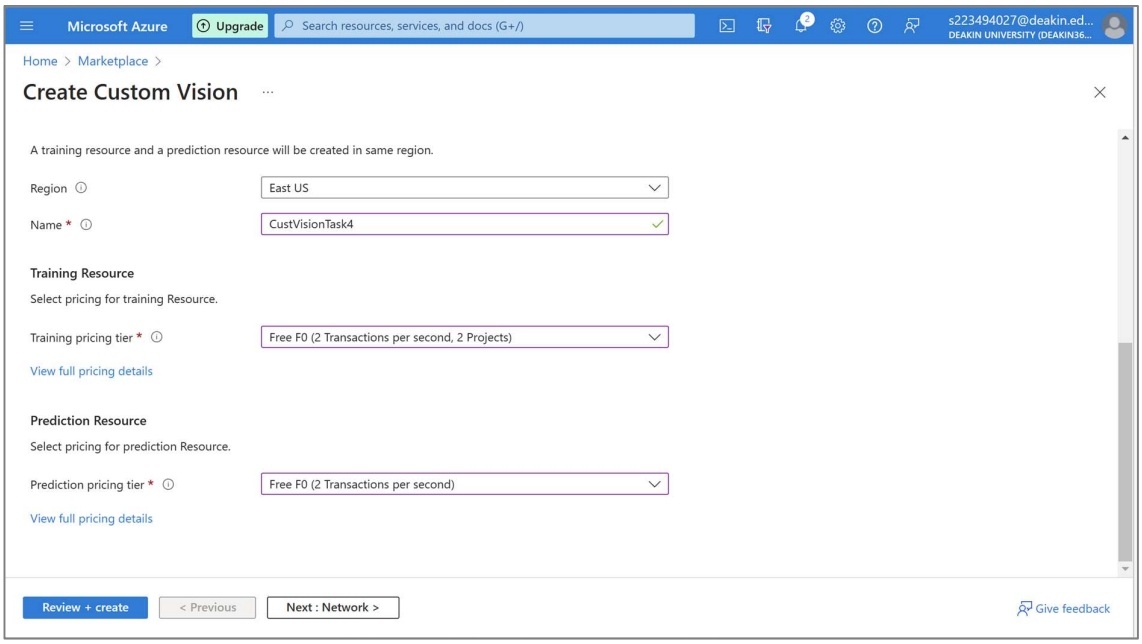


Fig 1: Create custom vision

While creating Custom Vision, here there are three options given to create “Both” that means training and prediction. Here we opt for selecting “Both” the train and test. We have to capture 2 separate keys and endpoint, the same is used for creating

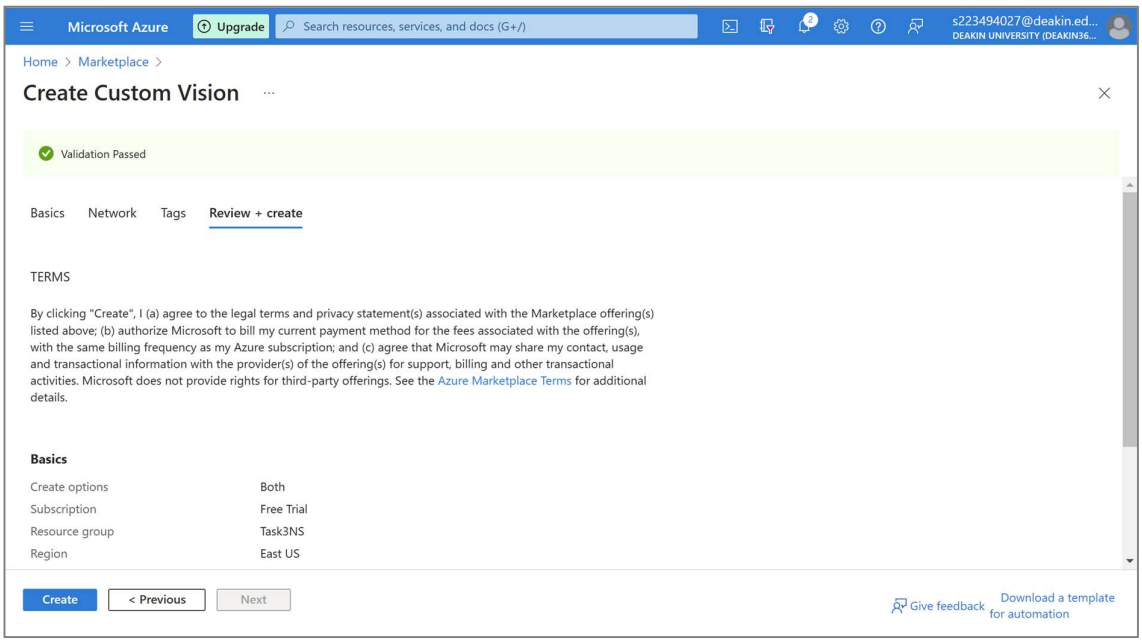


Fig 2: create custom vision

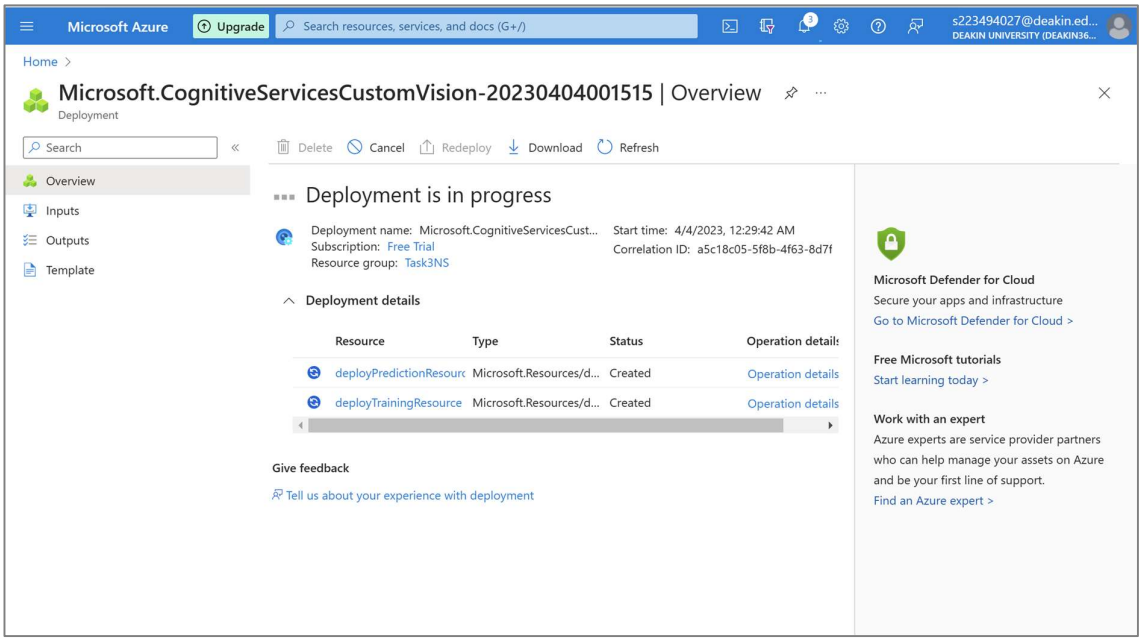


Fig 3 Deploying the custom vision

- Open “customvision.ai”
- Create a new project in custom vision

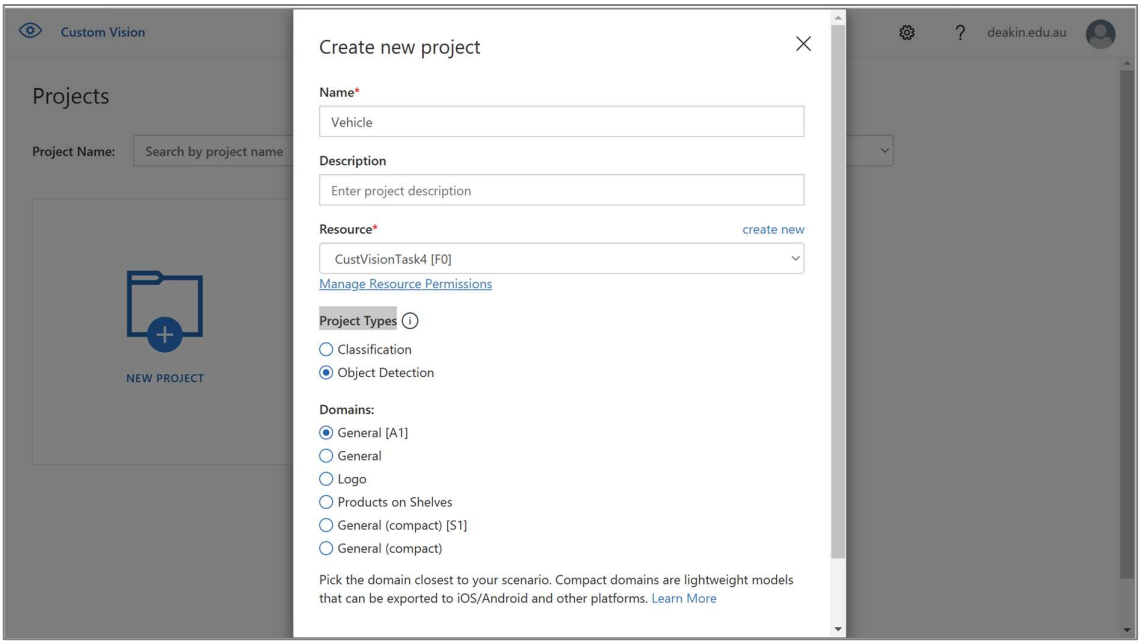


Fig 4 Create new project

❖ Under “Add images” upload all images of vehicle for “Training Images”.

