Module 23
AWS Rekognition – Objects Detection

Overview

This is a continuation of the AWS Rekognition service of Module 22 where you used the service to detect text in images. In this exercise you will use the service to detect objects inside images. Objects detection allows you to solve different types of problems. To get an idea of what can be accomplished, do this manual exercise:

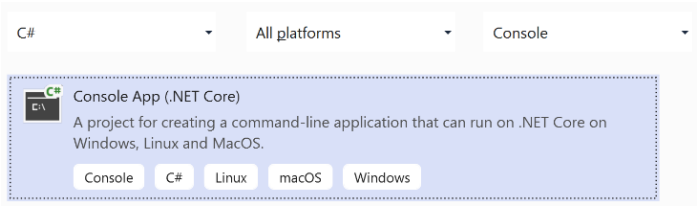
1. Login to the AWS console and go to the Rekognition service.
2. Under Demos, click on **Object and scene detection**. Under **Results** on the right, notice the objects classification results with degrees of confidence score value.
3. Expand the Request and Response sections to see how the JSON of the request and response look like.
4. Click on the city skyline picture.
5. You can also upload a picture of your choosing and see how the model identifies its content.

Steps

1. Open the Visual Studio solution named `AWSRekognitionSolution` that you created in an earlier module.

2. Right-click on `AWSRekognitionSolution` and choose menu Add → New Project…

3. Under Visual C#, select a project template of type Console App (.NET Core). Name the project `DetectObjectsInImage`.



4. Right-click on project `DetectObjectsInImage` and select menu **Set as StartUp Project**.

5. To the `DetectObjectsInImage` project, add references to the following 2 DLLs:

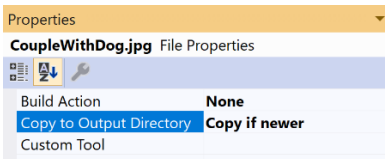
   AWSSDK.Core.dll

AWSSDK.Rekognition.dll

6. Right-click on project `DetectObjectsInImage` and choose menu **Add → New Folder**. Name the folder **Images**.

7. Right-click on the **Images** folder and choose menu **Add → Existing Item…**. Select all 5 images provided with this module (in a folder named "Images") and add them.

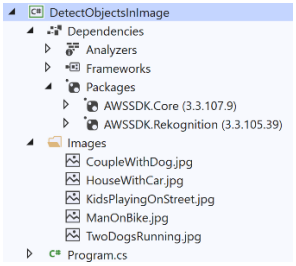
8. Click on each image and in the Properties pane, set property Copy to Output Directory to Copy if newer.



What this does is that when you build the project, a folder called Images will be created in the **bin** directory with all 5 images copies to it.

This way, in your code you can use "Image/<image_name>" as the relative path of each image.

9. Your project structure should now look like this:



10. Complete the program to do the following:

    a) For each image in the **Images** folder, use the AWS Rekognition SDK to detect the objects in the image. Start first by setting confidence to 90% (by that you instruct the SDK to return objects it is 90%+ certain of their identities).

```
const float MIN_CONFIDENCE = 90F;
```

    Print each image and the detected objects under it with the confidence in between parenthesis (example output shown in Figure 1 below):

    Classes/methods you need to use:

    `DetectLabelsRequest`

```
DetectLabelsResponse
DetectLabelsAsync
```

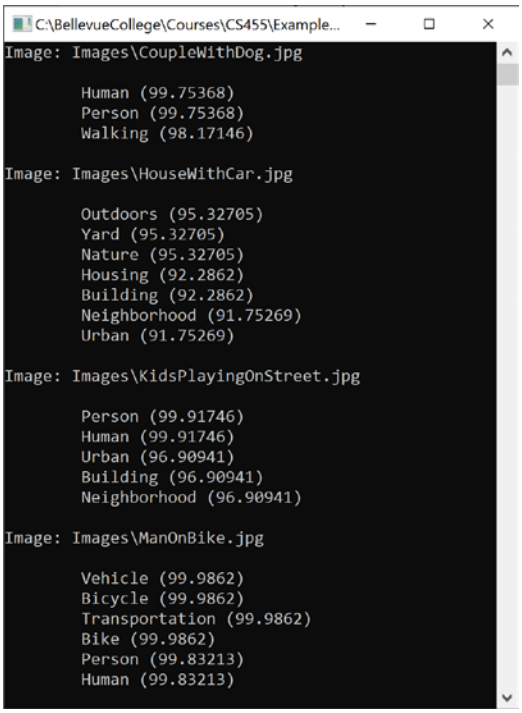


Figure 1: Output with Min Confidence = 90.

b) Relax the Min Confidence to 75 and re-run the program. Notice that more objects will be returned.

Challenge Exercise

1. Assume that you are using this service in a real-world security application where several camera installed in the lobby of a building analyzes if a person is carrying a weapon when he/she enters the building. Repeat the above for all pictures in folder **ChallengeImages**.

   Can your program identify people carrying weapons and alert security?
   (You may need to first inspect the labels the service is returning in order to know the string to check for to classify the image as carrying weapon or not).

   Get a few additional pictures from the web (some with people carrying weapons in a building and others with people with no weapons). Test the accuracy of your program in detecting weapons. Try pictures where the weapon is partially concealed or partially visible (a more likely real-world scenario).

2. If you are curious, experiment with other types of pictures with different content. See what the model gives you back.

What to Submit:

Nothing to submit for this module.