El objetivo de este Laboratorio es explorar una aplicación básica de Windows que utiliza la API de Computer Vision para crear un proyecto, agregarle etiquetas, cargar imágenes, entrenar el proyecto, obtener la URL de punto final de predicción predeterminada para el proyecto y usar el punto final para prueba programáticamente una imagen. Puede usar este ejemplo de código abierto como plantilla para crear su propia aplicación para Windows utilizando la Custom Vision API.

Requisitos previos

Requisitos de la plataforma

Este ejemplo ha sido probado usando .NET Framework usando Visual Studio 2017, Community Edition

Biblioteca de cliente de entrenamiento

Es posible que deba instalar la biblioteca del cliente según su configuración en Visual Studio. La forma más fácil de obtener la biblioteca cliente de capacitación es instalar el paquete Microsoft.Cognitive.CustomVision.Training de nuget.

Puede instalarlo a través del administrador de paquetes de Visual Studio. Acceda al administrador de paquetes navegando a través de:

Tools -> Nuget Package Manager -> Package Manager Console

En esa consola, agregue el nuget con:

Install-Package Microsoft.Cognitive.CustomVision.Training -Version 1.0.0

La clave de la API de capacitación

También necesita tener una clave API de entrenamiento. La clave API de capacitación le permite crear, administrar y capacitar programáticamente proyectos de Vision personalizada. Que se muestra en el video de creación del API de Computer Vision en Microsoft Azure.

Maratona Bots

Laboratorio: Creación de una aplicación de Custom Vision

Paso 1: cree una aplicación de consola y prepare la clave de capacitación y las imágenes necesarias para el ejemplo.

Inicie Visual Studio 2017, Community Edition, abra la solución de Visual Studio denominada **CustomVision.Sample.sln** en el subdirectorio donde se encuentra esta práctica de laboratorio:

Resources/Starter/CustomVision.Sample/CustomVision.Sample.sln

Este código define y llama a dos métodos auxiliares. El método llamado GetTrainingKeyprepara la clave de entrenamiento. El que se llama LoadImagesFromDisk carga dos juegos de imágenes que este ejemplo usa para entrenar el proyecto, y una imagen de prueba que el ejemplo carga para demostrar el uso del punto final de predicción predeterminado. Al abrir el proyecto, se debe mostrar el siguiente código desde la línea 35:

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Threading;
using Microsoft.Cognitive.CustomVision;

namespace CustomVision.Sample
{
   class Program
   {
      private static List<MemoryStream> MbikesImages;

      private static List<MemoryStream> RbikesImages;

      private static MemoryStream testImage;

      static void Main(string[] args)
      {
         // Puede Agregar la clave de entrenamiento aquí, pasarla en la línea de comandos o escribirla
cuando el programa se ejecute
         string trainingKey = GetTrainingKey("<your key here>", args);

         // Crear la API, pasar un objeto de credenciales que contenga la clave de entrenamiento
         TrainingApiCredentials trainingCredentials = new TrainingApiCredentials(trainingKey);
         TrainingApi trainingApi = new TrainingApi(trainingCredentials);

      }

      private static string GetTrainingKey(string trainingKey, string[] args)
      {
         if (string.IsNullOrWhiteSpace(trainingKey) || trainingKey.Equals("<your key here>"))
         {
            if (args.Length >= 1)
            {
               trainingKey = args[0];
```

```
            }

            while (string.IsNullOrWhiteSpace(trainingKey) || trainingKey.Length != 32)
            {
                Console.Write("Enter your training key: ");
                trainingKey = Console.ReadLine();
            }
            Console.WriteLine();
        }

        return trainingKey;
    }

    private static void LoadImagesFromDisk()
    {
        // Esto carga las imágenes que se cargarán desde el disco a la memoria
        MbikesImages = Directory.GetFiles(@"..\..\..\..\Images\Mountain").Select(f => new
MemoryStream(File.ReadAllBytes(f))).ToList();
        RbikesImages = Directory.GetFiles(@"..\..\..\..\Images\Racing").Select(f => new
MemoryStream(File.ReadAllBytes(f))).ToList();
        testImage = new MemoryStream(File.ReadAllBytes(@"..\..\..\..\Images\test\bike2.jpg"));

    }
  }
}
```

Paso 2: Crea un proyecto de servicio de Custom Vision

Para crear un nuevo proyecto del Servicio de Custom Vision denominado "Tipo de bicicleta", agregue el siguiente código en el cuerpo del Main()método después de la llamada anew TrainingApi().