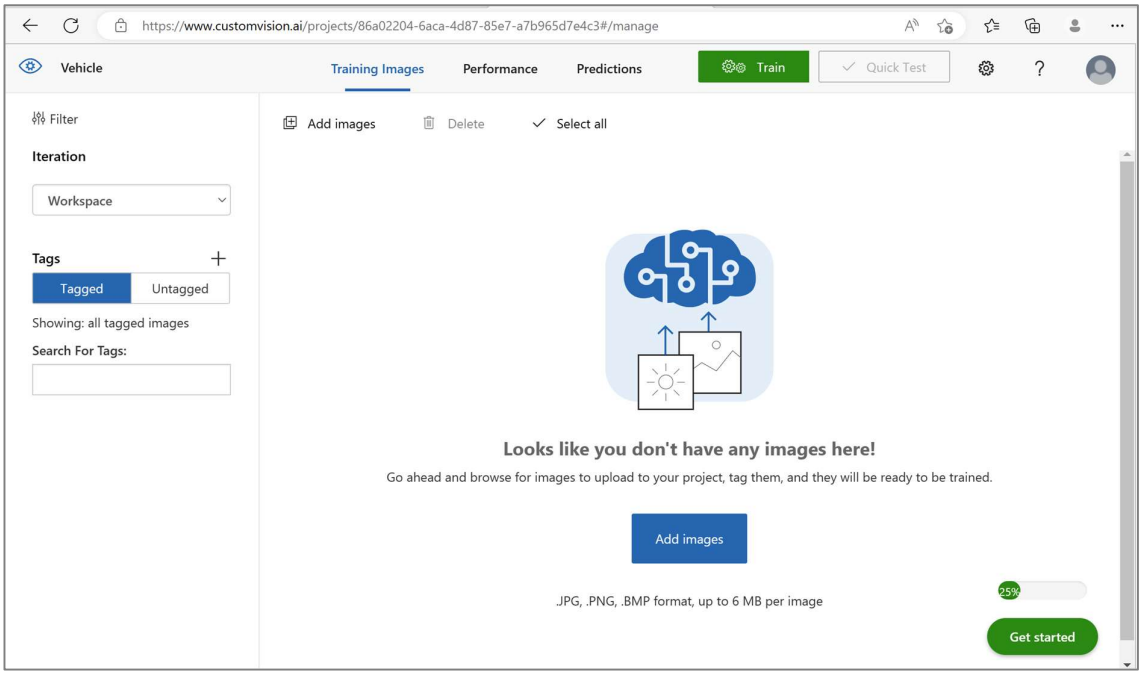


Fig 5 Adding Images

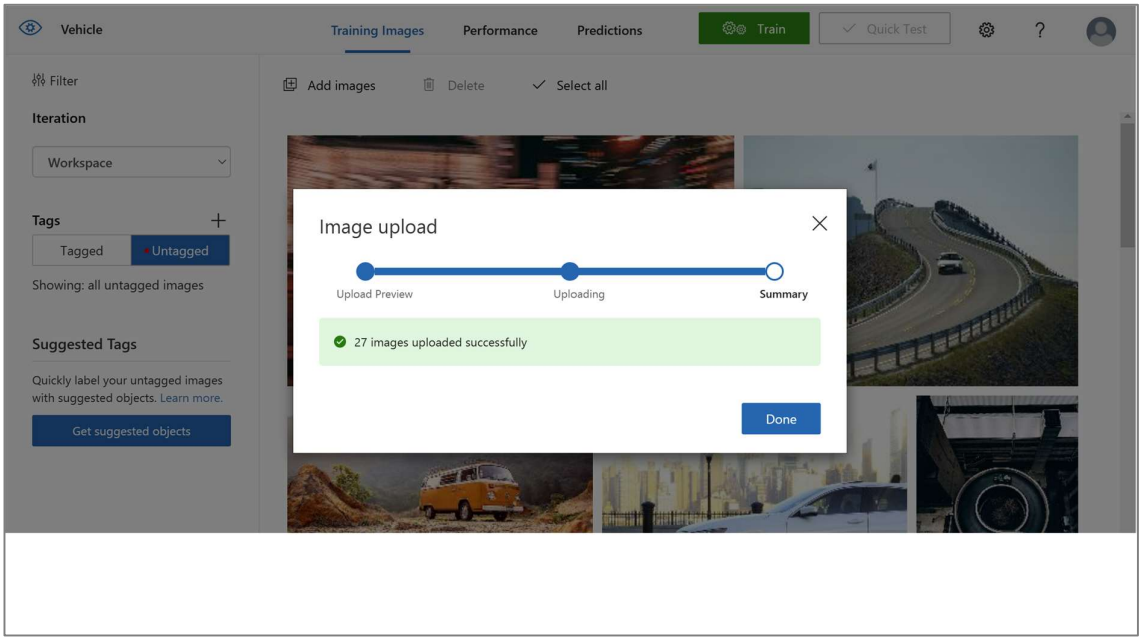


Fig 6 Uploading the Images

❖ Once the images are uploaded, tag each image.

