Analyze Images with Azure AI Vision

Azure AI Vision is an artificial intelligence capability that enables software systems to interpret visual input by analyzing images. In Microsoft Azure, the **Vision** Azure AI service provides pre-built models for common computer vision tasks, including analysis of images to suggest captions and tags, detection of common objects and people.

Clone the repository for this course

If you have not already cloned the **Azure AI Vision** code repository to the environment where you're working on this lab, follow these steps to do so. Otherwise, open the cloned folder in Visual Studio Code.

1. Start Visual Studio Code.
2. Open the palette (SHIFT+CTRL+P) and run a **Git: Clone** command to clone the `https://github.com/MicrosoftLearning/mslearn-ai-vision` repository to a local folder (it doesn't matter which folder).
3. When the repository has been cloned, open the folder in Visual Studio Code.

4. Wait while additional files are installed to support the C# code projects in the repo.

> **Note**: If you are prompted to add required assets to build and debug, select **Not Now**. If you are prompted with the Message *Detected an Azure Function Project in folder*, you can safely close that message.

Provision an Azure AI Services resource

If you don't already have one in your subscription, you'll need to provision an **Azure AI Services** resource.

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. Select **Create a resource**.
3. In the search bar, search for *Azure AI services*, select **Azure AI Services**, and create an Azure AI services multi-service account resource with the following settings:

   - **Subscription**: *Your Azure subscription*
   - **Resource group**: *Choose or create a resource group (if you are using a restricted subscription, you may not have permission to create a new resource group - use the one provided)*
   - **Region**: *Choose from East US, West US, France Central, Korea Central, North Europe, Southeast Asia, West Europe, or East Asia**
   - **Name**: *Enter a unique name*
   - **Pricing tier**: Standard S0

   *Azure AI Vision 4.0 full feature sets are currently only available in these regions.
4. Select the required checkboxes and create the resource.
5. Wait for deployment to complete, and then view the deployment details.
6. When the resource has been deployed, go to it and view its **Keys and Endpoint** page. You will need the endpoint and one of the keys from this page in the next procedure.

Prepare to use the Azure AI Vision SDK

In this exercise, you'll complete a partially implemented client application that uses the Azure AI Vision SDK to analyze images.

> **Note**: You can choose to use the SDK for either **C#** or **Python**. In the steps below, perform the actions appropriate for your preferred language.

1. In Visual Studio Code, in the **Explorer** pane, browse to the **Labfiles/01-analyze-images** folder and expand the **C-Sharp** or **Python** folder depending on your language preference.

2. Right-click the **image-analysis** folder and open an integrated terminal. Then install the Azure AI Vision SDK package by running the appropriate command for your language preference:

   C#

   Code Copy

   ```
   dotnet add package Azure.AI.Vision.ImageAnalysis -v 1.0.0-beta.3
   ```

   Note: If you are prompted to install dev kit extensions, you can safely close the message.

   Python

   Code Copy

   ```
   pip install azure-ai-vision-imageanalysis==1.0.0b3
   ```

   Tip: If you are doing this lab on your own machine, you'll also need to install `matplotlib` and `pillow`.

3. View the contents of the **image-analysis** folder, and note that it contains a file for configuration settings:

   - **C#**: appsettings.json
   - **Python**: .env

   Open the configuration file and update the configuration values it contains to reflect the **endpoint** and an authentication **key** for your Azure AI services resource. Save your changes.

4. Note that the **image-analysis** folder contains a code file for the client application:

   - **C#**: Program.cs
   - **Python**: image-analysis.py

   Open the code file and at the top, under the existing namespace references, find the comment **Import namespaces**. Then, under this comment, add the following language-specific code to import the namespaces you will need to use the Azure AI Vision SDK:

   C#

   Code Copy

   ```
   // Import namespaces
   using Azure.AI.Vision.ImageAnalysis;
   ```

   Python

   Code Copy

   ```
   # import namespaces
   from azure.ai.vision.imageanalysis import ImageAnalysisClient
   from azure.ai.vision.imageanalysis.models import VisualFeatures
   from azure.core.credentials import AzureKeyCredential
   ```

View the images you will analyze

In this exercise, you will use the Azure AI Vision service to analyze multiple images.

1. In Visual Studio Code, expand the **image-analysis** folder and the **images** folder it contains.
2. Select each of the image files in turn to view them in Visual Studio Code.

Analyze an image to suggest a caption

Now you're ready to use the SDK to call the Vision service and analyze an image.

1. In the code file for your client application (**Program.cs** or **image-analysis.py**), in the **Main** function, note that the code to load the configuration settings has been provided. Then find the comment **Authenticate Azure AI Vision client**. Then, under this comment, add the following language-specific code to create and authenticate a Azure AI Vision client object:

C#

Code Copy

```
// Authenticate Azure AI Vision client
ImageAnalysisClient client = new ImageAnalysisClient(
    new Uri(aiSvcEndpoint),
    new AzureKeyCredential(aiSvcKey));
```

Python

Code Copy

```
# Authenticate Azure AI Vision client
cv_client = ImageAnalysisClient(
    endpoint=ai_endpoint,
    credential=AzureKeyCredential(ai_key)
)
```

1. In the **Main** function, under the code you just added, note that the code specifies the path to an image file and then passes the image path to two other functions (**AnalyzeImage** and **BackgroundForeground**). These functions are not yet fully implemented.