```
 // Crear un nuevo proyecto
Console.WriteLine("Creating new project:");
var project = trainingApi.CreateProject("Bike Type");
```

Paso 3: Agrega etiquetas a tu proyecto

Para agregar etiquetas a su proyecto, inserte el siguiente código después de la llamada a CreateProject("Bike Type");.

```
// Hacer dos etiquetas en el nuevo proyecto
var MbikesTag = trainingApi.CreateTag(project.Id, "Mountain");
var RbikesTag = trainingApi.CreateTag(project.Id, "Racing");
```

Paso 4: Sube imágenes al proyecto

Para agregar las imágenes que tenemos en la memoria del proyecto, inserte el siguiente código después de la llamada al CreateTag(project.Id, "Racing")método.

```
 // Agregar algunas imágenes a las etiquetas
Console.WriteLine("\\tUploading images");
LoadImagesFromDisk();

// Las imágenes se pueden cargar una a la vez
foreach (var image in MbikesImages)
{
  trainingApi.CreateImagesFromData(project.Id, image, new List< string> () {
MbikesTag.Id.ToString() });
}

// O subido en un solo lote
trainingApi.CreateImagesFromData(project.Id, RbikesImages, new List< Guid> () {
RbikesTag.Id });
```

Paso 5: Capacitar al proyecto

Ahora que hemos agregado etiquetas e imágenes al proyecto, podemos entrenarlo. Inserte el siguiente código después del final del código que agregó en el paso anterior. Esto crea la primera iteración en el proyecto. Luego podemos marcar esta iteración como la iteración predeterminada.

```
 // Ahora hay imágenes con etiquetas empezar a entrenar el proyecto
Console.WriteLine("\\tTraining");
var iteration = trainingApi.TrainProject(project.Id);

// La iteración devuelta estará en curso, y se puede consultar periódicamente para ver
cuándo ha completado
while (iteration.Status == "Training")
{
   Thread.Sleep(1000);

   // Volver a consultar la iteración para obtener su estado actualizado
   iteration = trainingApi.GetIteration(project.Id, iteration.Id);
}

// La iteración está ahora entrenada. Convertirlo en el punto final del proyecto
predeterminado
iteration.IsDefault = true;
trainingApi.UpdateIteration(project.Id, iteration.Id, iteration);
Console.WriteLine("Done!\\n");
```

Paso 6: Obtenga y use el punto final predeterminado de predicción

Ahora estamos listos para usar el modelo de predicción. Primero obtenemos el punto final asociado con la iteración predeterminada. Luego enviamos una imagen de prueba al proyecto usando ese punto final. Inserta el código después del código de entrenamiento que acabas de ingresar.

```
 // Ahora hay un punto final entrenado, puede ser usado para hacer una predicción
// Obtener la clave de predicción, que se utiliza en lugar de la clave de entrenamiento cuando se hacen predicciones
var account = trainingApi.GetAccountInfo();
var predictionKey = account.Keys.PredictionKeys.PrimaryKey;
// Crear un extremo de predicción, pasar un objeto de credenciales de predicción que contenga la clave de predicción obtenida
PredictionEndpointCredentials predictionEndpointCredentials = new PredictionEndpointCredentials(predictionKey);
PredictionEndpoint endpoint = new PredictionEndpoint(predictionEndpointCredentials);
// Hacer una predicción contra el nuevo proyecto
Console.WriteLine("Making a prediction:");
var result = endpoint.PredictImage(project.Id, testImage);
// Bucle sobre cada predicción y escribir los resultados
foreach (var c in result.Predictions)
{
  Console.WriteLine($"\\t{c.Tag}: {c.Probability:P1}");
}
Console.ReadKey();
```

Paso 7: Ejecuta el ejemplo

Construye y ejecuta la solución. Se le pedirá que ingrese su clave API de capacitación en la aplicación de la consola cuando ejecute la solución, así que tenga esto listo. El entrenamiento y la predicción de las imágenes pueden tomar 2 minutos. Los resultados de predicción aparecen en la consola.