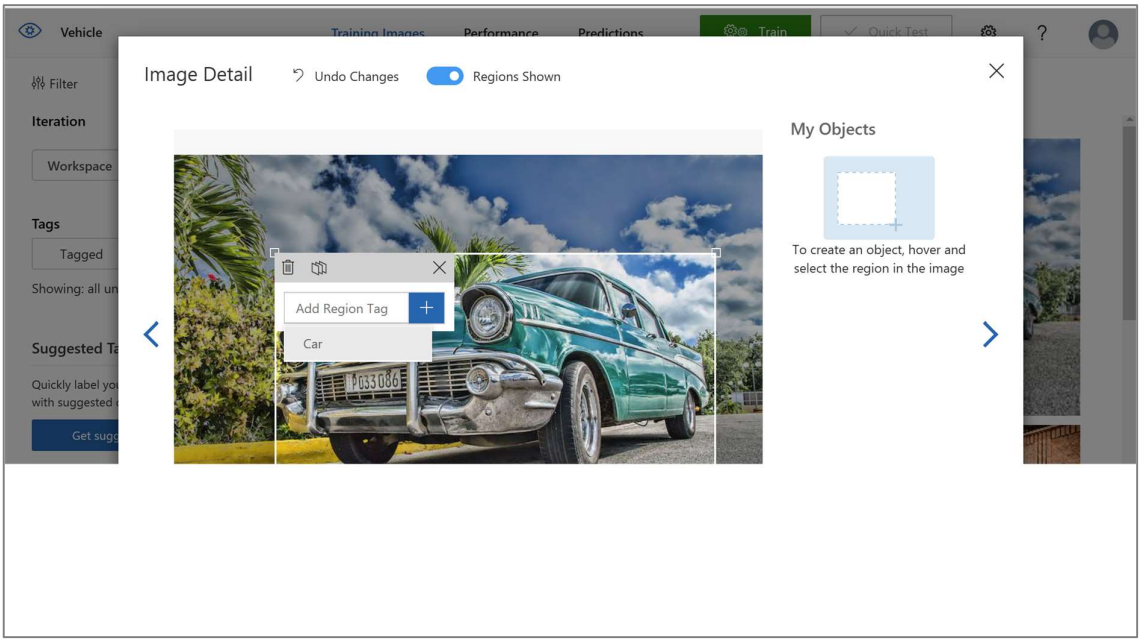


Fig 7 Tag the Images

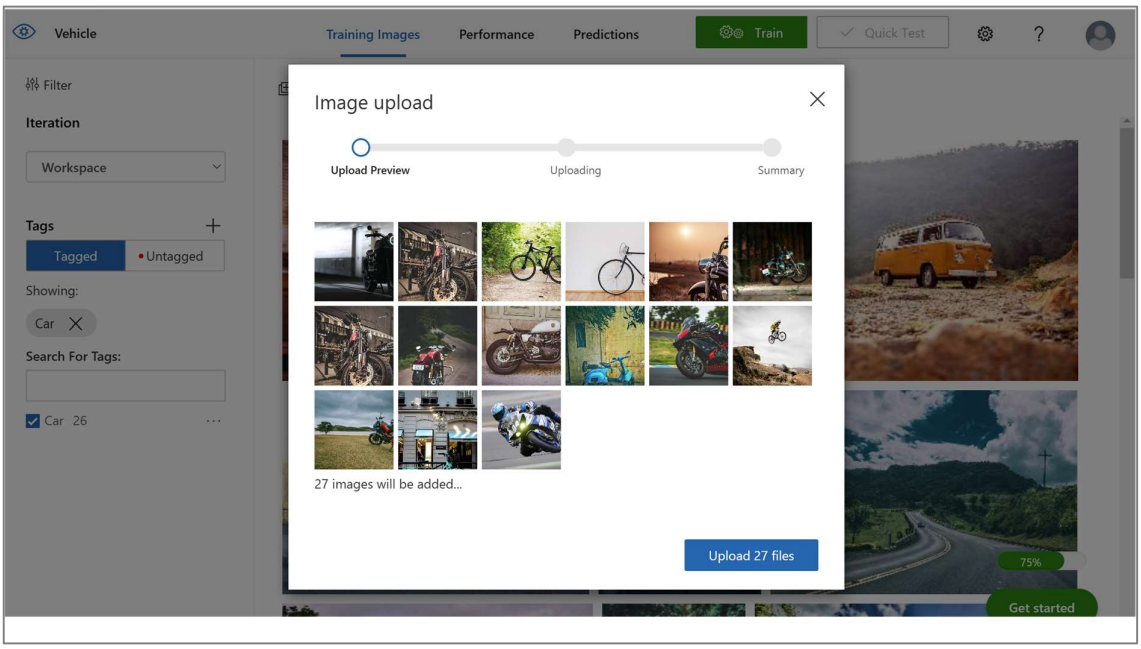


Fig 8 Upload the images of Bike

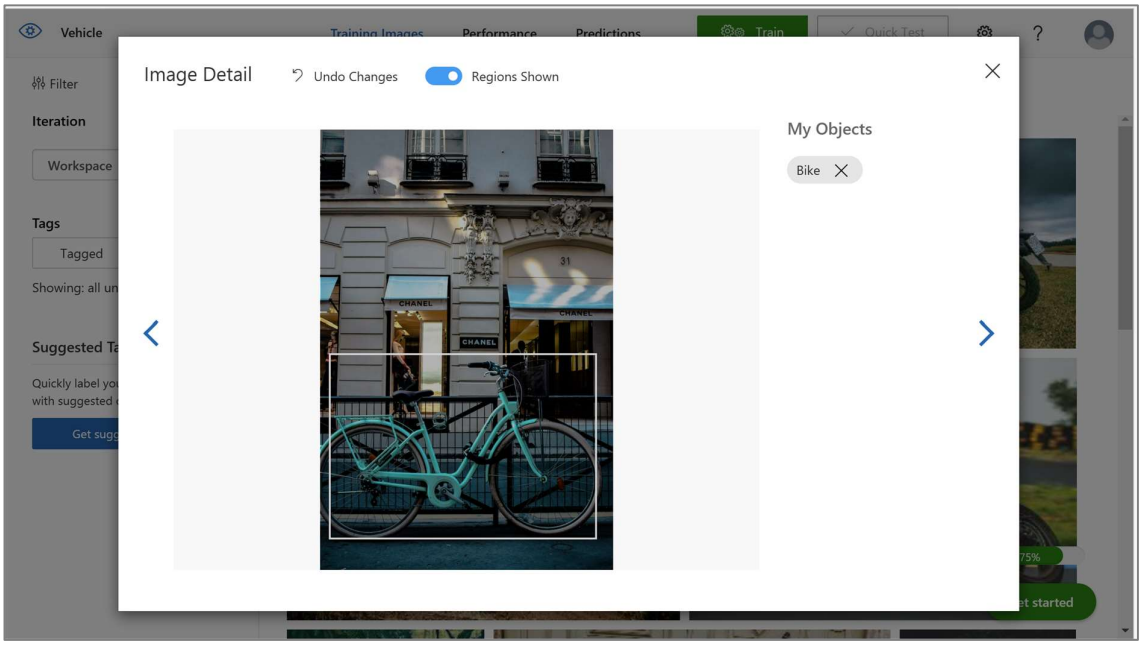


Fig 9 Tag Bikes

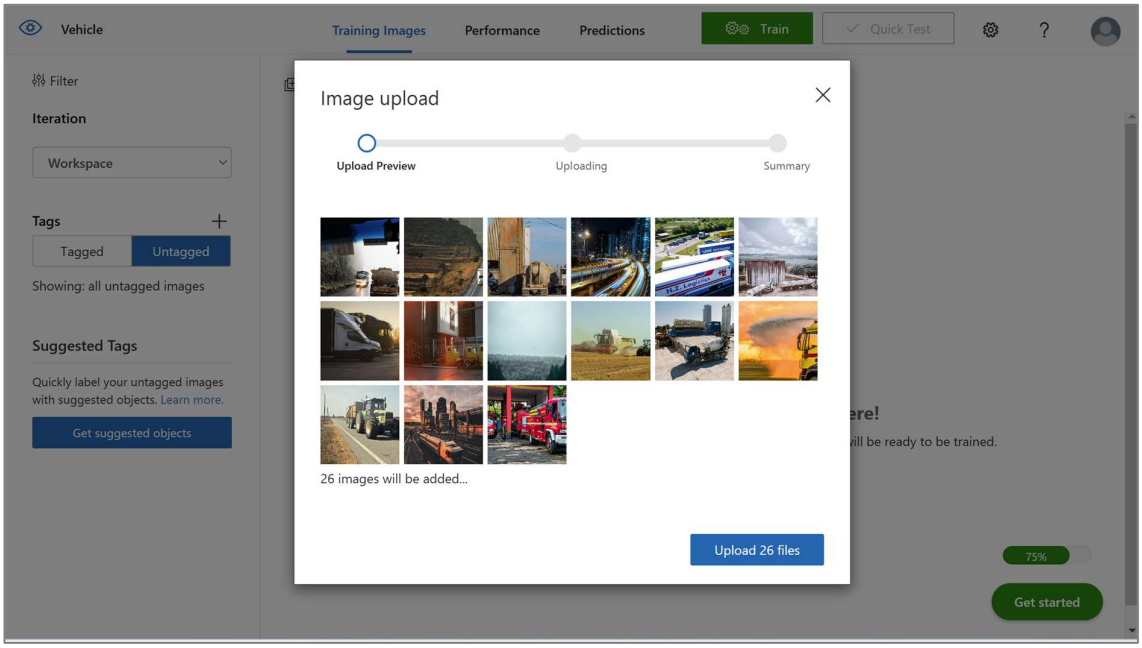


Fig 10 Upload images of Truck

❖ Once all images are uploaded and tagged. Select the option for “Training”