2. In the **AnalyzeImage** function, under the comment **Get result with specify features to be retrieved**, add the following code:

C#

Code Copy

```
// Get result with specified features to be retrieved
ImageAnalysisResult result = client.Analyze(
    BinaryData.FromStream(stream),
    VisualFeatures.Caption |
    VisualFeatures.DenseCaptions |
    VisualFeatures.Objects |
    VisualFeatures.Tags |
    VisualFeatures.People);
```

Python

Code Copy

```
# Get result with specified features to be retrieved
result = cv_client.analyze(
    image_data=image_data,
    visual_features=[
        VisualFeatures.CAPTION,
        VisualFeatures.DENSE_CAPTIONS,
        VisualFeatures.TAGS,
        VisualFeatures.OBJECTS,
        VisualFeatures.PEOPLE],
)
```

1. In the **AnalyzeImage** function, under the comment **Display analysis results**, add the following code (including the comments indicating where you will add more code later.):

C#

Code Copy

```
// Display analysis results
// Get image captions
if (result.Caption.Text != null)
{
    Console.WriteLine(" Caption:");
    Console.WriteLine($"   \"{result.Caption.Text}\", Confidence 
{result.Caption.Confidence:0.00}\n");
}

// Get image dense captions
Console.WriteLine(" Dense Captions:");
foreach (DenseCaption denseCaption in result.DenseCaptions.Values)
{
    Console.WriteLine($"   Caption: '{denseCaption.Text}', Confidence: 
{denseCaption.Confidence:0.00}");
}

// Get image tags


// Get objects in the image


// Get people in the image
```

Python

Code Copy

```
# Display analysis results
# Get image captions
if result.caption is not None:
    print("\nCaption:")
    print(" Caption: '{}' (confidence: {:.2f}%)".format(result.caption.text,
result.caption.confidence * 100))

# Get image dense captions
if result.dense_captions is not None:
    print("\nDense Captions:")
    for caption in result.dense_captions.list:
        print(" Caption: '{}' (confidence: {:.2f}%)".format(caption.text, caption.confidence *
100))

# Get image tags


# Get objects in the image


# Get people in the image
```

1. Save your changes and return to the integrated terminal for the **image-analysis** folder, and enter the following command to run the program with the argument **images/street.jpg**:

C#

Code Copy

```
dotnet run images/street.jpg
```

Python

Code Copy

```
python image-analysis.py images/street.jpg
```

1. Observe the output, which should include a suggested caption for the **street.jpg** image.
2. Run the program again, this time with the argument **images/building.jpg** to see the caption that gets generated for the **building.jpg** image.
3. Repeat the previous step to generate a caption for the **images/person.jpg** file.

Get suggested tags for an image

It can sometimes be useful to identify relevant *tags* that provide clues about the contents of an image.

1. In the **AnalyzeImage** function, under the comment **Get image tags**, add the following code:

C#

Code Copy

```
// Get image tags
if (result.Tags.Values.Count > 0)
{
    Console.WriteLine($"\n Tags:");
    foreach (DetectedTag tag in result.Tags.Values)
    {
        Console.WriteLine($"   '{tag.Name}', Confidence: {tag.Confidence:F2}");
    }
}
```

Python

Code Copy

```
# Get image tags
if result.tags is not None:
    print("\nTags:")
    for tag in result.tags.list:
        print(" Tag: '{}' (confidence: {:.2f}%)".format(tag.name, tag.confidence * 100))
```

1. Save your changes and run the program once for each of the image files in the **images** folder, observing that in addition to the image caption, a list of suggested tags is displayed.

Detect and locate objects in an image

Object detection is a specific form of computer vision in which individual objects within an image are identified and their location indicated by a bounding box..

1. In the **AnalyzeImage** function, under the comment **Get objects in the image**, add the following code:

C#

Code Copy

```
// Get objects in the image
if (result.Objects.Values.Count > 0)
{
    Console.WriteLine(" Objects:");

    // Prepare image for drawing
    stream.Close();
    System.Drawing.Image image = System.Drawing.Image.FromFile(imageFile);
    Graphics graphics = Graphics.FromImage(image);
    Pen pen = new Pen(Color.Cyan, 3);
    Font font = new Font("Arial", 16);
    SolidBrush brush = new SolidBrush(Color.WhiteSmoke);

    foreach (DetectedObject detectedObject in result.Objects.Values)
    {
        Console.WriteLine($"   \"{detectedObject.Tags[0].Name}\"");

        // Draw object bounding box
        var r = detectedObject.BoundingBox;
        Rectangle rect = new Rectangle(r.X, r.Y, r.Width, r.Height);
        graphics.DrawRectangle(pen, rect);
        graphics.DrawString(detectedObject.Tags[0].Name,font,brush,(float)r.X, (float)r.Y);
    }

    // Save annotated image
    String output_file = "objects.jpg";
    image.Save(output_file);
    Console.WriteLine("  Results saved in " + output_file + "\n");
}
```