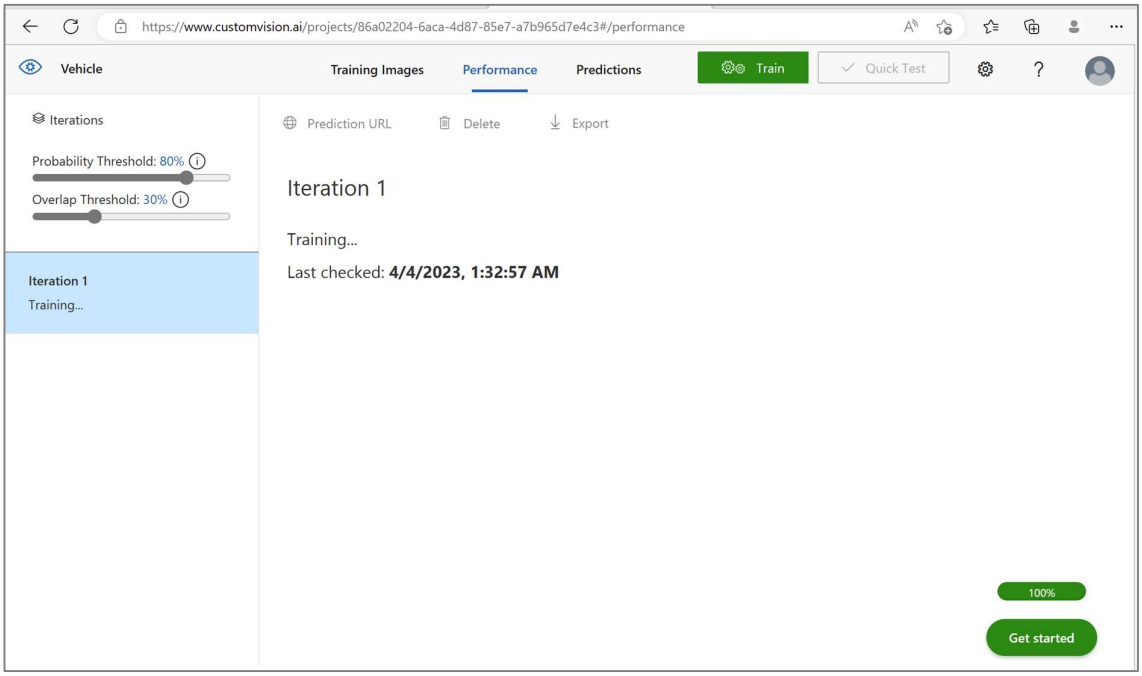


Fig 11 Training under different Threshold

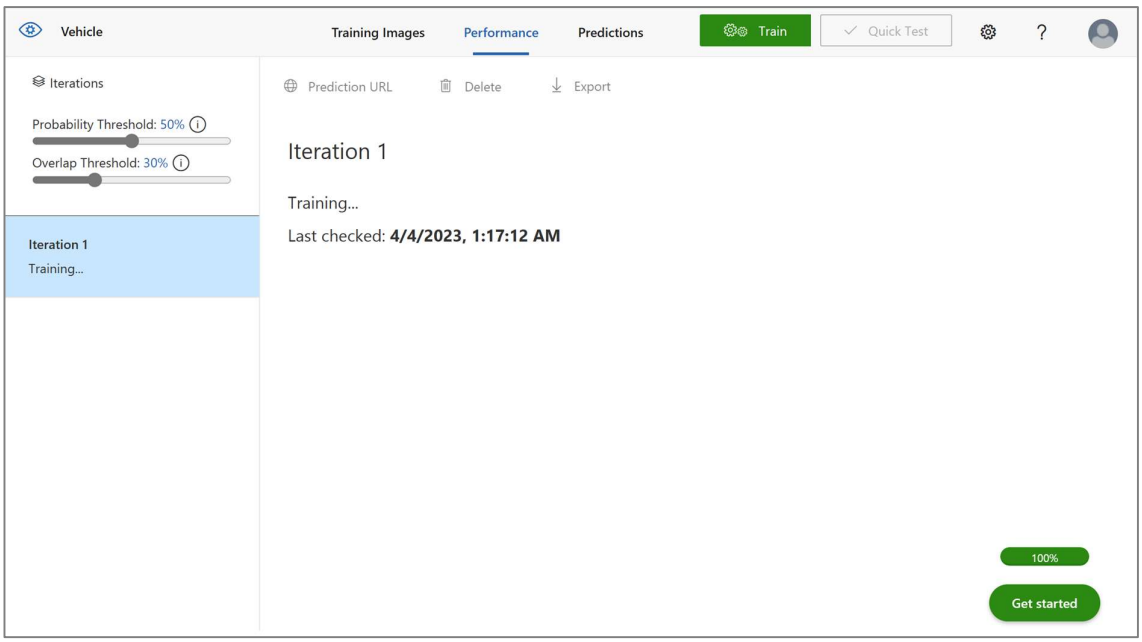


Fig 12 Training under different Threshold

❖ Performance of the model-based training

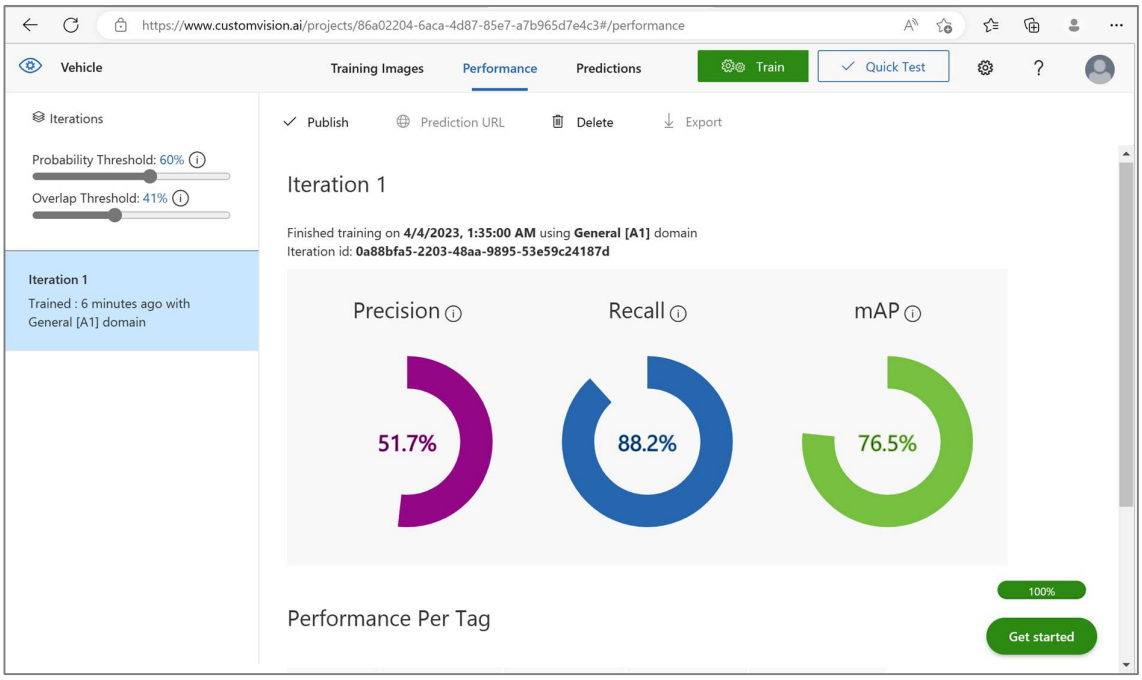


Fig 13 Performance of model

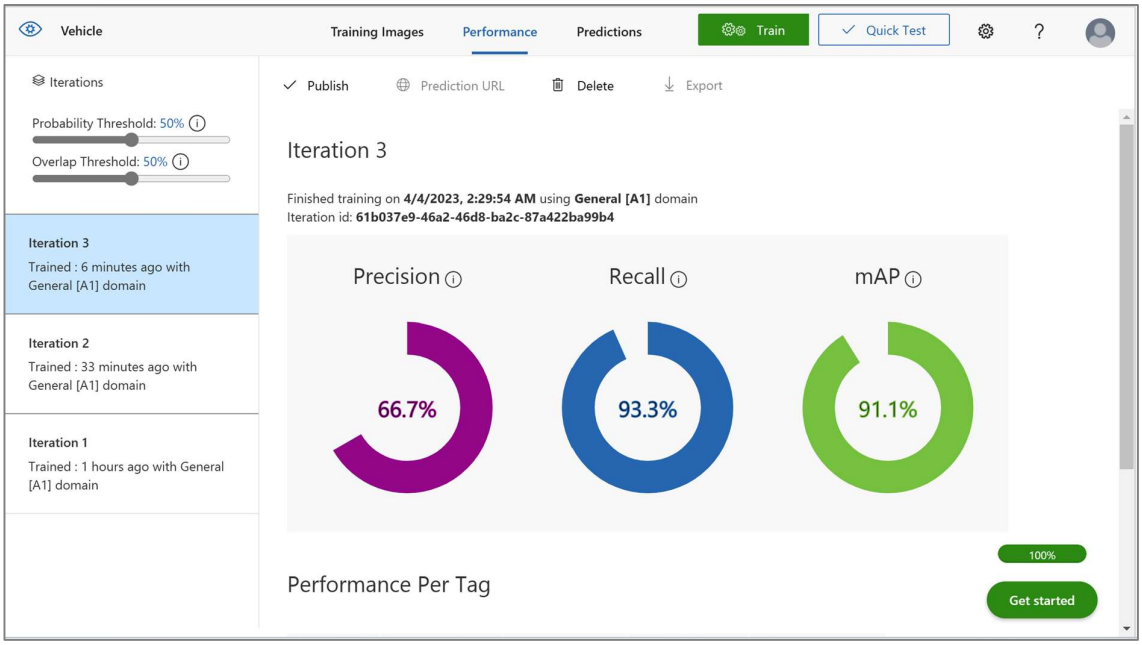


Fig 14 Performance of model

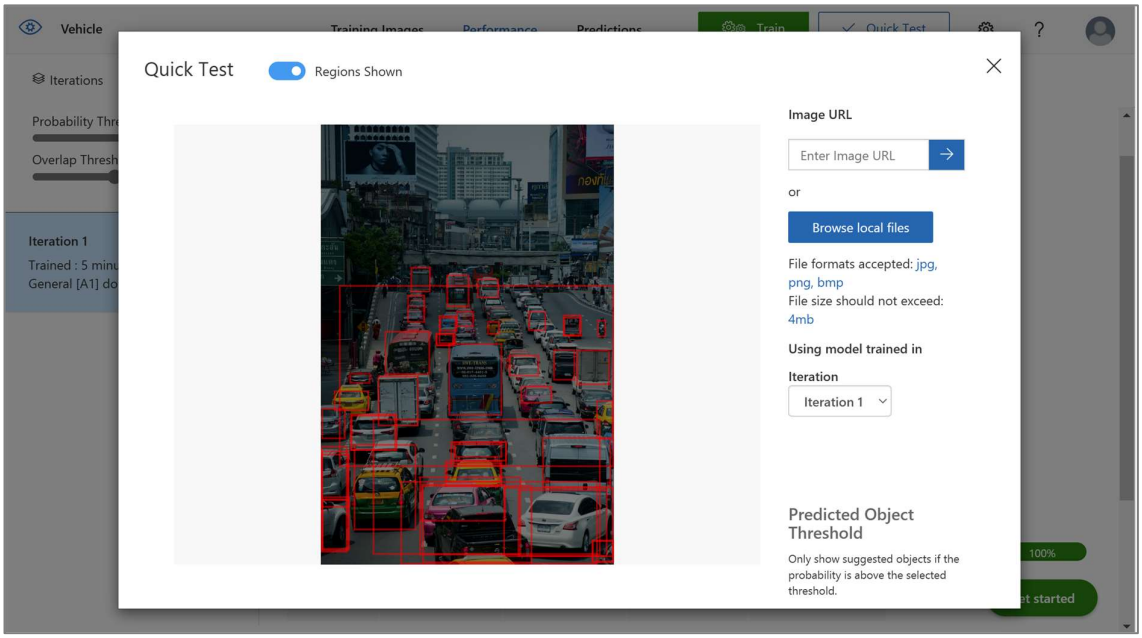


Fig 15 Test Results

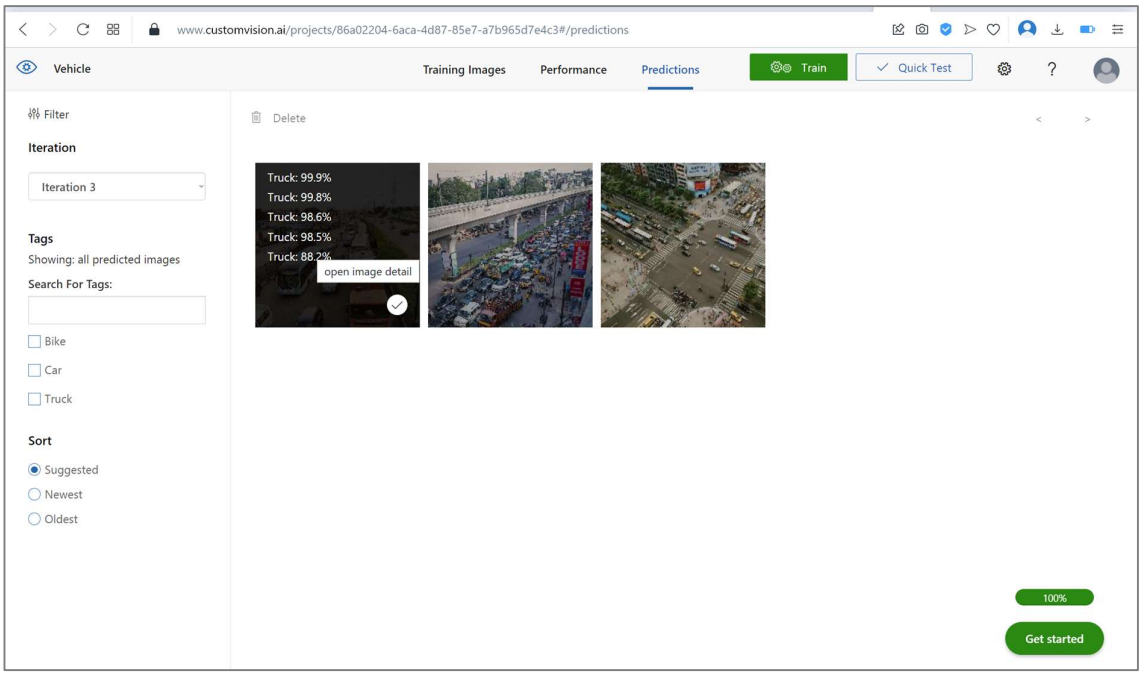


Fig 16 Test results

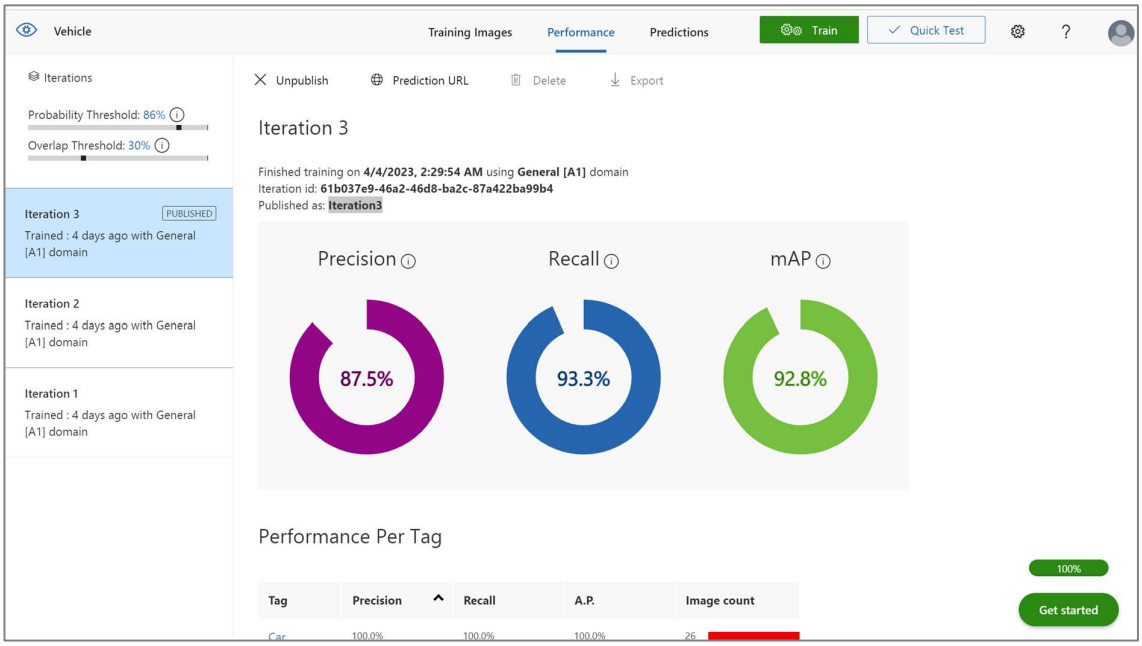


Fig 17 Performance of the model

Test based on the training

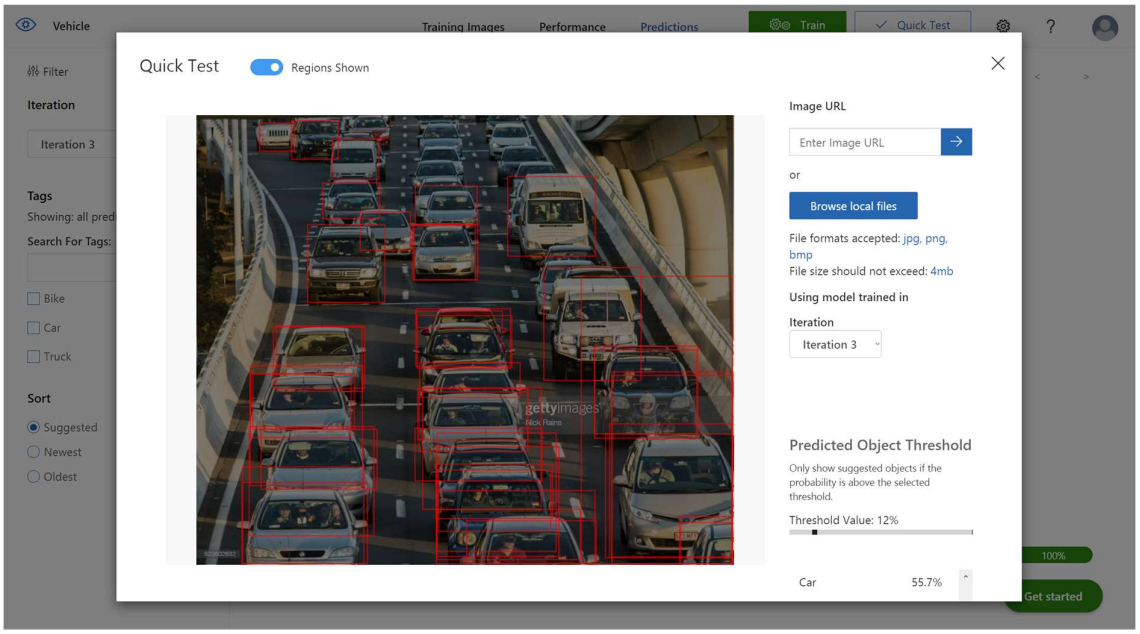


Fig 18 Quick Test

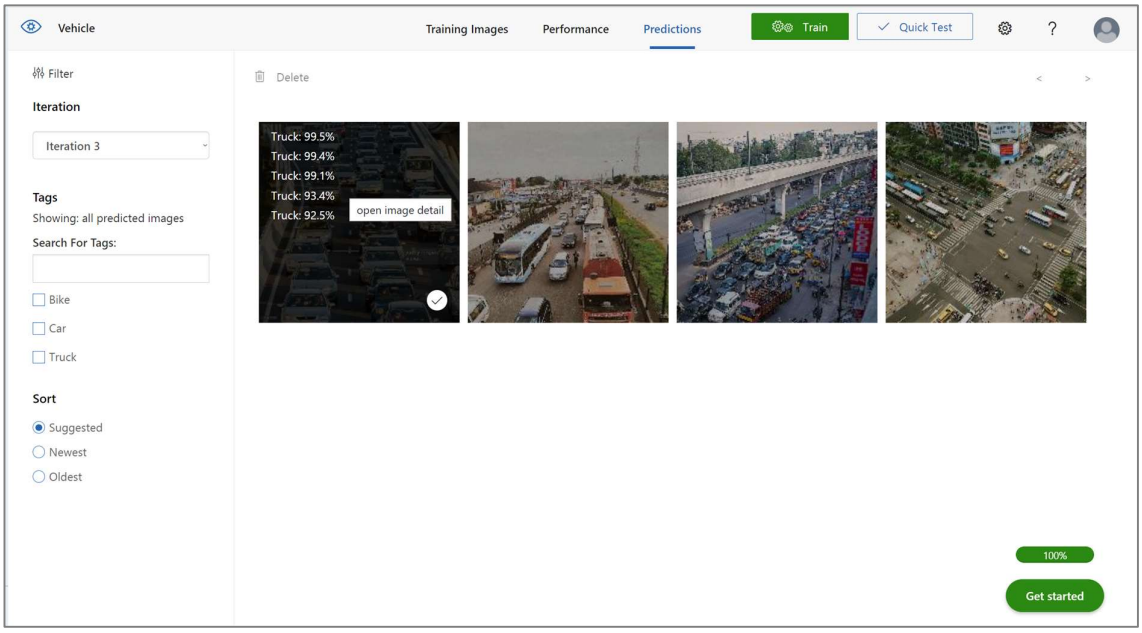


Fig 19 Test Performance

Some cars are incorrectly captured as Trucks. Those where retagged as Cars.

Works Cited

Anon., 2023. *Microsoft.* [Online]
Available at: https://learn.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/quickstarts/image-classification?tabs=visual-studio&pivots=programming-language-python
[Accessed 8 April 2023].

Anon., 2023. *pexels.* [Online]
Available at: https://www.pexels.com/search/Bike/
[Accessed April 2023].

Anon., 2023. *pexels.* [Online]
Available at: https://www.pexels.com/search/Truck/
[Accessed April 2023].

Anon., 2023. *Pexels.* [Online]
Available at: https://www.pexels.com/search/car/
[Accessed April 2023].

Anon., n.d.
http://www.eecs.qmul.ac.uk/~sgg/QMUL_Junction_Datasets/Junction/Junction.html.
[Online]
Available at:
http://www.eecs.qmul.ac.uk/~sgg/QMUL_Junction_Datasets/Junction/Junction.html

Nakisa, Dr Bahareh, 2023. *Great Learning.* [Online]
Available at: https://learn.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/get-started-build-detector

Object Classification:

We will create a new project under custom vision AI for object classification by using Python SDK

Import all libraries related to Custom Vision from Azure Cognitiveservices.vision

```
from azure.cognitiveservices.vision.customvision.training import CustomVisionTrainingClient
from azure.cognitiveservices.vision.customvision.prediction import CustomVisionPredictionClient
from azure.cognitiveservices.vision.customvision.training.models import ImageFileCreateBatch, ImageFileCreateEntry, Region
from msrest.authentication import ApiKeyCredentials
import os, time, uuid
```

Update the credentials like train and prediction key and resource id

```
# Replace with valid values
ENDPOINT = "https://custvisiontask4.cognitiveservices.azure.com/"
training_key = "499fbdd9093f4b578db913630e0beda8"
prediction_key = "1777acc8e1d94784bbb8e4d128949795"
prediction_resource_id = "/subscriptions/c094e16e-df8d-4a15-bbc5-0dd2487e788c/resourceGroups/Task3NS/providers/Microsoft.Cognitiv
```

Create the client interface by updating the credentials.

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

Create the project name for Object detection.

```
publish_iteration_name = "Iteration1"

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)
```

```
Creating project...
```

```
obj_detection_domain = next(domain for domain in trainer.get_domains() if domain.type =="ObjectDetection" and \
                            domain.name =="General")
print(obj_detection_domain)
```

```
{'additional_properties': {}, 'id': 'da2e3a8a-40a5-4171-82f4-58522f70fbc1', 'name': 'General', 'type': 'ObjectDetection', 'expo
rtable': False, 'enabled': True, 'exportable_platforms': [], 'model_information': None}
```

Tag the images. Here we use 3 sets of vehicle images to process train and test of the model. Tag all the three vehicles under the project ID created

```
# Make two tags in the new project
car_tag = trainer.create_tag(project.id, "Car")
bike_tag = trainer.create_tag(project.id, "Bike")
truck_tag = trainer.create_tag(project.id, "Truck")
```

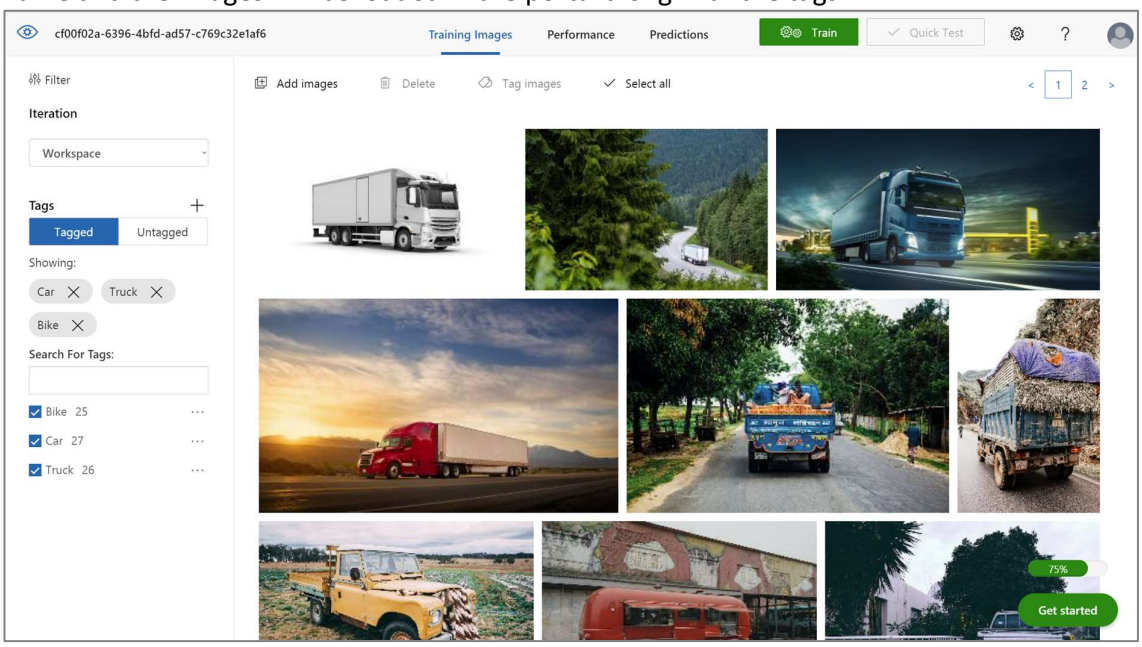
Upload the images for Object Detection

```
base_image_location = os.path.join (os.path.dirname("C:/Users/Dell/OneDrive/Documents/Deakin/Reference/"))

print("Adding images...")

tagged_images_with_regions = []


for file_name in Bike_image_regions.keys():
    print(file_name)
    x,y,w,h = Bike_image_regions[file_name]
    regions = [Region(tag_id = bike_tag_tag.id, left =x, top = y, width = w, height=h )]

    with open(base_image_location+ "images/Bike/"+file_name +".jpg", mode="rb")as image_contents:
        tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), \
                                                               regions=regions))


for file_name in Car_image_regions.keys():
    print(file_name)
    x,y,w,h = Car_image_regions[file_name]
    regions = [Region(tag_id = car_tag.id, left =x, top = y, width = w, height=h )]

    with open(base_image_location+ "images/Car/"+file_name +".jpg", mode="rb")as image_contents:
        tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), \
                                                               regions=regions))
for file_name in Truck_image_regions.keys():
```

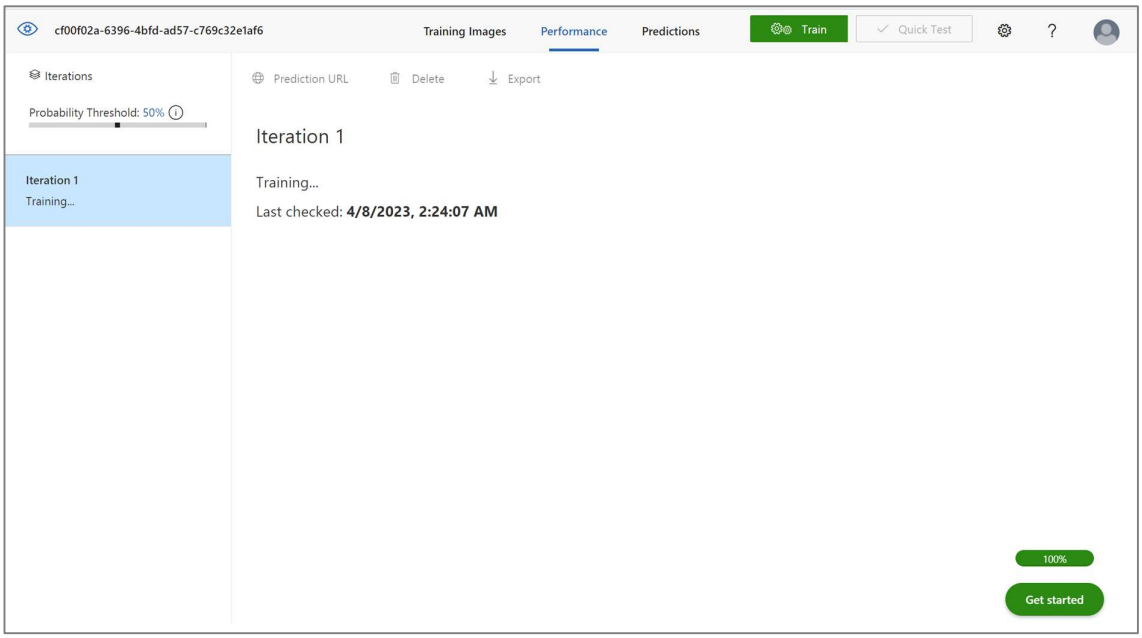
```
    with open(base_image_location+ "images/Bike/"+file_name +".jpg", mode="rb")as image_contents:
        tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), \
                                                               regions=regions))


for file_name in Car_image_regions.keys():
    print(file_name)
    x,y,w,h = Car_image_regions[file_name]
    regions = [Region(tag_id = car_tag.id, left =x, top = y, width = w, height=h )]

    with open(base_image_location+ "images/Car/"+file_name +".jpg", mode="rb")as image_contents:
        tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), \
                                                               regions=regions))
for file_name in Truck_image_regions.keys():
    print(file_name)
    x,y,w,h = Truck_image_regions[file_name]
    regions = [Region(tag_id = truck_tag.id, left =x, top = y, width = w, height=h )]

    with open(base_image_location+ "images/Truck/"+file_name +".jpg", mode="rb")as image_contents:
        tagged_images_with_regions.append(ImageFileCreateEntry(name=file_name, contents=image_contents.read(), \
                                                               regions=regions))



upload_result = trainer.create_images_from_files(project.id, images = tagged_images_with_regions)
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...
```