Python

Code Copy

```
# Get objects in the image
if result.objects is not None:
    print("\nObjects in image:")

    # Prepare image for drawing
    image = Image.open(image_filename)
    fig = plt.figure(figsize=(image.width/100, image.height/100))
    plt.axis('off')
    draw = ImageDraw.Draw(image)
    color = 'cyan'

    for detected_object in result.objects.list:
        # Print object name
        print(" {} (confidence: {:.2f}%)".format(detected_object.tags[0].name, 
detected_object.tags[0].confidence * 100))

        # Draw object bounding box
        r = detected_object.bounding_box
        bounding_box = ((r.x, r.y), (r.x + r.width, r.y + r.height))
        draw.rectangle(bounding_box, outline=color, width=3)
        plt.annotate(detected_object.tags[0].name,(r.x, r.y), backgroundcolor=color)

    # Save annotated image
    plt.imshow(image)
    plt.tight_layout(pad=0)
    outputfile = 'objects.jpg'
    fig.savefig(outputfile)
    print('  Results saved in', outputfile)
```

1. Save your changes and run the program once for each of the image files in the **images** folder, observing any objects that are detected. After each run, view the **objects.jpg** file that is generated in the same folder as your code file to see the annotated objects.

Detect and locate people in an image

People detection is a specific form of computer vision in which individual people within an image are identified and their location indicated by a bounding box.

1. In the **AnalyzeImage** function, under the comment **Get people in the image**, add the following code:

C#

Code Copy

```
// Get people in the image
if (result.People.Values.Count > 0)
{
    Console.WriteLine($" People:");

    // Prepare image for drawing
    System.Drawing.Image image = System.Drawing.Image.FromFile(imageFile);
    Graphics graphics = Graphics.FromImage(image);
    Pen pen = new Pen(Color.Cyan, 3);
    Font font = new Font("Arial", 16);
    SolidBrush brush = new SolidBrush(Color.WhiteSmoke);

    foreach (DetectedPerson person in result.People.Values)
    {
        // Draw object bounding box
        var r = person.BoundingBox;
        Rectangle rect = new Rectangle(r.X, r.Y, r.Width, r.Height);
        graphics.DrawRectangle(pen, rect);

        // Return the confidence of the person detected
        //Console.WriteLine($"   Bounding box {person.BoundingBox.ToString()}, Confidence:
{person.Confidence:F2}");
    }

    // Save annotated image
    String output_file = "persons.jpg";
    image.Save(output_file);
    Console.WriteLine("  Results saved in " + output_file + "\n");
}
```

Python

Code Copy

```
# Get people in the image
if result.people is not None:
    print("\nPeople in image:")

    # Prepare image for drawing
    image = Image.open(image_filename)
    fig = plt.figure(figsize=(image.width/100, image.height/100))
    plt.axis('off')
    draw = ImageDraw.Draw(image)
    color = 'cyan'

    for detected_people in result.people.list:
        # Draw object bounding box
        r = detected_people.bounding_box
        bounding_box = ((r.x, r.y), (r.x + r.width, r.y + r.height))
        draw.rectangle(bounding_box, outline=color, width=3)

        # Return the confidence of the person detected
        #print(" {} (confidence: {:.2f}%)".format(detected_people.bounding_box,
detected_people.confidence * 100))

    # Save annotated image
    plt.imshow(image)
    plt.tight_layout(pad=0)
    outputfile = 'people.jpg'
    fig.savefig(outputfile)
    print('  Results saved in', outputfile)
```

1. (Optional) Uncomment the **Console.Writeline** command under the **Return the confidence of the person detected** section to review the confidence level returned that a person was detected at a particular position of the image.
2. Save your changes and run the program once for each of the image files in the **images** folder, observing any objects that are detected. After each run, view the **objects.jpg** file that is generated in the same folder as your code file to see the annotated objects.

> **Note**: In the preceding tasks, you used a single method to analyze the image, and then incrementally added code to parse and display the results. The SDK also provides individual methods for suggesting captions, identifying tags, detecting objects, and so on - meaning that you can use the most appropriate method to return only the information you need, reducing the size of the data payload that needs to be returned. See the .NET SDK documentation or Python SDK documentation for more details.

Clean up resources

If you're not using the Azure resources created in this lab for other training modules, you can delete them to avoid incurring further charges. Here's how:

1. Open the Azure portal at `https://portal.azure.com`, and sign in using the Microsoft account associated with your Azure subscription.
2. In the top search bar, search for *Azure AI services multi-service account*, and select the Azure AI services multi-service account resource you created in this lab.
3. On the resource page, select **Delete** and follow the instructions to delete the resource.

More information

In this exercise, you explored some of the image analysis and manipulation capabilities of the Azure AI Vision service. The service also includes capabilities for detecting objects and people, and other computer vision tasks.

For more information about using the **Azure AI Vision** service, see the Azure AI Vision documentation.