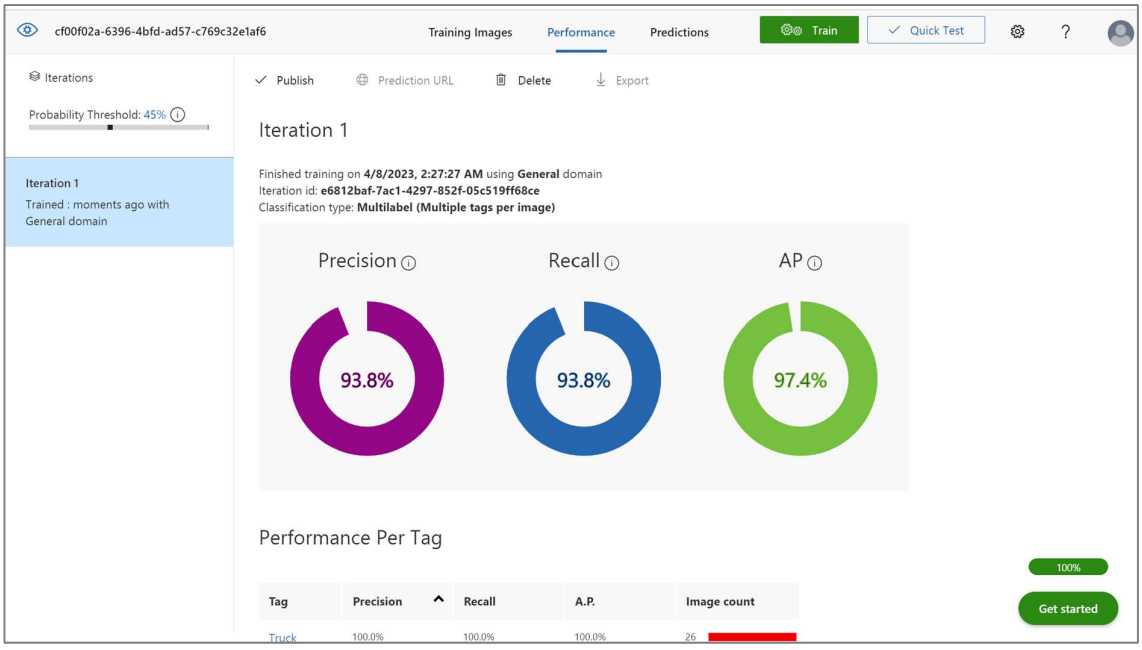
Once the images are uploaded, check the same in Custom_Vision.ai. We can find the project name and the images will be loaded in the portal along with the tags.

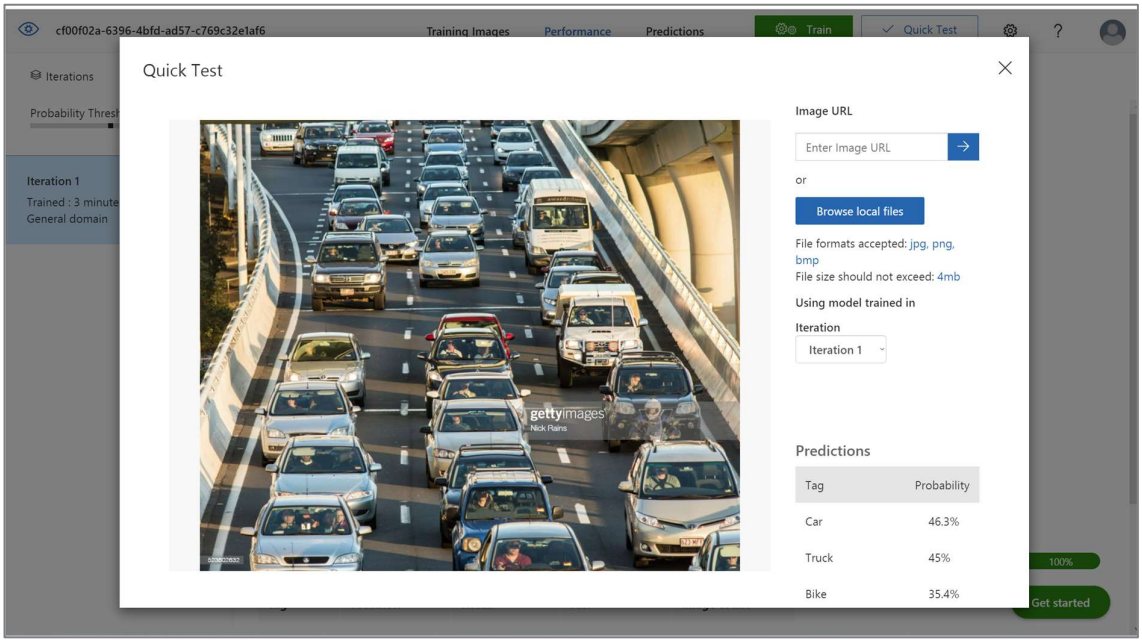


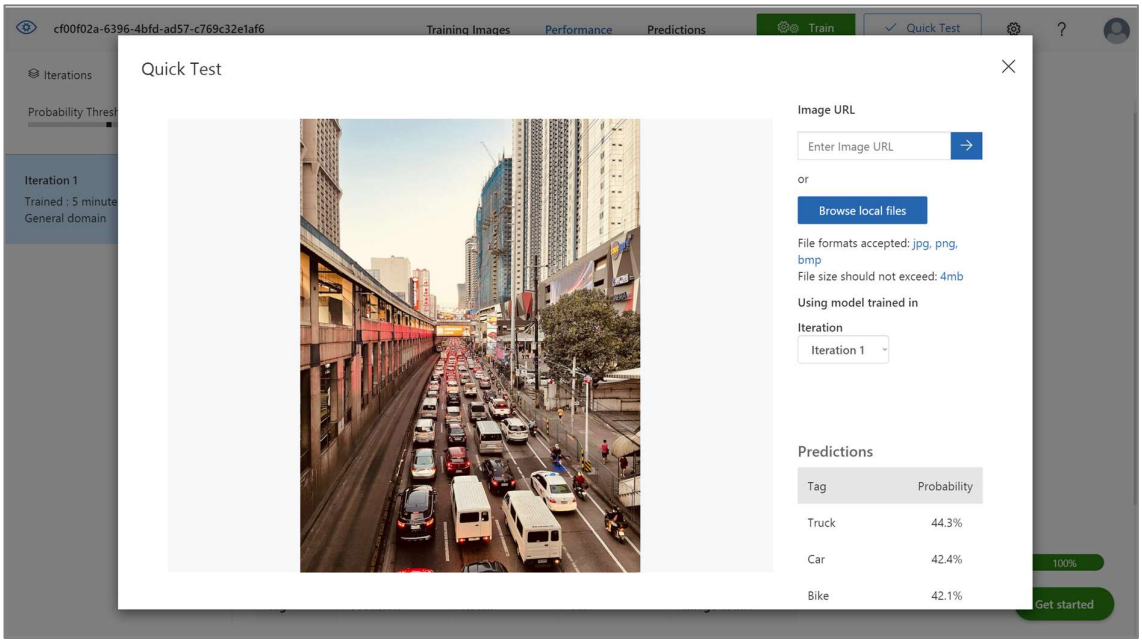
Once the images are loaded, Train the model in custom vision.ai.



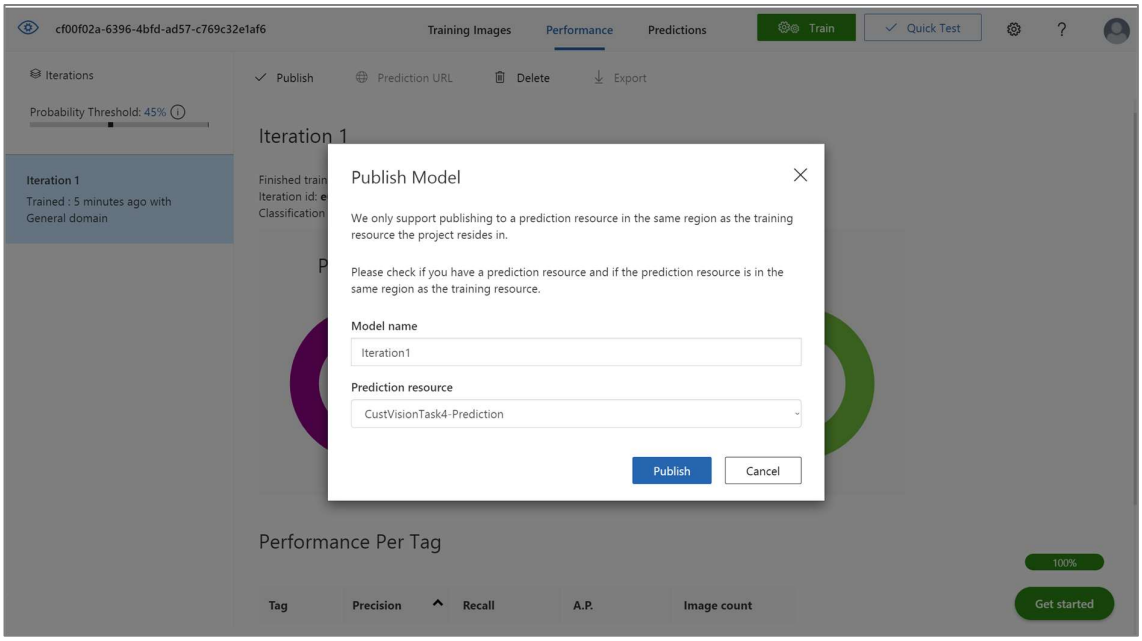
After training the model , we could find the model has classified each objects with model performance as 98% for Recall and precision. After validating the performance of the model, then the quick test are performed to validate the same tags can rightly identify the vehicles in traffic.

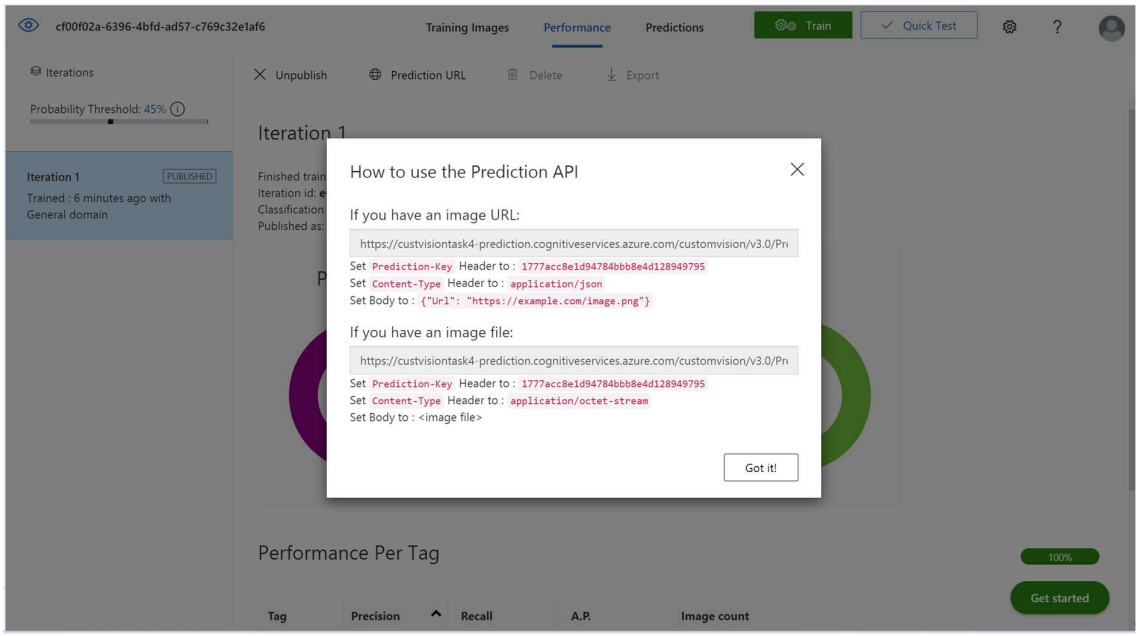


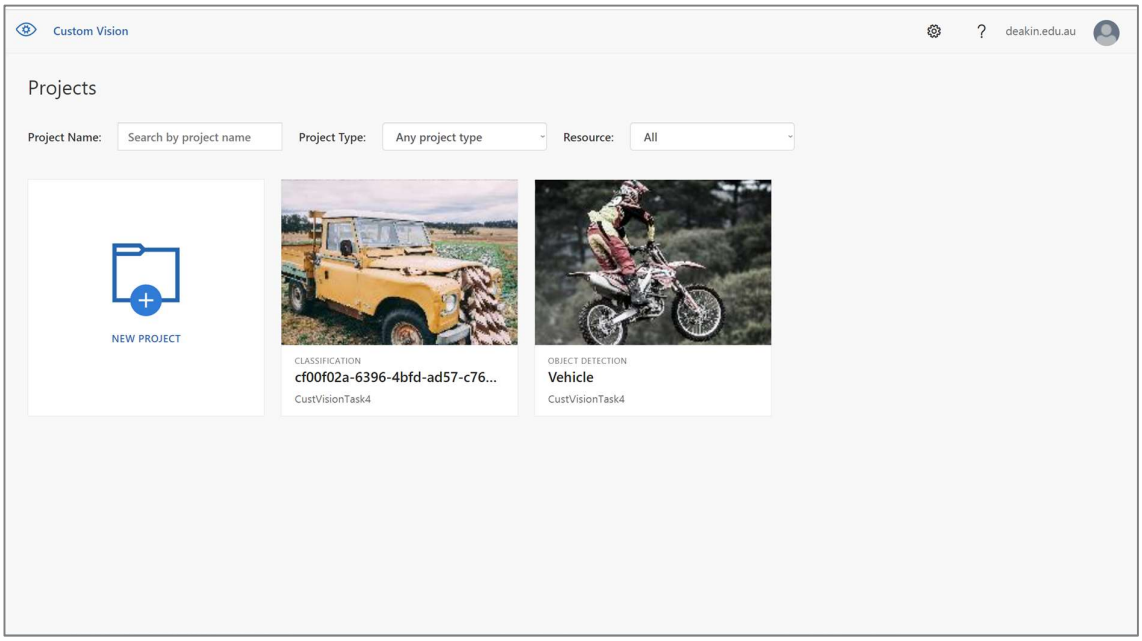




Once the train and test are giving good performance, next we will be publishing the model.







Video file to Frames

We have used the UK Live camera for traffic detection from website

```
# Vehicle Detection

VEHICLE_END_POINT="https://southcentralus.api.cognitive.microsoft.com/customvision/v3.0/Prediction/a67015ae-609e-46f7-bda5-5026e464745f/detect/iterations/Iteration1/url "

VEHICLE_PREDICTION_KEY="cdd99b37a8494380bfb5af6a243a9aaf"
```

Import all libraries

```
import os
import cv2
import numpy as np
import json
import pandas as pd
import asyncio
import aiohttp
import fnmatch
import matplotlib.pyplot as plt
import random
import textwrap
import datetime
from PIL import Image
import time
```

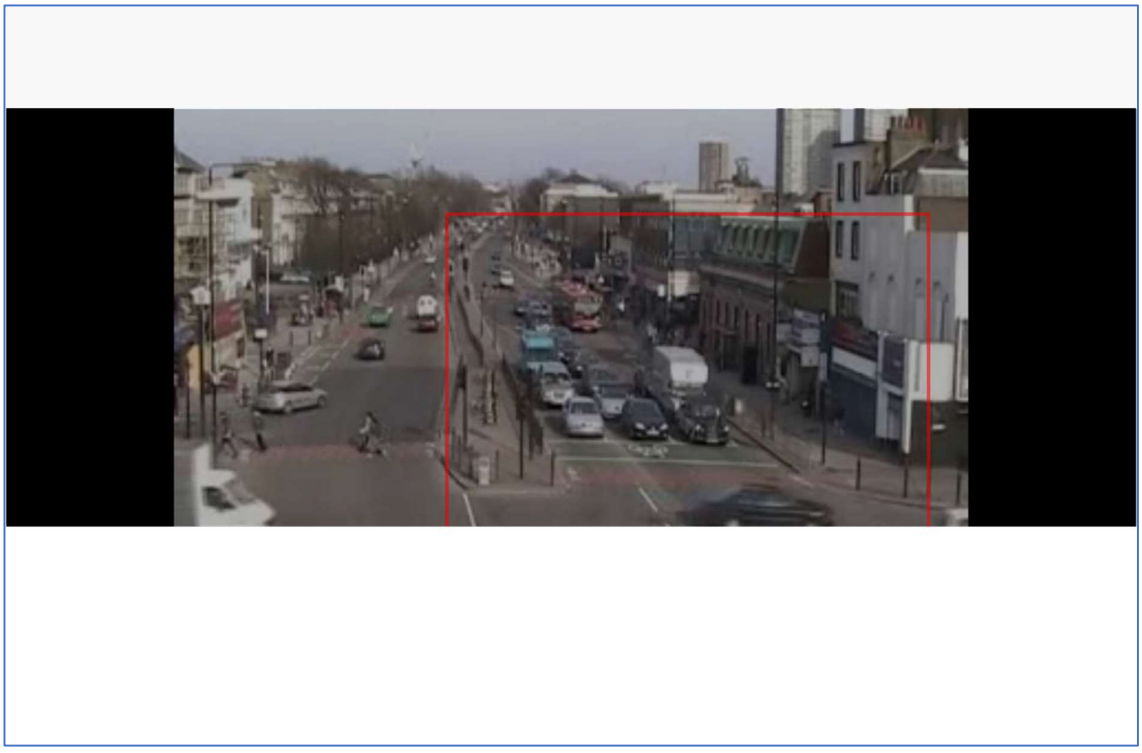
```
POST_URL = GlobalVariables.VEHICLE_END_POINT

HEADERS = {'Prediction-key':GlobalVariables.VEHICLE_PREDICTION_KEY, "Content-Type" : "application/json"}

MAX_CONNECTION = 100

WIDTH = 0

HEIGHT = 0
```

Saving the video as images

```
# Saving processed video as image

images =[]
byteImages = []
vidObj = cv2.VideoCapture("Output/output_video.mp4")
count = 0
success = 1
currentDir = os.getcwd()
if not os.path.isdir("frames"):
    os.mkdir("frames")
    while success:
        success, image = vidObj.read()
        cv2.imwrite("frames/frame%d.jpg" % count,image)
        count +-1

image_num =(len(fnmatch.filter(os.listdir(os.getcwd()+ "/frames"), '*.jpg')))

#deploy images from frames directory
i = 0
for i in range (image_num):
    if i%75==0:
        a=plt.imread("frames/frame%d.jpg" % i)
        plt.imshow(a)
        plt.show()
```

We got around 8000 images from the video and they have been detected in the frames in custom vision

Test on the Image framed



Works Cited

Anon., 2023. *Microsoft.* [Online]
Available at: https://learn.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/quickstarts/image-classification?tabs=visual-studio&pivots=programming-language-python
[Accessed 8 April 2023].

Anon., 2023. *pexels.* [Online]
Available at: https://www.pexels.com/search/Bike/
[Accessed April 2023].

Anon., 2023. *pexels.* [Online]
Available at: https://www.pexels.com/search/Truck/
[Accessed April 2023].

Anon., 2023. *Pexels.* [Online]
Available at: https://www.pexels.com/search/car/
[Accessed April 2023].

Anon., n.d.
http://www.eecs.qmul.ac.uk/~sgg/QMUL_Junction_Datasets/Junction/Junction.html.
[Online]
Available at:
http://www.eecs.qmul.ac.uk/~sgg/QMUL_Junction_Datasets/Junction/Junction.html

Nakisa, Dr Bahareh, 2023. *Great Learning.* [Online]
Available at: https://learn.microsoft.com/en-us/azure/cognitive-services/custom-vision-service/get-started-build-detector